A PROJECT-BASED APPROACH FOR TEACHING NUMERICAL INTEGRATION IN CONTINUOUS SIMULATION

Yuzhong Shen
Masha Sosonkina

Department of Electrical and Computer Engineering
Old Dominion University
4700 Elkhorn Ave, Norfolk, VA, USA
{yshen,msosonki}@odu.edu

ABSTRACT

Differential equations are the main apparatus employed in continuous simulation for modeling dynamics in systems and processes, and an array of numerical integration methods is utilized to solve the differential equations. The standard practice for teaching numerical integration is to provide an expression of the derivative of the state variable with respect to time in a closed form and then find the solution using a numerical method. Many problems in the real-world cannot be represented using a simple differentiable function. It is desired to expose the students to a wide array of real-world problems. One powerful and promising way to accomplish this is through visualization aids. This paper presents a flow visualization project that exposes students to ocean flow visualization using real-world data. The project enables students to investigate a variety of numerical integration methods, analyze their performance, and explore various visualization and interaction techniques, thus improving their learning effectiveness.

Keywords: continuous simulation, numerical integration, flow visualization, streamline.

1 INTRODUCTION

Computer-based simulations imitate the operation of a real-world system or process over time by using models that are abstract representations of the key characteristics or behaviors of the real-world system or process. The results of computer-based modeling and simulation are used for managerial or technical decision making, saving cost and improving safety. The three major simulation paradigms are discrete event simulation, continuous simulation, and Monte Carlo simulation. Continuous simulation is a type of simulation in which state variables change with respect to time. The state variables in a continuous simulation can take on any value within the specified range and are updated based on the advance of a simulation clock. Continuous simulation is typically applied to natural sciences phenomena, such as biological, chemical, and environmental processes. Differential equations, both ordinary and partial, are heavily utilized in continuous simulation to model various natural phenomena, systems, and processes, such as analog circuits, fluid dynamics, ballistics and missile trajectory, numerical weather prediction (temperature, humidity, wind velocity, tide height, etc.), hydraulics, aircraft dynamics, hedge funds and insurance actuarial analysis, control systems (Brezinski and Wuytack, 2001; MathWorks, 2022; Zill, 2000).

To create a continuous simulation, a model for the system, natural phenomena, or processes, must be built first. The various variables (the unknowns of the problem) of interest have to be identified and the equations they satisfy need to be formulated. When these equations are too complicated, they must be simplified by neglecting the terms whose influence is small compared to that of the other terms. In many situations, these equations are ordinary or partial differential equations, or integral equations. Various numerical methods have been developed for solving differential equations, such as Euler method and higher-order Runge-Kutta

methods (Burden, Faires, and Burden, 2015; Klee and Allen, 2011). The Runge-Kutta methods developed by the German mathematicians Carl Runge and Wilhelm Kutta are considered as one of the most important landmarks of the development of numerical analysis in the 20th century. The well-known Runge-Kutta fourth-order four-stage (RK4) is considered a golden standard for numerical integration.

Project-based learning (PBL) is an instructional approach that emphasizes learning through the completion of a project. Students engage in a process of inquiry, research, and problem-solving, with the goal of developing a product or solution that addresses a real-world problem or challenge. PBL has been found to be an effective method of teaching and learning in a variety of educational settings (Guo et al., 2020; National Research Council, 2000, 2012a, 2012c, 2018; Project Kaleidoscope, 2002). PBL has been found to have numerous benefits for students (Almulla, 2020; National Research Council, 2003, 2012b). PBL provides students with a more engaging and meaningful learning experience, as they are motivated by the challenge of solving real-world problems and creating something tangible. PBL promotes the development of essential skills such as critical thinking, communication, collaboration, and creativity. PBL helps students to develop a deeper understanding of academic content by applying it to real-world situations. PBL fosters students' personal and social development by providing opportunities to work in teams, develop leadership skills, and reflect on their learning. While PBL has many benefits, designing and implementing PBL projects can be time-consuming and challenging for teachers, especially those who are not familiar with the method (National Research Council, 2012a; Wagner & Kingston, 2022). PBL projects require significant resources, such as technology, materials, and space, which may not be available in all schools. In addition, PBL may not be appropriate for all content areas, particularly those that require a more direct instruction approach. Therefore, effective PBL projects must be carefully designed by taking into account a wide range of factors, such as academic content, target audience, cost, complexity, and extensibility.

This paper presents the design of a project-based learning approach for teaching numerical integration in a continuous simulation course. The project utilizes real data from the northeast Pacific Ocean to generate streamlines using RK-4 numerical integration. Applying numerical integration to real-world data to generate visualization of ocean flow allows students to apply abstract numerical methods to real-world problems and obtain immediate visual feedback that reflects the performance of numerical integration. The project is succinct and self-contained, i.e., it does not require installation of any additional software dependencies, improving learning efficacy. The project is not only used to teach earn numerical integration theories and methods but also to demonstrate efficient software engineering practices, making it useful for multiple courses. The remainder of this paper is organized as follows. Section 2 discusses the background. Section 3 briefly describes the two courses for which the flow visualization project will be utilized. Section 4 presents the details of the flow visualization project in terms of numerical integration and software design. Section 5 discusses several use cases of the project. Finally, Section 6 discusses future work and concludes this paper.

2 BACKGROUND

While Old Dominion University's Bachelor of Science degree program in Modeling and Simulation Engineering (BSMSE) program has national reputation and has graduated very high-quality students who have gone onto top tier graduate programs and high paying careers, it suffered the common pitfalls of niche programs, primarily in the form of insufficient advertising/recruitment to properly educate the community about the program. In addition, universities are anticipating a large decline in enrollment starting in 2025 due to drastic drops in birth rates following the recession in 2007 (Kline, 2019). Numerous universities have started to announce the elimination or consolidation of degrees and programs, starting prior to COVID-19. For the sustainability of such programs, no matter the strength or merits of the program, it becomes beneficial to align these programs with well-established programs. This allows the sharing of resources to include personnel and available course offerings. Old Dominion University made a decision to transform the BSMSE degree into a major under the Bachelor of Science in Computer Engineering (BSCpE) degree. This takes advantage of the strong computational engineering focus of the existing program to benefit the computer engineering program while providing the M&S engineering students a

stronger grounding in engineering with a clear application domain. The curriculum was then developed so that graduates satisfy computer engineering requirements while providing a depth of knowledge in M&S. A description of the curriculum and how it fits within a Computer Engineering degree program was presented in (Leathrum, Shen, and Gonzalez, 2021). This program consolidation also strengthens Old Dominion University's commitment in Virginia's Tech Talent Investment Program that stipulates a certain number of graduates in Computer Science and Computer Engineering each year.

The BSCpE degree now consists of two majors: Computer Engineering (CpE), and Modeling and Simulation Engineering (MSE). The two majors share a common computer engineering core that consists of 10 courses and each major has its own core courses. The MSE major core comprises four courses: ECE 306 Discrete Systems Modeling and Simulation, ECE 320 Continuous Systems Modeling and Simulation, ECE 348 Simulation Software Design, and ECE 406 Computer Graphics and Visualization. Their offering semesters and dependency are illustrated in Figure **1**. This paper focuses on an integrative approach that utilizes various aspects of the same project for ECE 320 Continuous Systems Modeling and Simulation and ECE 406 Computer Graphics and Visualization to offer students a more cohesive learning experience.

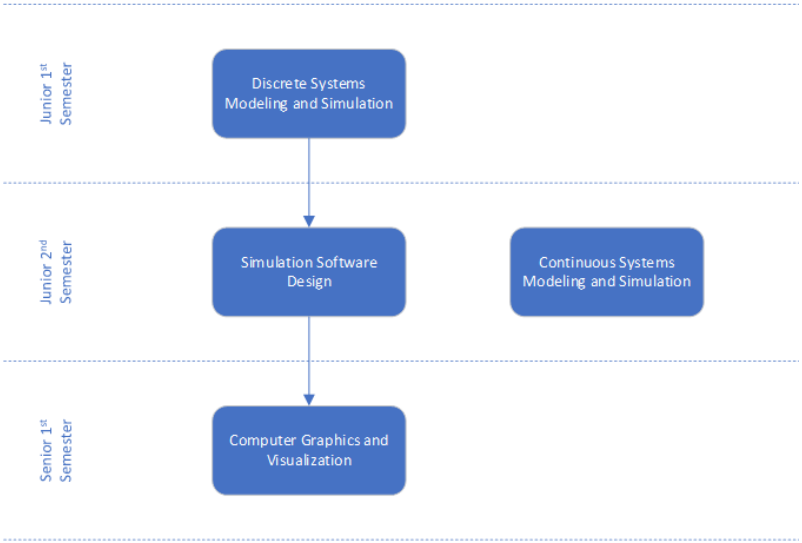


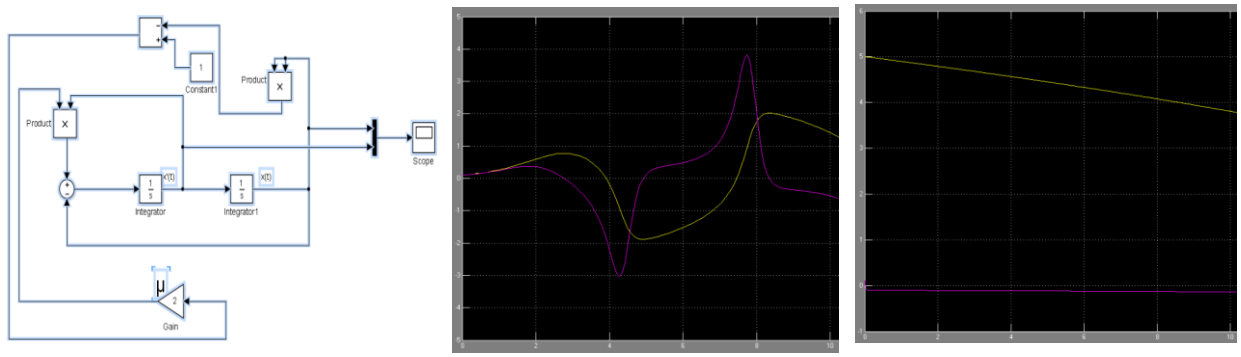
Figure 1: The core courses for the Modeling and Simulation Engineering major under Computer Engineering, in addition to the common Computer Engineering core.

3 AN INTEGRATIVE PEDAGOGY FOR COURSE INSTRUCTION

The flow visualization project proposed in this paper will be mainly utilized in two required courses: ECE 320 Continuous System Modeling and Simulation and ECE 406 Computer Graphics and Visualization. ECE 320 students will work on numerical integration aspects of the project while ECE 406 students will develop different visualization and interaction techniques for the project. This integrative approach provides students a more cohesive learning experience across courses and hands-on experience with real-world projects. This section provides a brief description of ECE 320 and ECE 406.

ECE 320 Continuous System Modeling and Simulation is a 3-credit course that provides an introduction to the fundamentals of modeling and simulating continuous-state, time-driven systems. Topics include differential equation representation of systems, formulation of state variable equations, model representation using block diagrams, stock-flow diagrams and bond graphs, numerical integration, and techniques for numerical solution of differential equations including the Taylor algorithm and the Runge-Kutta and Adams families of methods. Tools and environments for continuous simulations are introduced. Application domains include electrical systems, signals (including sampling), physical, and biological

systems. Although some experimentation procedures have been introduced into the course to make up for the separate laboratory course, which accompanied the MSIM version of ECE 320, they are not nearly enough to let students compare numerical simulations with observation results. In particular, in ECE 320, students experiment with a variety of input-parameter and initial-solution ranges and step sizes, which they feed into a Simulink simulation and observe the resulting solutions with a particular (set) of numerical methods. For example, consider a well-known van der Pol oscillator (van der Pol, 1926). Its Simulink diagram may be represented as in Figure 2(a).



(a) (b) (c)

Figure 2: A van del Pol Oscillator. (a) Simulink diagram of van der Pol oscillator with two first-order integrator blocks. Oscillator simulation of 10-second behavior: (b) oscillating when the initial state variable value is small, and (c) steady decreasing when initial value is large. Displacement curves are in yellow, velocity in magenta.

With the value of μ equal to 2, the oscillator will change the behavior depending on the initial value of the state variable. The results simulated with the Runge-Kutta 4 method available in MATLAB are shown in Figure 2(b) and (c) for the initial x = 0.1 and x = 5, respectively, with the same initial velocity value of 0.1. This typical assignment type already lets students appreciate the different nature of trajectories based on many initial factors. However, in such assignments students have no access to steering and feedback mechanisms without redoing the entire simulation. Hence, a visualization or other real-physical system simulation would be valuable and innovative for the Continuous Simulation course to expand its toolkit of systems to simulate and to make up for the experimental observations, which were done in a lab course.

ECE 406 Computer Graphics and Visualization is a 3-credit course that provides a practical treatment of computer graphics and visualization with emphasis on modeling and simulation applications. It covers digital image and signal processing basics such as sampling and discrete Fourier transform, computer graphics fundamentals, visualization principles, and software architecture for visualization in modeling and simulation. Written communication and information literacy skills are stressed in this course. It teaches OpenGL programming for developing interactive visualization for modeling and simulation applications. Unity game engine is utilized to illustrate advanced concepts and techniques.

4 FLOW VISUALIZATION FOR TEACHING NUMERICAL INTEGRATION

4.1 Flow Visualization Using Streamlines

Visualization is heavily utilized in modeling and simulation and other fields to gain insights to a phenomenon or scientific experiments or communicate a message to the general public. Scientific visualization is an important area of visualization that renders scientific data to enable scientists and engineers to understand, illustrate, and gain insight from their data. Fluid mechanics is a branch of physics concerned with the mechanics of fluids and the forces on them (White, 2011). It has applications in a wide range of disciplines, such as mechanical, aerospace, civil, chemical and biomedical engineering.

Computational fluid dynamics (CFD) is branch of fluid mechanics that uses numerical integration, differential equations, data structures and algorithms to analyze and solve problems that involve fluid flows. Visualization is heavily used in CFD to perceive qualitative and quantitative information from the massive data or experiments. Therefore, flow visualization is an ideal topic that permeates the two courses ECE 320 Continuous System Modeling and Simulation and ECE 406 Computer Graphics and Visualization in a cohesive way.

The velocity vector field of a flow in 3D space is the most critical information in CFD and flow visualization. Various visualization techniques have been developed to illustrate 3D vectors, such as oriented lines, oriented *glyphs* (both 2D and 3D markers, e.g., arrows or similar markers), and complex vector visualization (Hansen & Johnson, 2004). However, direct visualization of a velocity vector field is rarely useful due to the quantity of the data and the low, raw level of data. Advanced flow visualization techniques have been developed to gain insight of the velocity vector field and the characteristics of the flow. In particular, *streamlines*, *streaklines*, and *pathlines* are heavily utilized in CFD and flow visualization (Hansen & Johnson, 2004). *Streamlines* are family of curves whose tangent vectors constitute the velocity of the velocity vector field. *Streaklines* are the loci of points of all the fluid particles that have passed continuously through a particular spatial point in the past. *Pathlines* are the trajectories that individual fluid particles follow. In steady flow, where the velocity vector field does not change with time, the streamlines, streaklines, and pathlines coincide. To provide a real-world project that is meaningful and manageable to students, this paper focuses on visualization of steady flows.

Streamlines are defined by

$$\frac{d\vec{x}}{ds} \times \vec{u}(\vec{x}) = \mathbf{0}, \tag{1}$$

where × denotes the cross product and $x(s)$ is the parametric representation (with parameter *s*) of just one streamline at one time instant (Hansen & Johnson, 2004; Liu et al., 2006; Ugeng et al., 1996).

The position on the streamline can be calculated as (Hansen & Johnson, 2004; Liu et al., 2006; Ugeng et al., 1996)

$$\vec{x}_S = \int_{t_0}^{t_1} \vec{u}(\vec{x}) ds. \tag{2}$$

For steady flows, the parameter *s* in (2) can be replaced by time *t*, as the velocity vector field does not change with time and is the same for all times. Equation (1) can be solved numerically using the Runge-Kutta Order 4 method (RK4) (Liu et al., 2006; Ugeng et al., 1996) as follows.

$$\vec{x}_{n+1} = \vec{x}_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + 4k_4)h, \tag{3}$$

$$t_{n+1} = t_n + h, \tag{4}$$

for $n = 0,1,2,3, \ldots$, and an appropriate step size *h* using

$$k_1 = \vec{u}(t_n, \vec{x}_n), \tag{5}$$

$$k_2 = \vec{u}\left(t_n + \frac{h}{2}, \vec{x}_n + h\frac{k_1}{2}\right), \tag{6}$$

$$k_3 = \vec{u}\left(t_n + \frac{h}{2}, \vec{x}_n + h\frac{k_2}{2}\right), \tag{7}$$

$$k_4 = \vec{u}(t_n + h, \vec{x}_n + hk_3). \tag{8}$$

4.2 Project-based Learning of Numerical Integration Using Flow Visualization

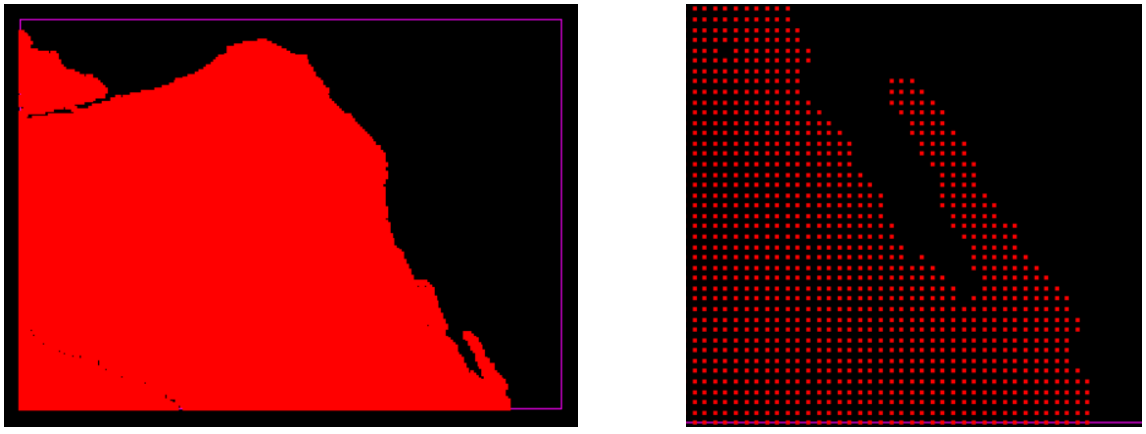
The standard practice for teaching numerical integration is to provide an expression of the derivative of the state variable with respect to time in a closed form and then find the solution using a numerical method,

e.g., RK4, one step at a time on the trajectory sought as a solution. However, many problems in the real-world cannot be represented using a simple differentiable function. For example, many electrical circuit problems may entail in algebraic loops or some flow problems, such as artery flows, may produce bifurcating trajectories. Therefore, it is desired to expose the students to a wide array of real-world problems with their complexities. We deem that an appropriate way to accomplish this is through visualization aids.

To address the above challenge, we have developed a project for visualization of ocean flow with streamlines using real-world data. The benefits of this approach are at least threefold: 1) students will understand the nature of real-world problems from multiple aspects, e.g., data sources, specific applications and state variables; 2) students will understand the process of solving a real-world problem from scratch; and 3) the visualization may be adapted, steered, and controlled to demonstrate the problems presenting difficulties to the numerical solution techniques.

The region of visualization is the Northeast Pacific Ocean (180° W ~ 78° W, 20° N ~ 62° N) as shown in Figure **3**(a). The entire region is defined on a 468 × 337 grid; Figure **3**(b) shows a blow-up view of part of the grid. The grid size along the longitudinal direction is 0.175°, while that along the latitudinal direction is 0.125°. The input data are an ocean flow velocity vector field defined on the grid. It is worth noting that while ocean is a 3D space, its velocity vector field is dominated by the horizontal components that are parallel to the ocean surface, i.e., these along longitudinal and latitudinal directions; the vertical component (perpendicular to the ocean surface) is very small and thus can be ignored. Therefore, the flow velocity vector field is provided in the form of two binary files: one for the longitudinal direction and the other for the latitudinal direction. Binary formats are used for the input files to reduce file sizes and increase file access speed. Each velocity component is represented as a 32-bit floating point number. The goals for the entire project include the following.

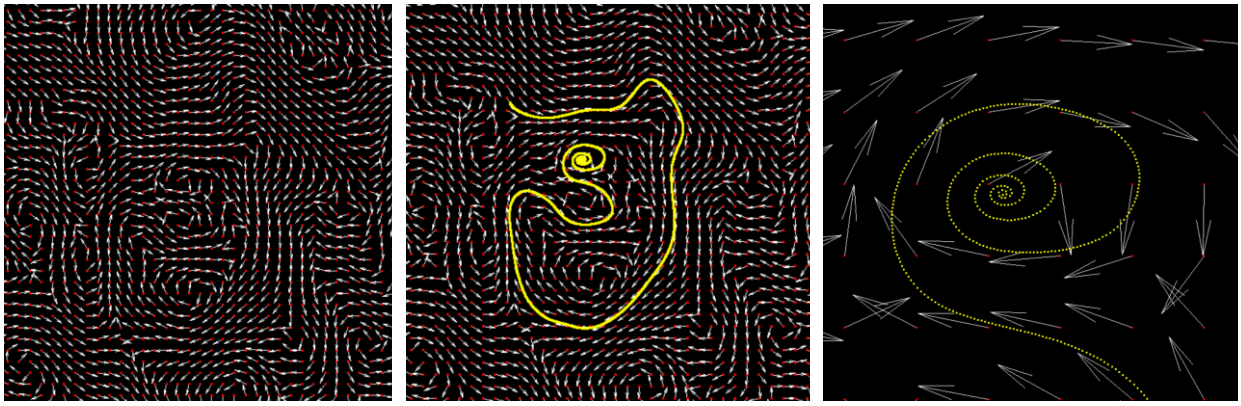
1. Generate a visualization of the velocity vector field visualization using 2D glyphs (2D arrows) (ECE 406).
2. Generate streamline (pathline) visualization using the Runge-Kutta numerical integration (ECE 320 and ECE 406).
3. Investigate the stability and performance of RK4 by changing the step size (ECE 320).
4. Explore other numerical integration techniques, e.g., multistep methods, and examine their performance (ECE 320).
5. Generate appropriate user interactions to facilitate usage of the project, e.g., pan, zoom, center, selection of starting points for streamlines (ECE 406).



(a) (b)

Figure 3: Visualization region and its grid. (a) The Northeast Pacific Ocean (180° W ~ 78° W, 20° N ~ 62° N) is utilized for flow visualization in this project. The entire region is defined on a 468 × 337 grid. The red color is used to represent the region. (b) A blow-up view of part of the grid.

Figure 3(a) shows the velocity vector field using 2D arrows, one type of 2D glyphs. Figure 3(b) shows one of streamline generated. The user first selects a start point (by pressing the 's' key) and the program then computes the points on the streamline using RK4 integration and connects them to form the final streamline. Note that Figure 3(b) is a blow-up view of Figure 3(a) showing the individual points on the streamline (or pathline for steady flows). It can be clearly seen that an attracting focus is located at the center of the area being visualized. Visualization facilitates detection of important features in the flow, such as critical points (saddle, attracting node, repelling node, center, attracting focus, and repelling focus) (Wang et al., 2013, 2022), which otherwise would be difficult to detect directly from the raw velocity vector field. Figure **4** shows some results of the visualization.



(a) (b) (c)

Figure 4: Flow visualization. (a) Velocity visualization using 2D glyphs. (b) Streamline visualization. (c) Pathline visualization.

To improve teaching effectiveness, the scope of sample projects or software code provided to students must be concise, e.g., number of lines of code, number of classes, etc. Overly complicated samples often overwhelm students and distracting students from learning the most essential theories and concepts. The flow visualization project discussed in this paper was developed from scratch using C++ and OpenGL. The entire project is very succinct with only a total of 753 lines of C++ code (C++20), including 4 header files and 4 source code files. It mainly utilizes two classes: Simulation and its derived class FlowViz, as illustrated by the class diagram shown in Figure **5**. The Simulation class is a base class that include interfaces for setting basic parameters of an interactive simulation and execution of the simulation, such as SetClock() for setting the simulation clock and Run() for starting the simulation. The FlowViz class implements a specific simulation for flow visualization and contains methods for numerical integration, visualization of the flow, and user interactions. The entire implementation of the RK4 integration, including 2D linear interpolation of the velocity based on the grid, is contained in the function GeneratePoints_RK4(), which has only 162 lines of code. To understand the implementation of the RK4 integration and analyze its performance, students only need to read the GeneratePoints_RK4() function. Similarly, students can replace this function with another function that implements a different numerical integration method, e.g., GeneratePoints_Euler() for Euler method.

The entire project is self-contained, i.e., it contains all the dependencies needed, and does not need any additional third-party libraries. This greatly reduces the burden on students, improving their learning efficacy. The project makes use of advanced software design techniques, such as polymorphism with virtual methods, nested methods using lambda expression, resulting in a clean software architecture to facilitate software reuse, maintenance, and revision.

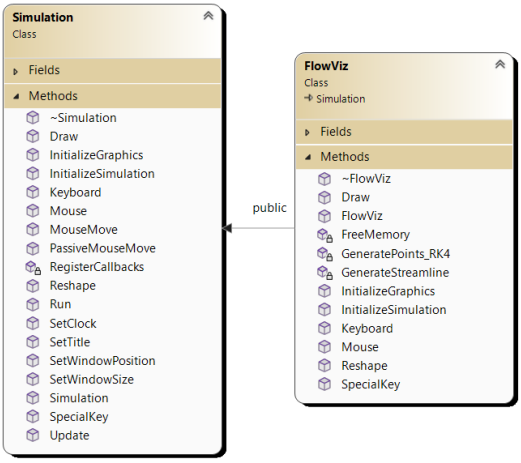
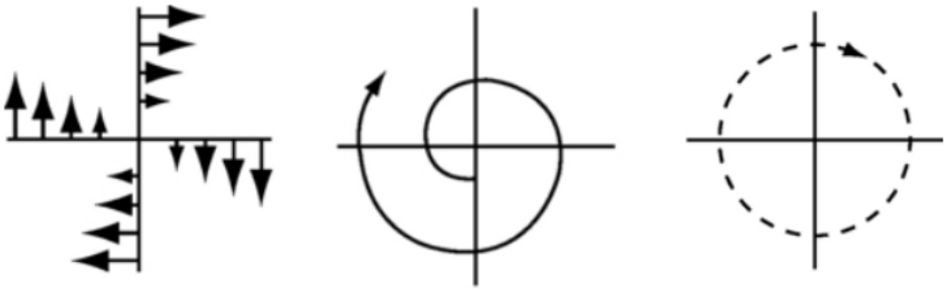


Figure 5: The class diagram of the Flow Visualization project.

5 DISCUSSION

The Flow Visualization can be utilized in ECE 320 Continuous System Modeling and Simulation and ECE 406 Computer Graphics and Visualization in multiple ways. Here two possible use cases are briefly described.

ECE 320 teaches different numerical integration methods and analysis of the performance of these methods in terms of their convergence, accuracy (round-off, truncation and discretization error), stability, and computational complexity. Figure **6** shows the results of generating streamlines for a rotational velocity vector field using different numerical integration techniques, namely Euler method and RK4. It can be clearly seen that Euler method does not converge (Figure **6**(a)) while RK4 does (Figure **6**(b)). In ECE 320, this project may be assigned along with research paper presentation activity taking place mid-semester. Specifically, students who are interested in graphics and visualization may be inclined to do this project and present their findings on the RK methods and possible flow trajectories.



(a) (b) (c)

Figure 6: Computation of streamline using different numerical methods. (a) a rotational velocity vector field. (b) The result generated using Euler method. (c) The result generated using RK4 integration. Drawn after (Hansen & Johnson, 2004)**.**

Students in ECE 406 can implement different types of visualization techniques and investigate their effectiveness. Different types of 2D (or 3D) glyphs can be utilized. For example, the magnitude of a velocity vector can be represented by the length of the 2D glyph (e.g., arrow) representing that vector. Another possible option for representing the magnitude of a vector is to use colormaps, e.g., red for a larger value while blue for a smaller one. Various colormap schemes can be explored. Different user interaction

techniques can be implemented. Currently, a right mouse button click is used to re-center the screen. Another re-centering option is to let user select a rectangular region and the selected region is mapped to the full window. Level of detail (LOD) is an advanced visualization technique that changes the resolution of a 2D/3D model depending on the size of the model on the screen. Currently all the grid points are displayed no matter how many grid points are in the viewing frustum. Using LOD, the number of grid points displayed should be reduced when there are too many grid points in the current viewing volume. An adaptive algorithm can be developed to automatically determine the number of grid points to be displayed based on the number of grid points in the viewing frustum.

While the project has not been fully utilized in the courses yet, preliminary trial use by several students provided positive feedback. A detailed usability and applicability study will be conducted in Spring 2024 when ECE 320 is offered.

6 CONCLUSION

This paper presented a preliminary design of a flow visualization project that facilitates student learning of numerical integration in a continuous simulation course and explores various visualization and user interaction techniques in a computer graphics and visualization course. The northeast Pacific Ocean data were utilized to enable students to have an appreciation of the nature and the complexity of real-world problems in terms of data sources and the process of solving real-world problems from scratch. Advanced software engineering techniques were employed to present a clean and expandable software architecture that expedites student learning and software development. Future work includes expanding the software architecture for visualization of problems in other domains, e.g., electrical engineering.

REFERENCES

Almulla, M. A. 2020. The Effectiveness of the Project-Based Learning (PBL) Approach as a Way to Engage Students in Learning. *SAGE Open*. https://doi.org/https://doi.org/10.1177/2158244020938702

Brezinski, C. and L. Wuytack. 2001. *Numerical Analysis: Historical Development in the 20th Century* (C. Berzinski & L. Wuytack, Eds.). Elsevier.

Burden, R. L., J. D. Faires, and A. M. Burden. 2015. *Numerical Analysis* (10th ed.). Cengage Learning.

Guo, P., N. Saab, L. S. Post, and W. Admiraal. 2020. A review of project-based learning in higher education: Student outcomes and measures. *International Journal of Educational Research, 102*. https://doi.org/https://doi.org/10.1016/j.ijer.2020.101586

Hansen, C. D., and C. Johnson. 2004. *Visualization Handbook*. Academic Press.

Klee, H. and R. Allen. 2011. *Simulation of Dynamic Systems with MATLAB and Simulink*.

Kline, M. 2019. The Looming Higher Ed Enrollment Cliff. *Higher Ed HR Magazine* (Fall).

Leathrum, J. F., Y. Shen, and O. Gonzalez. 2021. A New M&S Engineering Program with a Base in Computer Engineering. 2021 Winter Simulation Conference.

Liu, Z., R. Moorhead, and J. Groner. 2006. An Advanced Evenly-Spaced Streamline Placement Algorithm. *IEEE Transactions on Visualization and Computer Graphics, 12*(5), 965-972.

MathWorks. 2022. *What is Numerical Analysis*. https://www.mathworks.com/discovery/numerical-analysis.html#:~:text=Numerical%20analysis%20is%20a%20branch,or%20prohibitively%20expensive%20to%20calculate.

National Research Council. 2000. *How People Learn*. The National Academies Press.

National Research Council. 2003. *Improving Undergraduate Instruction in Science, Technology, Engineering and Mathematics*. T. N. A. Press.

National Research Council. 2012a. *Education for Life and Work: Developing Transferable Knowledge and Skills in the 21st Century (2012)*. The National Academies Press.

National Research Council. 2012b. *Improving Adult Literacy Instruction: Options for Practice and Research*. The National Academies Press.

National Research Council. 2012c. *Report of a Workshop on Science, Technology, Engineering, and Mathematics (STEM) Workforce Needs for the U.S. Department of Defense and the U.S. Defense Industrial Base*. The National Academies Press.

National Research Council. 2018. *How People Learn II: Learners, Contexts, and Cultures*. The National Academies Press.

Project Kaleidoscope. 2002. *Recommendations for Action in Support of Undergraduate Science, Technology, Engineering, and Mathematics*.

Ugeng, S.-K., C. Sikorski, and K. L. Ma. 1996. Efficient Streamline, Streamribbon, and Streamtube Constructions on Unstructured Grids. *IEEE Transactions on Visualization and Computer Graphics*, *2*(2), 100-109.

van der Pol, B. 1926. LXXXVIII. On "relaxation-oscillations". *The London,, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 978-992. https://doi.org/https://doi.org/10.1080/14786442608564127

Wagner, K. and S. Kingston 2022. School Leaders Play an Essential Role in Making High Quality PBL Happen for Students. *PBL Evidence Matters*, *2*(2), 1-5.

Wang, M., J. Tao, C. Wang, C. K. Shene, and S. H. Kim. 2013. *FlowVisual: Design and Evaluation of a Visualization Tool for Teaching 2D Flow Field Concepts* 120th ASEE Annual Confernece & Exposition, Atlanta, GA.

Wang, M., J. Tao, C. Wang, C. K., Shene, and S. H. Kim 2022. *FlowVisual: Introduction to Flow Visualization Concepts*. https://www3.nd.edu/~cwang11/2dflowvis.html

White, F. M. 2011. *Fluid Mechanics*. McGraw-Hill.

Zill, D. 2000. *A First Course in Differential Equations: The Classic Fifth Edition*. Cengage Learning.

AUTHOR BIOGRAPHIES

YUZHONG SHEN is a Professor of Electrical and Computer Engineering at Old Dominion University. He holds a PhD in Electrical Engineering from the University of Delaware. His research interests lie in visualization and computer graphics, virtual reality, augmented reality, transportation modeling and simulation, signal and image processing, and general modeling and simulation. His email address is yshen@odu.edu.

MASHA SOSONKINA is a Professor 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, parallel numerical algorithms, energy-efficient computing systems, fault-tolerant algorithms and applications, computational science and engineering, and data-science infrastructure. Her email address is msosonki@odu.edu.