www.ijonest.net

Implementation of the High-Performance Computing Summer Institute at Jackson State University

Shuang Z. Tu
Jackson State University, USA

Chao Jiang
Jackson State University, USA

To cite this article:

Tu, S.Z. & Jiang, C. (2024). Implementation of the high-performance computing summer institute at Jackson State University *International Journal on Engineering, Science, and Technology (IJonEST), 6*(2), 112-131. https://doi.org/10.46328/ijonest.200

International Journal on Engineering, Science, and Technology
2024, Vol. 6, No. 2, 112-131 https://doi.org/10.46328/ijonest.200

Implementation of the High-Performance Computing Summer Institute at Jackson State University

Shuang Z. Tu, Chao Jiang

Article Info

Article History

Received:
20 August 2023
Accepted:
17 November 2023

Keywords

Engineering education
High performance computing
Summer institute
ChatGPT

Abstract

Between May 25, 2023 and June 21, 2023, we hosted the inaugural four-week High-Performance Computing Summer Institute at Jackson State University. This endeavor was made possible through the support of a three-year NSF CISE-MSI grant. The primary objective of this Summer Institute revolved around the engagement, education, and empowerment of minority and underrepresented students in the realm of High-Performance Computing (HPC) within the field of engineering. Nine undergraduate students with diverse background were recruited to participate in this program. Throughout the program, we immersed these students in a comprehensive curriculum that covered various critical facets of HPC. This curriculum encompassed hands-on instruction in Linux operating system command-line operations, C programming within the Linux environment, fundamental HPC concepts, parallel computing utilizing the Message Passing Interface (MPI) library, and GPU computing through OpenCL. Additionally, we delved into foundational aspects of fluid mechanics, geometric modeling, mesh generation, flow simulation via our in-house flow solvers, and the visualization of solutions. At the end of the program, every participant was tasked with delivering an oral presentation and submitting a written report encapsulating their acquired knowledge and experiences during the program. We are excited to share a detailed overview of our program's implementation with our audience. This includes insights into our utilization of ChatGPT to enhance C programming learning and our suggestion of the NSF ACCESS resources to gain access to HPC systems. We are proud to announce that the program has achieved remarkable success, as evidenced by the positive feedback we received from the participants.

Introduction

Our recently funded three-year NSF grant titled “Collaborative Research: CISE-MSI: DP: FET: Modernizing Numerical Flow Solvers with Heterogeneous Computing” is a joint project with the University of Mississippi. The educational goal of this project is to leverage our expertise in developing numerical flow solvers to expose and teach students various parallel computing and programming skills. We believe that parallel and distributed computing must be an essential part of engineering and computer science education. In this project, we proposed several educational components at both Jackson State University (JSU) and the University of Mississippi. The educational component at JSU is to host an annual 4-week High Performance Computing Summer Institute (HPCSI). The goal of the summer program is to attract, educate and nurture minority and underrepresented students in the field of HPC.

High-performance Computing (HPC)

High-performance computing (HPC) has become an indispensable tool in the field of science and engineering for a multitude of reasons, addressing the ever-growing complexity and scope of modern engineering challenges and handling vast amount of data. The need for quick and accurate simulations, data analysis, and modeling has grown exponentially. HPC systems enable engineers to tackle these challenges by providing the computational horsepower required to carry out compute-intensive simulations and process vast amounts of data. Whether designing advanced aircraft using aerodynamic simulations, optimizing structures using structural analysis, or studying the behavior of materials at the atomic level, HPC empowers engineers to perform tasks that would be inconceivable with traditional computing resources. HPC allows engineers to quickly explore numerous design possibilities, test hypotheses, and solve complex differential equations governing physical problems. Besides, in the age of Big Data and sophisticated computational algorithms, high-performance computing has become a fundamental enabler, driving innovation and efficiency. With the hope to simulate real-world conditions, high-performance computing is an essential catalyst for innovation, efficiency, and progress within the field of science and engineering.

Parallel Computing

HPC and parallel computing are closely intertwined. While HPC refers to the use of powerful computing systems to address computationally intensive problems efficiently, parallel computing is a fundamental technique used to achieve high-performance computing. Parallel computing is a method where multiple processors or processor cores work together simultaneously to solve a problem. With parallel computing, a computing task is partitioned into sub-tasks, i.e. dividing the workload among multiple processing units, such as central processing units (CPUs) and/or graphics processing units (GPUs). All sub-tasks are executed simultaneously. This division of workload among processing units speeds up computations, making it possible to conduct complex simulations and handle large datasets that would be impractical or impossible with sequential processing.

Parallel computing techniques can be classified into various categories, such as task parallelism and data parallelism, each tailored to different computational problems. These techniques are implemented through parallel programming, where algorithms and software are designed to exploit the capabilities of multiple processors. For example, parallel programing paradigms OpenMP and Pthreads are suitable for shared-memory systems while the Message Passing Interface (MPI) is suitable for both the shared-memory systems and distributed-memory systems. Parallel programming implements parallel computing, enabling HPC systems to tackle complex, compute-intensive or data-intensive problems efficiently and harness the full potential of modern supercomputing resources.

In the past, parallel computing only utilizes multiple CPUs or CPU cores to speed up engineering simulations. Today's HPC systems are often also equipped with other processing units including general purpose GPUs, field-programmable gate arrays (FPGAs) and other specialized accelerators, in addition to multiple CPUs or CPU cores. Such diverse processing units introduce a concept of heterogeneous computing. With heterogeneous computing, all computing units work together to perform and speed up computations. Heterogeneous computing requires specialized software and programming frameworks to effectively manage and distribute tasks across different processing units. Open standards like OpenCL, and CUDA have been developed to facilitate this process, making it easier for developers to harness the power of heterogeneous architectures.

Lack of Education in HPC

The lack of education in HPC is a notable issue in many academic institutions including Jackson State University (JSU). Many individuals are unaware of the significance and applications of HPC in various domains. This lack of awareness can prevent them from seeking or offering HPC education. As a result, traditional computer science and engineering curricula often do not emphasize HPC education. This omission leaves students unprepared for real-world applications that demand HPC skills. Developing and maintaining an HPC curriculum requires resources and funding, which may not be available at all institutions. In addition, many educational institutions lack the necessary parallel computing infrastructure for teaching HPC effectively. Access to specialized hardware and software can be limited, which hinders hands-on learning.

Addressing the lack of education in HPC is vital, as this field is integral to advancing scientific research, engineering, data analytics, and simulations across numerous disciplines. Increasing investments in HPC education, developing accessible curriculum resources, and fostering collaborations between educational institutions and HPC centers are essential steps to bridge this educational gap and prepare students for the challenges and opportunities in high-performance computing.

In this paper, we aim to present how we implement our NSF-sponsored 4-week HPC Summer Institute at JSU. The paper is organized as follows. We first present the participant recruitment process and the development of agenda plans centered around essential HPC topics we deemed necessary. Subsequently, we delve into an in-depth description of each of these topics. Lastly, concluding remarks, discussions and recommendations intended to benefit forthcoming summer programs with similar goals are given.

HPC Summer Institute Recruitment and Agenda

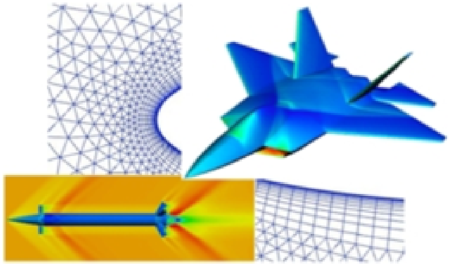
Recruitment

The HPC Summer Institute was planned to be held during the period between May 25, 2023 and June 21, 2023, which coincides with the first summer session of Jackson State University. The recruitment flyer (see Figure 1) was distributed campus wide in early January right after the 2023 spring semester started. As can be seen, we required applicants submit a statement of interest about their understanding of HPC to ensure applicants have some basic concepts of HPC before they attend the Summer Institute.

By the end of March 2023, we accepted nine applications. Among the selected applicants, five are majoring in computer science majors, three in engineering, and one in accounting. Three of the applicants are female, three are international students, and five are of African American heritage. The GPA ranged from 3.2 to 4.0.

JSU 1877 | JACKSON STATE UNIVERSITY®

High Performance Computing Summer Institute @ Jackson State University



The NSF sponsored 2023 High Performance Computing Summer Institute at Jackson State University will be held during weekdays between May 25, 2023 and June 21, 2023 at 9am – 4pm. Selected participants will receive $2160 stipend plus lunch. Participants will be responsible for their transportation and lodging.

This application package contains:

- *Application Form* (this page)
- *Personal Statement* (Applicants are expected to do some research and describe their understanding of High Performance Computing (HPC), explain why they want to attend this program and possibly other information you want to tell us. Limit your statement within one page please)
- *Latest College Transcript* (Embed your university transcript to the end of this file)

The application package must be combined into a single PDF file. Please send your completed package to Dr. Shuang Tu at shuang.z.tu@jsums.edu by March 1, 2023. Top ten applicants will be selected to attend this program. The acceptance notice will be sent out by Dr. Tu by March 15, 2023.

Application Form

Name: , Gender: ☐ Female ☐ Male
J#: Email: Phone:
Major: , Grade: ☐ Junior ☐ Senior, GPA:
Emergency Contact Name: Emergency Contact Phone:
Will you be committed to the program once accepted? Yes

Contact Information: Dr. Shuang Tu | shuang.z.tu@jsums.edu | (601) 979-1275

Figure 1. Advertising flyer for the High-Performance Computing Summer Institute

HPC Topics

High-performance computing in engineering is inherently multidisciplinary. It involves the knowledge preparation in the following topics:

1. Physics
   (a) Classic mechanics (e.g., Newton's laws of motion)
   (b) Laws of thermodynamics (e.g., first law of thermodynamics)
   (c) Fluid mechanics (e.g., aerodynamics, hydrodynamics) and structural mechanics
   (d) Combustion, chemical reactions, electromagnetics, etc.
2. Mathematics
   (a) Calculus and differential equations
   (b) Linear algebra
3. Numerics - various numerical methods
   (a) Iterative methods, numerical integration, linear system solver
   (b) Finite difference/volume/element methods etc.
4. Computer usage
   (a) Unix/Linux environment
   (b) Programming languages (C/C++, Fortran etc.)
5. Pre- and post-processing of data (e.g., mesh generation, solution visualization, line plots, etc.)

Summer Institute Agenda

Based on the topics listed in the previous sub-section, the agenda of the four-week Summer Institute was planned as shown in Table 1. The Summer Institute took place in a computer lab that featured computers running the Fedora Linux operating system. Instructors conducted morning sessions by delivering lectures, while participants were given the opportunity to engage in practical, hands-on exercises on the assignments during the afternoon sessions. Our lecture slides were sent to participants electronically after each morning session.

Table 1. HPC Summer Institute Agenda

Week	Day	Topic
Week 1	Day 1	Linux OS and Commands
	Day 2	Linux OS and Commands
	Day 3	C Programming on Linux
	Day 4	C Programming on Linux
	Day 5	C Programming on Linux
Week 2	Day 1	C Programming on Linux
	Day 2	Computational Methods
	Day 3	Computational Methods
	Day 4	Computational Methods
	Day 5	Computational Methods
Week 3	Day 1	MPI Programming
	Day 2	MPI Programming
	Day 3	MPI Programming
	Day 4	GPU Computing

Week	Day	Topic
	Day 5	GPU Computing
Week 4	Day 1	Practice of Computational Fluid Dynamics
	Day 2	Mesh Generation
	Day 3	Flow Simulation
	Day 4	Solution Visualization
	Day 5	Participant Oral Presentations

HPC Summer Institute Topics in More Detail

In this section, the topics listed in Table 1 are elaborated in more detail.

Linux Operating Systems and Commands

Linux has become the dominant operating system for high-performance computing for several reasons such as its excellent scalability, stability and reliability, security, and cost-effectiveness. Linux operating system is powering most of the world's most powerful supercomputers and scientific computing clusters. For example, all the supercomputers in the top500 list (top500.org) are based on the Linux kernel.

Basic Linux Commands for General Users

Linux provides an extensive set of commands that allows users to efficiently navigate and interact with the Linux system through the command-line terminal. Most of our Summer Institute participants are newcomers to the Linux environment. They felt intimidated to interact with the OS through the command-line interface. Nevertheless, the Linux commands listed in

Table 2 are the most frequently used by ordinary users and suffice for everyday tasks. In addition, understanding the concepts of absolute path and relative path helps to navigate through directories more efficiently.

Table 2. Basic Linux commands

Command category	Commands	Usage
File/Directory	pwd	Display the current working directory's path
	ls	List files and directories in the current location
	cd	Change the current working directory
	mkdir	Create a new directory
	cp	Copy files or directories
	mv	Move or rename files and directories
	rm	Remove files or directories
Text editor	nano or vi	Text editors for creating and modifying text files
Text viewer	cat	Display the contents of a short text file

Command category	Commands	Usage
	less	View a text file page by page
Search	grep	Search for specific text patterns in files
	find	Locate files and directories on the system
Remote Operation	ssh	Securely connect to remote machines
	scp	Securely copy files between local and remote systems

We demonstrated the use of every command and explained some commonly used options for each command. We also emphasized the importance of the good practice of structuring contents using directories and sub-directories. Significant amount of time was spent on demonstrating the use of `vi` as a source code editor. We leverage our Red Hat Academy-designed Lab Environment to provide participants self-grading lab sessions on practicing basic Linux commands.

C Programming on Linux Environment

In the realm of scientific computing, high-level programming languages such as C, C++ and Fortran are commonly used to implement numerical algorithms by writing and compiling computer codes. These languages are favored because of their high performance, extensive support for mathematical libraries (e.g., BLAS, LAPACK), and their ability to take full advantage of parallel computing and vectorization, making them ideal for scientific computing to solve large-scale complex problems.

Most of our participants have taken one or two programming courses. The languages they learned are most likely Java and Python. Therefore, we think it is necessary to teach participants the C programming. Participants typically have experience with editing and compiling source code within a graphical Integrated Development Environment (IDE). However, programming in a Linux environment often favors the command-line approach. Therefore, we initially demonstrated to participants how to edit C source code using the command-line `vi` editor and subsequently guided them through the process of compiling C programs using the `gcc` compiler.

We did not create our own C tutorial since there are an abundant existing online resources. In this Summer Institute, we chose the C tutorial offered at https://www.geeksforgeeks.org/c-programming-language/. Figure 2 lists the C programming topics offered at the website. Each topic is clickable leading to more detailed sub-topics. In our lecture sessions, we explained each topic with the examples provided by ChatGPT. Since ChatGPT is a conversational tool, it can easily create a relevant example according to the prompt context. It turned out ChatGPT is an excellent teaching assistant and our participants loved the way of learning C programing with the help from ChatGPT.

The website also provides extensive programming examples at https://www.geeksforgeeks.org/c-programming-examples/. We selected some examples for the participants as hands-on practices in the afternoon sessions. We asked the participants to edit the source code in `vi` and compile the source code using `gcc` in the command-line terminal.

- C Basics
- C Variables and Constants
- C Data Types
- C Input and Output
- C Operators
- C Control Statements Decision-Making
- C Functions
- C Arrays & Strings
- C Pointers
- C User-Defined Data Types
- C Storage Classes
- C Memory Management
- C Preprocessor
- C File Handling
- Miscellaneous
- C Interview Questions
- C Programming Examples

Figure 2. C Tutorial Offered at https://www.geeksforgeeks.org/c-programming-language/

Computational Methods

Several key ingredients of computational methods were briefly introduced to the participants. Each ingredient is summarized as follows. A computer project was assigned for each topic to implement the algorithm explained in the lecture sessions.

Time Integration Method

Time integration methods, also known as time-stepping methods, an indispensable tool in computational science and engineering. These methods are designed to advance the solution of time-dependent problems, typically governed by differential equations with temporal derivative terms, in discrete time steps. Time integration enables the modeling of systems that evolve over time, such as fluid flows, structural dynamics, chemical reactions, and more. These methods come in various forms, including explicit and implicit schemes, multistep methods, and adaptive time-stepping algorithms. Stability and accuracy are the two factors when choosing a time integration method. In this Summer Institute, the explicit methods of the Runge-Kutta family and implicit methods based on Backward Difference Formula (BDF) were introduced to the participants.

Spatial Discretization for Partial Difference Equations (PDEs)

Due to the lack of analytical solution to general partial differential equations (PDEs) governing physical phenomena such as heat transfer, fluid flow, electromagnetics, and structural mechanics, various computational methods are widely employed in mathematics and engineering to approximate the solutions of differential equations by discretizing the domain into a grid/mesh of finite points/cells/elements. As a result of such discretization, the originally continuous differential equations are transformed into a set of algebraic equations that can be solved computationally. The resulting system of algebraic equations is typically solved using linear algebra techniques. Methods like the direct solver, iterative solvers, and preconditioning are applied to solve these systems efficiently. Among the computational methods, the finite difference method (FDM), the finite volume method (FVM), the finite element method (FEM), and their variants, are the most commonly used methods in numerically solving PDEs.

The finite difference method (FDM) operates on the principle of approximating derivatives in the differential equation with finite difference approximations. To allow the finite difference approximation, the computational grids must be structured (e.g., grids with clear row or column grid lines). Therefore, FDM is not well-suited for problems with highly irregular or complex geometries.

The finite volume method (FVM) converts the integral form of the PDEs into a discrete form by considering the conservation of mass, momentum, and energy over control volumes within a computational mesh. It provides a natural way to account for the physical conservation principles, making it particularly suited for problems involving fluid flow and heat transfer. The control volumes in the computational mesh can be of arbitrary geometric shape. Therefore, the finite volume method has the advantage of being able to handle complex geometries and irregular grids, making it a powerful tool for simulating a variety of computational engineering applications.

The finite element method (FEM) is also able to handle complex geometries by discretizing the computational domain into interconnected finite elements, e.g., triangles or quadrilaterals in 2D or tetrahedra or hexahedra in 3D. The solution over each element is approximated using the shape functions. The Galerkin orthogonality principle is used to construct the weak form of the integral equations that can be evaluated by numerical quadrature rules.

Linear Equation System Solver

As a result of temporal and spatial discretization, the originally continuous partial differential equations are transformed into a set of algebraic equations that must be solved using linear algebra techniques. Linear equation solvers employ various methods, such as direct methods like the LU decomposition, or iterative techniques like the Jacobi method. The linear equation system resulting from the computational discretization in realistic computational science and engineering applications is typically large, sparse and ill-conditioned. To efficiently and accurately solve such systems, one must resort to practical iterative methods including the Generalized Minimal Residual (GMRES) method for non-symmetric systems and conjugate gradient (CG) method for symmetric systems. In both methods, an appropriate preconditioning technique must be employed to improve the conditioning of the system to enhance the convergence and accuracy.

Newton-Raphson Iterative Method

Many computational engineering problems involve the solution of nonlinear equations or systems of nonlinear equations in the form of $f(x) = 0$. Due to its superior quadratic convergence performance, the Newton-Rahphson method is the most efficient and widely adopted iterative method in finding the solution of nonlinear equations. The Newton-Raphson method can be derived from the Taylor's series expansion by keeping the first two terms. Given an initial guess, the method iteratively refines the solution by using the function's value and derivative at that point. By successively linearizing the function around the current estimate, the iteration converges rapidly to a more accurate solution (cf. Figure 3).

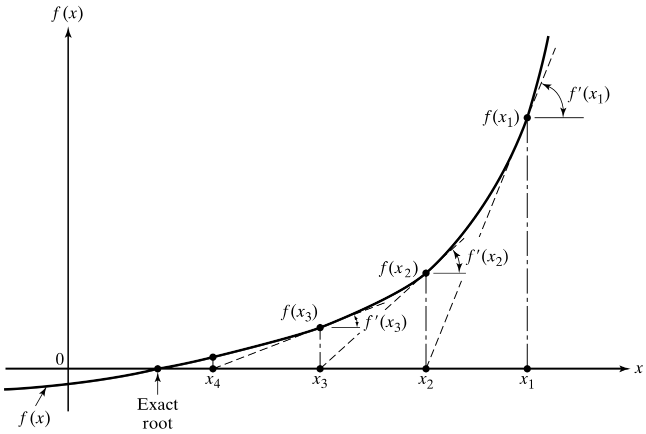


Figure 3. Graphical İllustration of the Newton-Raphson Method (diagram from pp. 65 of (Rao, 2002))

Numerical Integration

Numerical integration, also known as numerical quadrature, is another fundamental technique in computational mathematics used to approximate the definite integral of a function over a specified interval, i.e. $\int_a^b f(x)dx$. It provides a means to compute the area under a curve when an analytical solution is not readily available. Numerical integration methods divide the integration interval into discrete sub-intervals. The total area under the curve is obtained by summing the sub-areas within those sub-intervals. Traditional numerical integration methods, such as the mid-point rule, trapezoidal rule (cf. Figure 4), and Simpson's rules, approximate the integrand function over each sub-interval by a polynomial of some degree that can be easily integrated.

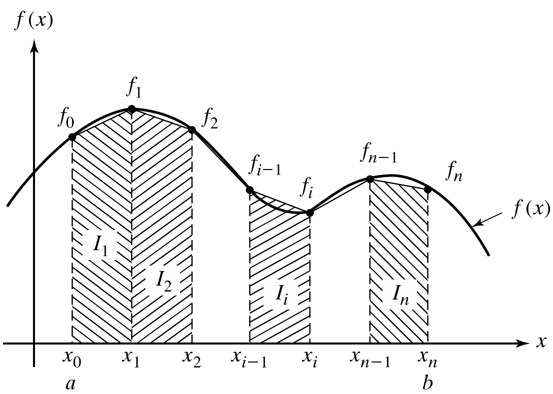


Figure 4. Illustration of Trapozoidal Rule for Numerical İntegration (Diagram from pp. 569 of (Rao, 2002))

By contrast, Gauss quadrature leverages orthogonal polynomials (often Legendre polynomials for the standard Gauss quadrature) to determine the positions of quadrature points and their corresponding weights. By utilizing these orthogonal polynomials, it ensures that the points are carefully placed in a way that minimizes the error, resulting in a very accurate approximation for a relatively small number of quadrature points. For n quadrature points, Gauss quadrature can exactly integrate a polynomial integrand up to degree $2n - 1$. Gauss quadrature is implemented by transforming physical coordinates to standard reference coordinates (cf. Eq. (1)). Therefore, quadrature points can be stored tabulated in reference coordinates. Due to its high efficiency and accuracy, Gauss quadrature rule is one of the most commonly used numerical integration techniques in computational engineering.

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} f\big(x(\xi)\big)d\xi \approx \frac{b-a}{2}\sum_{i=1}^{n} w_i F(\xi_i) \quad (1)$$

Introduction of Parallel Computing

MPI Programming

Message Passing Interface (MPI) library is one of the most commonly used parallel programing paradigms to access multiple processors simultaneously and speed up computation. The choice of the MPI parallel paradigm is due to its standardization, excellent platform independent portability and flexibility on both distributed memory and shared memory machines. The parallelism is achieved via the Single Program Multiple Data (SPMD) principle. The computational mesh is first partitioned across certain number of processes using the ParMETIS library (Karypis) (cf. Figure 5). The partitioning ensures the number of elements is roughly the same on each of the processes for the load balancing purpose. In addition, ParMETIS also minimizes the inter-process communication overhead. The same numerical solver program is then executed on each of the processes on its portion of the mesh simultaneously.

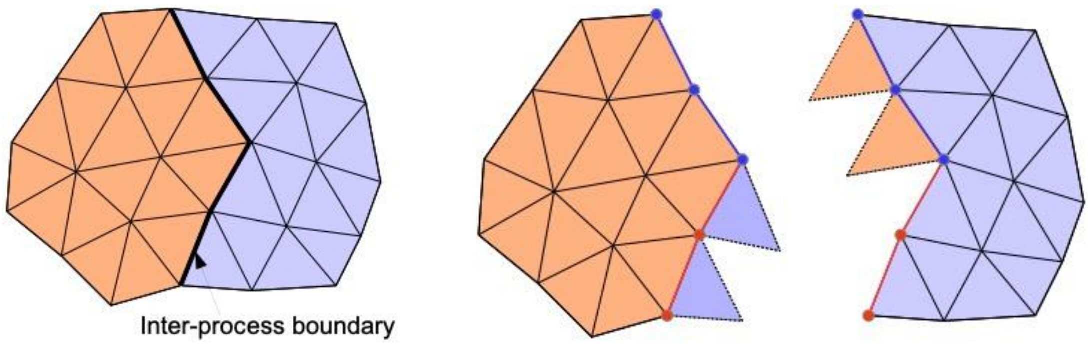


Figure 5. Mesh partitioning

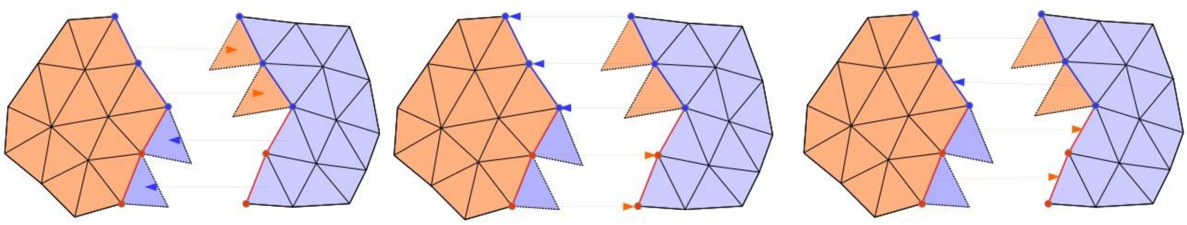


Figure 6. Inter-Process Communication. Left: Element Communication, Middle: Vertex Communication, and Right: Face Communication.

Inter-process communication occurs to synchronize the computation. Since the current cell-centered finite volume and nodal finite element solvers are constructed on compact computational stencils, the inter-processor communication involves only nodes, faces and elements on the partition boundaries (cf. Figure 6). This compactness makes it trivial to attain high parallelizability using MPI for fixed-topology meshes. Very efficient non-blocking MPI functions can be called to set up the inter-processor "gather" and "scatter" routines in the pre-processing stage (Johnson & Tezduyar, 1997; Tu et al., 2005). The communication overhead has been minimized thanks to these routines.

OpenCL Programming for GPU Computing

OpenCL is an industry standard, cross-platform programming frameworks for scalable heterogeneous computing platforms consisting of CPUs, GPUs and other co-processors. It includes a C-like language for programming kernels (i.e., codes that run on target OpenCL devices such as GPU) and APIs needed to define and control the platforms. OpenCL provides a parallel computing environment that targets data-parallelism and can greatly improve the performance of a wide spectrum of applications in numerous disciplines. A number of scientific and engineering applications have been successfully accelerated using OpenCL and CUDA (Both have a very similar programming model, and we choose OpenCL for better portability across different vendors). Compute and data intensive portions of solver are identified and programmed as kernels and offloaded to GPU. Thread mapping, memory access pattern, and different memory spaces must be carefully investigated and optimized in order to get the most out of powerful GPUs.

The OpenCL lectures were delivered by our invited speaker, Dr. Byunghyun Jang of the University of Mississippi, who is also our collaborator of this project.

Computational Fluid Dynamics (CFD) Basics

CFD refers to Computational Fluid Dynamics. CFD is an interdisciplinary branch of applied mathematics and engineering (see Figure 7) that involves the use of numerical methods and algorithms to simulate and study the behavior of fluids, such as gases and liquids, in a wide range of complex scenarios.

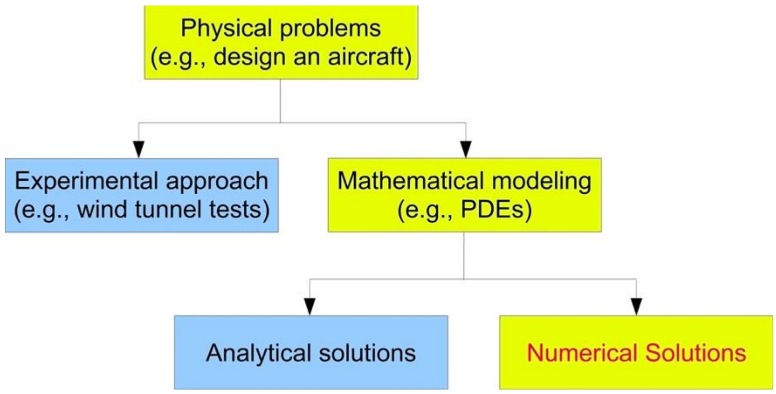


Figure 7. Numerical Solution of Physical Problems

CFD plays a vital role in various fields, including aerospace, automotive engineering, environmental science, and industrial processes. By discretizing the governing equations of fluid dynamics into discrete elements and solving them iteratively on a computer, CFD allows engineers and scientists to investigate fluid flow patterns, turbulence, heat transfer, and a multitude of related variables. With reliable CFD tools, expensive physical experiments (e.g., wind tunnel experiments) can be supplemented and even replaced by numerical experiments in the design process.

A typical CFD simulation involves the following tasks:

- Mesh generation based on the given geometry.
- Set up the solver inputs and run the simulation.
- Solution visualization and analysis.

We used a concrete example explained in next sub-section to explain how to conduct these tasks.

Flow Simulation and Visualization

As the final stage in the Summer Institute, we provided the participants a hands-on project on how to conduct CFD simulations using one of our in-house CFD flow solvers. Because of the time constraint, we asked participants to solve a 2-D problem as illustrated in Figure 8. Participants can solve this problem using multiple CPU cores on their local workstation to obtain the solution quickly.

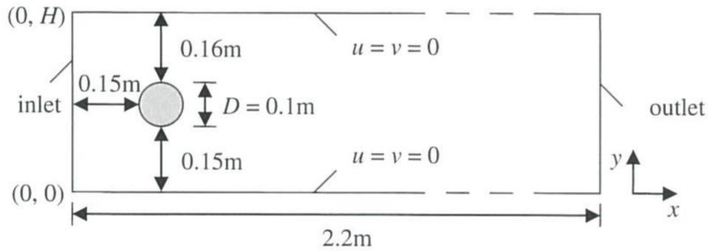


Figure 8. Geometry of the CFD Example.

This case is about a laminar incompressible flow around a circular cylinder placed in a channel. Abundant numerical results obtained via various numerical solvers are tabulated (Schafer, 1996). Therefore, this problem has been widely accepted as a standard test case to verify and validate an incompressible solver. Since no-slip boundary conditions are applied at the top and the bottom walls of the channel, a special parabolic inflow velocity profile must be given to account for the zero velocity at the inlet tips of both walls. The velocity profile is given as follows

$$u(0,y,t) = 4u_m y(H - y)/H^2, \quad v = 0$$

with $u_m = 1.5m/s$ and $H = 0.41m$ as the height of the channel. The mean velocity at inflow is $\overline{U} = \frac{2u_m}{3} = 1.0m$. The Reynolds number based on the mean velocity and the diameter of the cylinder is 100. At this Reynolds number, the flow behind the cylinder is expected to become non-stationary and periodic Karman vortex shedding should be seen.

From this example, participants can gain insight of several fluid dynamics concepts such as boundary layers, Reynolds number, vortex shedding etc..

Mesh Generation

We selected Gmsh (Geuzaine & Remacle) to generate an unstructured quadrilateral computational mesh for the geometry shown in Figure 8. Gmsh is an open-source finite element mesh generation and post-processing software tool designed for numerical simulations in various scientific and engineering disciplines. It provides a user-friendly and versatile environment for creating complex 3D and 2D meshes, which are essential for solving partial differential equations in fields such as computational fluid dynamics, structural analysis, electromagnetics, and more. Gmsh is known for its capability to generate high-quality meshes for a wide range of geometries and its flexibility in handling both structured and unstructured grids. With a graphical user interface and scripting capabilities, it offers ease of use for beginners and extensive customization options for advanced users.

We demonstrated the process of mesh generation using Gmsh in detail. Figure 9 shows the mesh generated by one of the participants. In this mesh, a square domain is created to enclose the inner cylinder to ensure the high mesh quality near the cylinder. The grid lines are clustered near the wall boundaries including the cylinder wall, and the top and bottom walls. The mesh file generated by Gmsh needs to be transformed into the format recognizable by our in-house flow solver.

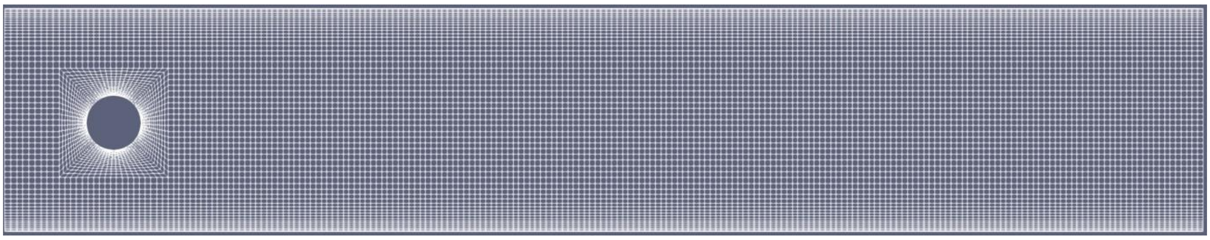


Figure 9. An Unstructured Quadrilateral Mesh Generated by One of the Participants

Flow Simulation

The CFD solver used in this simulation is one of our in-house flow solvers. This incompressible flow solver was developed based on a hybrid cell-centered finite volume method and vertex finite element method (Tu & Aliabadi, 2007). The solver can solve both 2-D and 3-D problems with computational domain discretized by hybrid elements. In addition, since the solver has been parallelized using the Message Passing Interface (MPI) library, the solver is able to solve large-scale problems on parallel computing systems.

Before the simulation can be started, an input file must be created. The input file contains simulation parameters such as the boundary conditions, linear solver (i.e. GMRES and CG) parameters, and other simulation control parameters. The input file, together with the mesh files, are provided to the flow solver executable for the simulation.

Solution Visualization

During the simulation, the solution is visualized using ParaView. ParaView (Kitware) is an open-source, cross-platform data visualization and analysis tool widely used in scientific computing, engineering, and data analysis. It is known for its ability to handle large datasets, conduct sophisticated visualizations, and interact with distributed computing resources using MPI parallel data processing. ParaView allows remote visualization without the need of transferring large data sets from remote supercomputing systems to local workstations.

Since the flow remains unsteady, the simulation is stopped when the oscillating flow pattern is clearly seen in ParaView. Participants are asked to use ParaView to create several images to show the simulated flow field. Figure 10 and Figure 11 are two such images corresponding to an instantaneous moment. Figure 10 shows the velocity magnitude field and Figure 11 shows the pressure contours in the domain. The vortex shedding nature can be clearly seen in the flow field. Participants are also asked to use Gnuplot to plot the drag and lift coefficients as a function of time. Figure 12 shows such plots.

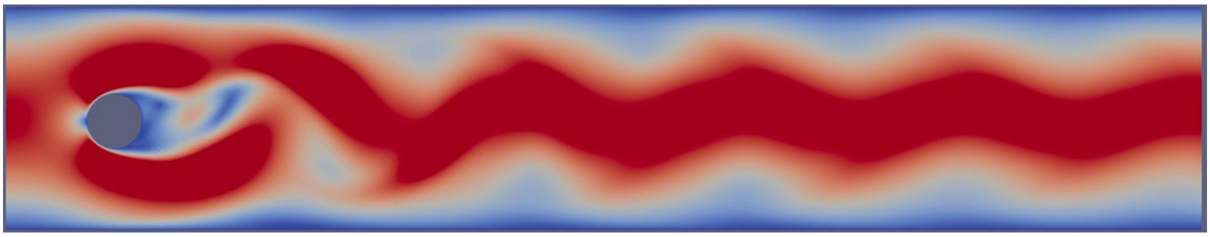


Figure 10. Simulation Result: Velocity Magnitude Field

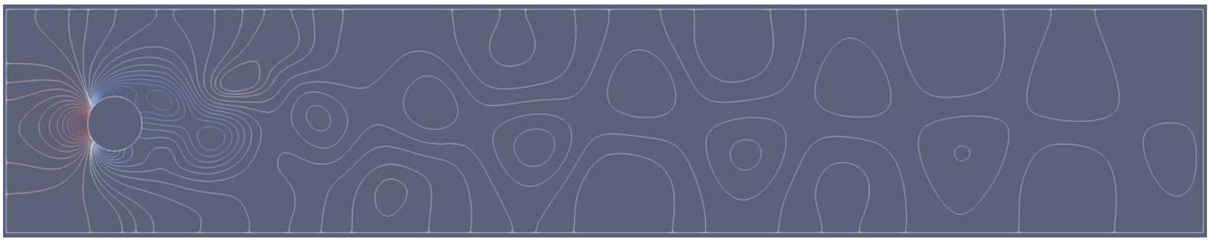


Figure 11. Simulation Result: Pressure Contours

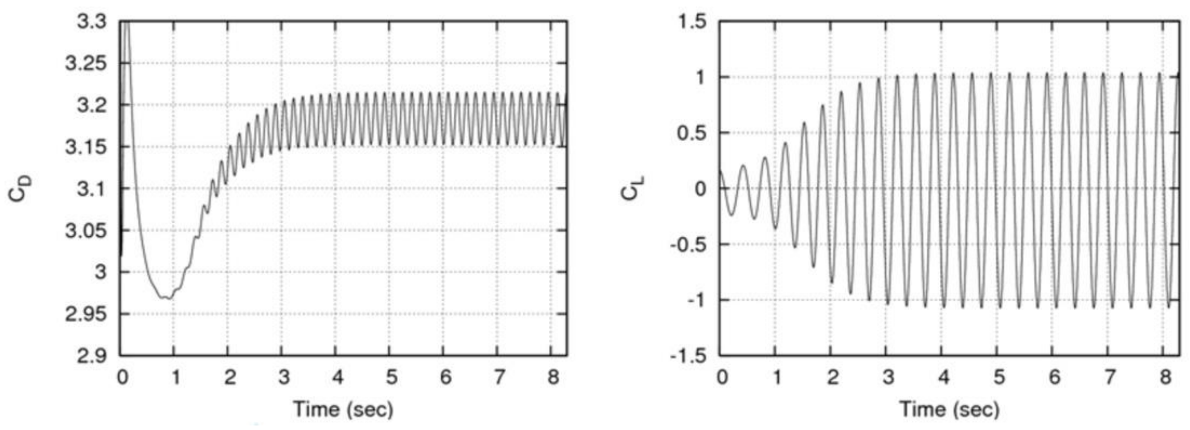


Figure 12. Simulation Result: Drag and Lift Coefficients

Conclusion, Discussion and Recommendations

In this paper, we reported how we executed our inaugural four-week High Performance Summer Institute sponsored by a National Science Foundation grant. The program welcomed a diverse group of nine Jackson State University undergraduate students, representing varied genders, majors, and nationalities. This summer initiative was an intensive and immersive experience, offering participants a comprehensive exposure to a wide spectrum of high-performance computing topics and extensive hands-on practice on each topic.

Towards the program's conclusion, participants were afforded the opportunity to engage in an HPC simulation of a computational fluid dynamics problem. This practical approach can be adapted to other computational engineering domains, including computational structural mechanics and computational electromagnetics.

High-Performance Computing (HPC) is inherently a multidisciplinary field, and acquiring proficiency in HPC skills can be a formidable challenge with a steep learning curve. We did not anticipate that this four-week summer program would transform the participants into HPC experts. Our primary goal was to enlighten and inspire the minority and underrepresented participants about the world of HPC, igniting their interest in this domain and, in the long tun, contributing to the growth of a diverse and skilled workforce in the HPC field.

This summer program has been executed effectively as evidenced by the positive feedback received from the participants. Below are some selected excerpts:

> "*I am writing to express my heartfelt gratitude for the incredible learning experience I had under your guidance during the HPC summer camp. The knowledge and skills I acquired during the one-month class have been truly transformative, and I am immensely grateful for the opportunity to learn from you. Your expertise, passion, and dedication to teaching have been evident throughout the program. Your ability to explain complex concepts in a comprehensive and engaging manner has not only deepened my understanding of computer science but also broadened my perspective on its interconnections with other fields, particularly mathematics.*" - By a participant.

> "*The summer camp has opened my eyes to the fascinating relationship between computer science and mathematics, and it has ignited a strong desire within me to explore this connection further. I am now more inspired than ever to pursue my studies in mathematics and delve deeper into the intricacies of this interdisciplinary realm. Your guidance and instruction have been instrumental in this newfound passion, and I am truly grateful for the impact you have had on my academic journey.*" - By a participant.

Indeed, two of the participants became our undergraduate research assistant starting in the fall semester of 2023.

Nevertheless, in the implementation of this summer program, we have some suggestions to share with the readers of this paper.

Early Distribution of Recruitment Flyers

It turned out that the program recruiting flyer should have been sent out earlier in the 2022 fall semester. Many students have already secured their internship positions by the time they saw our flyer. While many expressed their interest, they have committed their 2023 internship positions.

Use of ChatGPT as a Teaching Assistant

In our summer program, we experimented with incorporating ChatGPT as a teaching assistant for instructing C programming. Figure 13 provides an illustration of our utilization of ChatGPT to generate relevant examples demonstrating the use of 'for' and 'while' loops. As depicted, ChatGPT operates as a conversational-style generative chat tool, generating responses based on contextual cues. Students welcomed this way of learning computer programming.

Hands-on Practice

In this intensive 4-week summer program, students could easily become disengaged and overwhelmed if instructors solely delivered lectures without providing ample opportunities for hands-on projects. Our approach of morning lectures followed by afternoon hands-on practices proved instrumental in sustaining students' enthusiasm for learning and actively engaging them in the learning process. Moreover, learning HPC involves not just theoretical knowledge but also practical expertise.

Use ACCESS Resources

ACCESS is an acronym that stands for "Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support."

> *ACCESS is a program established and funded by the National Science Foundation (NSF) to help researchers and educators utilize the nation's advanced computing systems and services. Almost any computer application that requires more than a desktop or laptop could qualify as needing an advanced computing system. Examples include supercomputer applications, AI and machine learning, big data analysis and storage, and others.*" The ACCESS helps *"researchers and educators, with or without supporting grants, to utilize the nation's advanced computing systems and services –* ***at no cost.*** (from ACCESS website).

We did not have a chance to utilize this valuable resource since we did not have sufficient time to complete the application process before the summer program started. But we will definitely utilize this resource in future summer programs.

Finally, we expect that the insights and experience acquired during this inaugural summer program will improve the overall efficiency and effectiveness of our future summer programs, ultimately providing greater benefits to the participating students. Additionally, we hope that the readers of this paper will find valuable information to enhance their own similar summer programs.

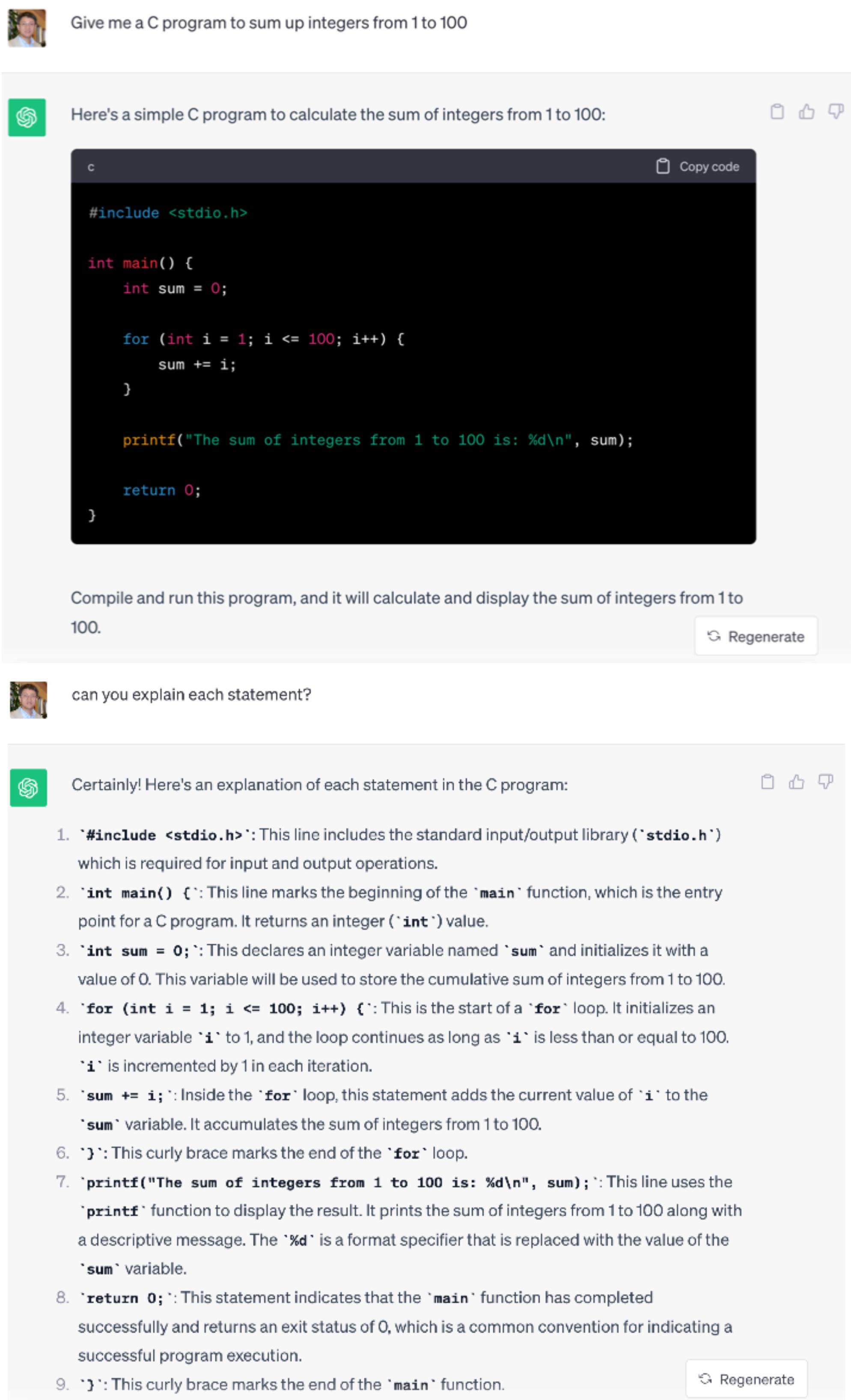


Figure 13. An Example of using ChatGPT as a Teaching Assistant

Acknowledgements

The 2023 High Performance Computing Summer Institute at Jackson State University is sponsored by U.S. National Science Foundation under the Award No. 2219542. We would also like to express our gratitude to Dr. Byunghyun Jang from the University of Mississippi for dedicating an entire day to deliver a lecture on OpenCL programming to our program participants.

References

ACCESS. https://access-ci.org/

ChatGPT. https://chat.openai.com.

Geuzaine, C. & Remacle, J.-F. A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities, http://gmsh.info.

Gnuplot. http://www.gnuplot.info/.

Johnson, A., & Tezduyar, T. (1997). Parallel computation of incompressible flows with complex geometries, *Int. J. Numer. Meth. Fluids* 24**,** 1321.

Karypis, G. (n.d.). *ParMETIS — parallel graph partitioning and fill-reducing matrix ordering*, https://github.com/KarypisLab/ParMETIS.

Kitware. ParaView, https://www.paraview.org.

Rao, S.S. (2002). *Applied Numerical Methods for Engineers and Scientists*, published by Prentice Hall, ISBN-10: 0-13-089480-X, ISBN-13: 978-0-13-089480-9.

Schafer, M, Turek, S. (1996). Benchmark computations of laminar flow around a cylinder, in: E.H. Hirschel, (Ed), *Flow Simulation with High-performance Computers II*, Notes on Numerical Fluid Mechanics, Vieweg, 547-566.

TOP500, https://www.top500.org/statistics/details/osfam/1/

Tu, S., Aliabadi, S., Johnson, A., & Watts, M. (2005). A robust parallel implicit finite volume solver for high-speed compressible flows, AIAA Paper 2005-1396.

Tu, S. & Aliabadi, S. (2007). Development of a Hybrid Finite Volume/Element Solver for Incompressible Flows on Unstructured Meshes, *International Journal of Numerical Methods in Fluids*, Vol. 55, No. 2, pp. 177-203.

Author Information

Shuang Z. Tu
https://orcid.org/0000-0002-4506-6447
Jackson State University
1400 J.R. Lynch St., Jackson, MS
USA
Contact e-mail: *shuang.z.tu@jsums.edu*

Chao Jiang
https://orcid.org/0009-0002-4871-0397
Jackson State University
1400 J.R. Lynch St., Jackson, MS
USA