Navigating the Floating-Point Seas: From Bitwise Reproducibility to Reduced Precision Computing

Bálint Siklósi

Theses of the PhD Dissertation

Supervisor:

Dr. István Zoltán Reguly, PhD

Pázmány Péter Catholic University

Roska Tamás Doctoral School of Sciences and Technology

Budapest, 2025

1 Introduction

Floating-point arithmetic is a cornerstone of modern scientific computing. From weather modeling to engineering simulations, it enables the digital approximation of complex, continuous real-world systems. However, floating-point operations are inherently limited by finite precision, rounding errors, and their non-associative nature. These limitations manifest as two key challenges in high-performance computing: lack of reproducibility and the tension between numerical accuracy and performance efficiency.

This dissertation addresses both problems through two interrelated themes. The first is bitwise reproducibility – the guarantee that computations produce bit-for-bit identical results across different runs, many hardware platforms, or parallelization levels. Reproducibility is critical in scientific contexts where conclusions must be robust, verifiable, and portable. Yet it is often undermined by parallel computation, where the order of floating-point operations is unpredictable. The second theme is reduced and mixed-precision computing, which offers the promise of higher performance and energy efficiency by selectively using lower-precision representations. However, such performance comes at the risk of introducing unacceptable numerical inaccuracies.

The aim of this research is to explore both challenges in the context of large-scale computational simulations. Specifically, the dissertation contributes generalizable solutions implemented in domain-specific frameworks – OP2 and OPS – that address reproducibility and precision trade-offs without compromising scalability or productivity. Throughout the research, methods are tested on benchmark problems and industrial-grade solvers, including the Rolls-Royce Hydra CFD application and the OpenSBLI library, providing a broad and practical evaluation context.

2 Methods and Tools

2.1 Floating-Point Arithmetic and Precision Challenges

Floating-point arithmetic underpins scientific computing by enabling the representation and manipulation of real numbers in digital form. A floating-point number is typically represented as

$$(-1)^{sign} \times significand \times base^{exponent},$$

where the *sign* bit indicates polarity, the *significand* (or mantissa) determines precision, the *base* is usually 2, and the *exponent* scales the magnitude. The IEEE 754 standard [1] defines common formats such as half-precision (16-bit), single-precision (32-bit), and double-precision (64-bit), balancing dynamic range and accuracy.

However, floating-point arithmetic introduces inherent limitations, notably rounding errors and non-associativity of operations [2, 3]. Because results must be rounded to fit the format, the order of operations affects the final outcome, which can cause variability in parallel computations where operation order is non-deterministic. In the Aero benchmark (OP2), running the same conjugate-gradient solver with different numbers of MPI processes produces small but measurable differences in results due to rounding and non-associativity [4].

This non-determinism complicates reproducibility, which is critical for debugging, verification, and scientific validation. Bitwise reproducibility-ensuring identical results across runs regardless of parallelism-requires controlling or eliminating sources of rounding variability, especially in reductions. Approaches like ReproBLAS [5] implement reproducible summations by carefully managing floating-point additions to guarantee bitwise identical results independent of execution order.

The increasing computational demands of scientific applications have motivated exploration of reduced and mixed-precision floating-point arithmetic to improve performance, memory usage, and energy efficiency.

Reduced-precision computing uses floating-point formats with fewer bits (e.g., 16-bit half precision or even 8-bit formats) to represent numbers, which can significantly decrease memory footprint and increase throughput on hardware optimized for low-precision operations. However, reducing precision inherently increases rounding errors and limits representable range, potentially affecting numerical accuracy and stability.

Mixed-precision arithmetic combines different floating-point precisions within a single computation to balance accuracy and efficiency. For example, intermediate calculations may be performed in lower precision, while critical accumulations or corrections use higher precision. Iterative algorithms, such as gradient descent or iterative solvers, are particularly amenable to mixed precision because initial coarse approximations can be refined progressively with higher precision steps.

This approach leverages hardware capabilities on modern HPC systems, including GPUs and AI accelerators, which often provide significantly higher throughput for low-precision operations. Mixed precision can yield large speedups and reduce energy consumption without sacrificing overall solution quality when carefully applied.

In this dissertation, the exploration of reduced and mixed-precision computing is motivated by these trade-offs: achieving computational efficiency gains while maintaining acceptable numerical accuracy and reproducibility.

2.2 OP2 Framework

OP2 is a domain-specific abstraction framework for parallel execution of unstructured mesh computations [6]. It enables developers to express computations over mesh sets and mappings in a high-level, platform-agnostic manner. The OP2 runtime then automatically generates optimized parallel code for various architectures, including multi-core CPUs and GPUs.

OP2 supports distributed memory parallelism via MPI and shared memory parallelism through OpenMP or CUDA, allowing scalable performance on heterogeneous HPC systems. It has been successfully applied to scientific applications such as fluid dynamics and finite element methods [4].

2.3 OPS and OpenSBLI

OPS (Oxford Parallel library for Structured mesh solvers) is a similar abstraction framework targeting structured mesh stencil computations [7]. It provides a high-level API and supports multiple parallel backends (MPI, OpenMP, CUDA), enabling portable and efficient execution on diverse HPC platforms.

OpenSBLI builds on OPS to automate generation of high-order finite difference solvers for compressible flow problems [8]. It translates symbolic problem specifications into optimized parallel code leveraging OPS, facilitating rapid development and experimentation with numerical methods.

Together, OP2, OPS, and OpenSBLI represent modern approaches to scientific computing that emphasize abstraction, automation, and performance portability, addressing both unstructured and structured mesh problems. Their high-level APIs allow scientists to focus on the numerical algorithms without modifying low-level code for different architectures. This separation of concerns enables automation in code generation and optimization, ensuring that applications remain maintainable and efficient across evolving HPC platforms.

3 New scientific results

This section presents the main scientific contributions of the dissertation in the form of thesis points. Each thesis group highlights a major area of research, while individual points describe specific results.

Thesisgroup I. – Algorithms for reproducible floating-point operations defined on unstructured mesh applications.

Thesis I.1. Starting from the OP2 DSL abstraction, I showed which floating-point operations – such as parallel reductions, indirect memory updates, and non-deterministic execution order – are responsible for the potential violation of the reproducibility property. I showed what steps – temporary array-based accumulation, deterministic graph coloring, and the use of ReproBLAS – can be taken to ensure that these operations still produce reproducible results.

In this part of the work, I began by analyzing the computational patterns inherent in unstructured mesh applications written using the OP2 DSL. These patterns typically involve indirect memory accesses and accumulation operations – particularly in reductions and read-write kernels – which are highly susceptible to non-determinism in parallel environments. I systematically identified the categories of floating-point operations in OP2 that can lead to violations of bitwise reproducibility, with a focus on those involving reductions (e.g., OP_INC and OP_RW) over shared data.

To address these issues, I proposed two main algorithmic solutions. First, a temporary array-based accumulation scheme was introduced, which ensures that increments are applied in a fixed, deterministic order based on globally unique element IDs. Second, the reproducibility of global reductions was addressed by integrating the ReproBLAS library into OP2, enabling deterministic summation of floating-point arrays regardless of thread count, process layout, or reduction tree structure. These methods were implemented with minimal intrusion into OP2's abstraction, preserving the productivity benefits of the DSL while ensuring deterministic output.

The effectiveness of these methods was demonstrated through test cases such as Aero and MG-CFD, where non-determinism in the results was visibly reduced or eliminated under varying process counts.

Thesis I.2. I introduced a new algorithm that ensures reproducible coloring on distributed, partitioned graphs, independent of the number of partitions. The algorithm builds on M. Osama's graph coloring method originally developed for GPUs, and extends it to ensure full determinism in a distributed setting. I demonstrated how these algorithms – temporary array-based accumulation, deterministic graph coloring, and ReproBLAS-based reductions – can be efficiently mapped to diverse parallelization strategies. I demonstrated that these methods achieve near-optimal performance on multi-core processors, distributed systems, and GPUs, and exhibit practical scalability across industrially representative applications.

Graph coloring is a widely used technique in parallel computing to avoid race conditions during updates to shared data. Traditional coloring methods, however, often depend on the mesh partitioning strategy, introducing variability when the number or configuration of partitions changes. To address this, I developed a novel distributed coloring algorithm that guarantees reproducible color assignments regardless of partitioning. The algorithm extends existing parallel coloring techniques with a deterministic ordering based on global element identifiers and hash-based tie-breaking. This design ensures that color assignments remain consistent even when the mesh is redistributed across varying numbers of MPI processes or affected by load balancing.

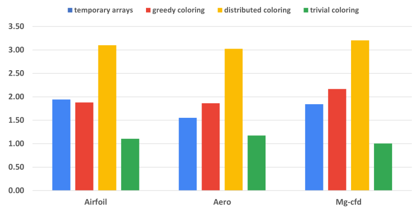
The reproducibility of this coloring method was validated across several OP2 applications, including the industrial-grade Hydra CFD code, where consistent parallel execution is critical for debugging and validation. Moreover, the algorithm is generic and applicable beyond OP2, making it a robust tool for deterministic parallelism in unstructured domains.

One of the key challenges in enabling reproducible execution is minimizing its performance overhead. To this end, the reproducibility techniques – including the new coloring algorithm, temporary array accumulation, and deterministic global reductions – were designed to integrate efficiently with OP2's code generation system and parallel backends. I demonstrated how these algorithms can be mapped onto different parallelization strategies with near-optimal performance across diverse architectures, including multi-core CPUs, distributed-memory clusters, and CUDA-enabled GPUs.

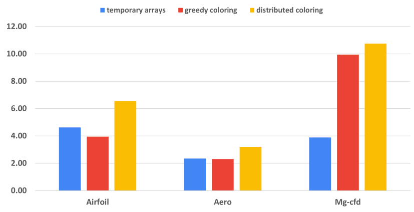
Extensive benchmarking was conducted on representative OP2 applications. Figures 1 and 2 present slowdowns relative to non-reproducible baseline. In the Hydra case study, the reproducibility infrastructure was tested at scale, and results showed that overheads remained within acceptable bounds – especially in light of the benefits offered by deterministic outputs.

On GPU systems, where controlling race conditions is particularly challenging, the proposed methods were successfully deployed using OP2's CUDA backend. Reproducible execution was achieved on Nvidia V100 accelerators without significant restructuring of user code, demonstrating the practical feasibility and portability of the proposed solutions across a broad range of real-world computing environments.

Publications related to this thesis group are: [J1], [C1], [C2], [C3], [C4].



(a)



(b)

Figure 1: Slowdown effect of the different methods compared to the non reproducible version. (**a**) Using 40 MPI-only processes on the Cirrus machine; (**b**) Using one MPI+CUDA GPU process on the Cirrus-GPU machine.

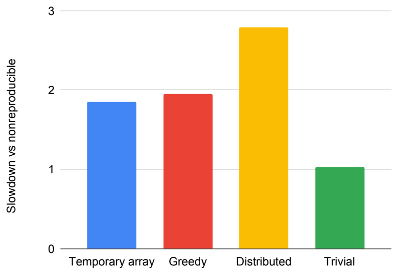


Figure 2: Slowdown of Hydra measured on an 8M mesh, 20 iterations, using the Cirrus-CPU machine.

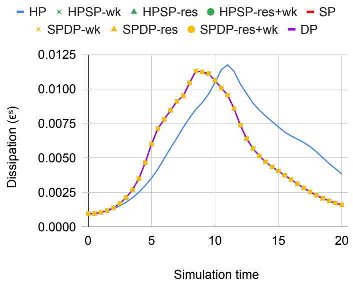
Thesis II.

I introduced a methodology for systematically reducing numerical precision in structured grid applications and measuring its impact on both performance and numerical accuracy. Applying this to a representative turbulent simulation (Taylor-Green vortex), I demonstrated that using 32-bit floating-point representation yields performance gains while maintaining acceptable accuracy, whereas 16-bit usage significantly alters the physical conclusions. Building on this, I extended the OpenSBLI framework to support a mixed-precision strategy, enabling specific state variables and temporary storage to be computed and stored at lower precision. This enables the evaluation of mixed precision configurations in a controlled, quantitative manner, and I showed that carefully selected combinations of 16- and 32-bit precision can preserve accuracy and lead to substantial runtime improvements on both modern CPU and GPU architectures.

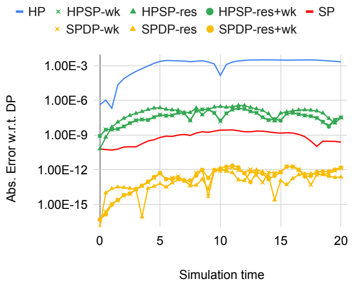
This part of the dissertation addresses the emerging opportunity – driven largely by hardware developments – to use reduced precision arithmetic in scientific simulations. Motivated by the increasing support for 16- and 32-bit floating-point operations on modern GPUs and CPUs, I developed a methodology to systematically evaluate how such precision reductions impact both simulation accuracy and runtime performance.

The methodology was applied to a canonical fluid dynamics problem: the Taylor-Green vortex, a 3D unsteady turbulence benchmark [9]. This problem is well-suited for sensitivity analysis, as it exhibits complex, nonlinear dynamics that can amplify numerical errors and highlight instability in lower precision settings.

Using OpenSBLI as the frontend for code generation and OPS as the backend for parallel execution, I implemented double-precision (FP64), single-precision (FP32), and half-precision (FP16) versions of the sim-



(a) Dissipation



(b) Absolute difference of dissipation, compared to double precision

Figure 3: Numerical accuracy of TGsym app using different precision levels. Mesh size = 256^3, $M = 0.5$, $Re = 800$. The simulations were run for 8000 iterations using the default method.

ulation. Accuracy was assessed using global quantities such as kinetic energy dissipation and numerical error norms, while performance was measured in terms of runtime per iteration and memory usage.

As it can be seen on Figure 3 and Tables 1–3, the results confirm that FP32 precision offers a good balance, delivering significant speedups (especially on GPU platforms) while preserving physically accurate behavior. However, simulations run entirely in FP16 precision showed large deviations in key quantities, leading to altered or unphysical results, particularly near peak dissipation. This reinforces the conclusion that while 16-bit arithmetic may be viable in localized contexts, it cannot be blindly applied across an entire CFD code without compromising scientific validity.

Importantly, the infrastructure developed for this analysis is not specific to the Taylor-Green vortex. Thanks to OpenSBLI's symbolic and backend-agnostic design, the same methodology can be reused to analyze other CFD models using finite difference discretizations on structured grids. As such, this work lays the groundwork for future investigations into precision-aware modeling practices in high-performance simulation workflows.

Building on the findings from full reduced-precision simulations, the thesis point continues on a more granular and flexible approach: mixed-precision computing. Rather than applying the same numerical format across all variables and operations, the strategy here was to assign precision levels selectively, based on the numerical role and stability sensitivity of each component.

To enable this, I extended OpenSBLI's code generation infrastructure to support variable-specific precision control. This required modifications to the symbolic frontend, variable type declarations, and the generated OPS kernel code, ensuring that temporary arrays, work buffers, and conservative state variables could all be defined with differing levels of floating-point precision. The implementation supports combinations such as FP64 for conserved variables and FP32 or FP16 for intermediate operations.

Using the Taylor-Green vortex problem again as a testbed, I evaluated several mixed-precision configurations, such as FP64 state variables with FP32 residuals, or FP32 state with FP16 temporaries. Performance benchmarks show that these configurations offer meaningful speedups without the loss of simulation accuracy observed in the pure FP16 runs. The runtime-per-iteration and memory usage improvements are summarized in Tables 1–3, highlighting gains of more than 2$\times$ speedup with negligible loss of fidelity in key physical quantities.

Publications related to this thesis are: [J2], [C5], [C6].

Publication related to this thesis, but still under review at the time of submission: [J3]

4 Impact and Applications

The methodologies developed in this dissertation address two critical challenges in computational science: numerical reproducibility and precision-aware computing. These contributions have direct applications

	Default		Storesome	
	runtime	speedup	runtime	speedup
HP	42.43 ms	2.43 ×	18.67 ms	2.69 ×
HPSP	47.71 ms	2.16 ×	22.13 ms	2.27 ×
SP	56.95 ms	1.81 ×	25.00 ms	2.01 ×
SPDP	74.02 ms	1.39 ×	37.60 ms	1.34 ×
DP	103.21 ms	1.00 ×	50.31 ms	1.00 ×

Table 1: Runtime per iteration and speedup compared to the double precision run time of the TGsym app are shown for the default and storesome generation methods. Mesh size=256^3, $M = 0.5$, Re=800, 800 iterations. The measurements are performed on a single NVIDIA A100-SXM4-40GB GPU with an AMD EPYC™ 7763 (Milan) CPU.

	Default		Storesome	
	runtime	speedup	runtime	speedup
HP	68.48 ms	5.71 ×	21.39 ms	8.12 ×
HPSP	105.22 ms	3.71 ×	47.31 ms	3.67 ×
SP	185.23 ms	2.11 ×	65.05 ms	2.67 ×
SPDP	249.11 ms	1.57 ×	123.22 ms	1.41 ×
DP	390.71 ms	1.00 ×	173.68 ms	1.00 ×

Table 2: Runtime per iteration and speedup compared to the double precision run of the TGsym app using the default and Storesome generation methods. Mesh size=256^3, Minf=0.5, Re=800, 800 iterations. The measurements are performed on Intel Xeon Platinum 8592+ CPU

	Default		Storesome	
	Memory	Gain	Memory	gain
HP	2.21 GB	4.00 ×	1.09 GB	4.00 ×
HPSP	2.72 GB	3.25 ×	1.60 GB	2.72 ×
SP	4.41 GB	2.00 ×	2.17 GB	2.00 ×
SPDP	5.43 GB	1.63 ×	3.19 GB	1.36 ×
DP	8.83 GB	1.00 ×	4.35 GB	1.00 ×

Table 3: Memory used with the TGsym app and memory gain compared to the double precision run. Size=256^3.

across industrial and academic simulation workflows.

Reproducibility techniques for unstructured mesh applications enable deterministic results in industrial CFD codes (e.g., aerospace and energy systems), where regulatory compliance requires consistent outcomes across hardware configurations. The OP2 integration delivers bitwise reproducibility without compromising parallel performance, enhancing debugging and certification processes. Notably, the HYDRA CFD solver used in aerospace design leverages OP2 for large-scale simulations, where deterministic behavior is vital for both engineering validation and regulatory approval.

Mixed-precision strategies align with modern GPU/accelerator architectures, exploiting lower-precision units for speedups while maintaining accuracy through precision-aware algorithms. This enables faster design cycles in aerodynamics and combustion modeling, particularly beneficial for real-time decision support and large parameter studies. A compelling example is OpenSBLI's deployment for high-fidelity simulations of aerofoil buffet phenomena at the Japan Aerospace Exploration Agency (JAXA), where production runs can last three to four weeks using 120 Nvidia V100 GPUs [10]. In such cases, any performance enhancement – such as that offered by mixed-precision techniques – can significantly reduce computational cost and turnaround time.

Embedding these enhancements within OP2/OPS ensures immediate usability: developers gain reproducibility and precision control without rewriting application logic. The infrastructure also enables future *adaptive precision* approaches, where simulations dynamically adjust accuracy based on local error estimates.

These production-ready solutions balance consistency, efficiency, and accessibility – meeting evolving demands in high-performance scientific computing while lowering barriers for under-resourced research groups.

Use of AI Assistance

Parts of this dissertation benefited from the assistance of AI tools. In particular, ChatGPT (OpenAI), Perplexity.ai, and GitHub Copilot were used for refining the phrasing of technical descriptions, organizing text, checking grammar, and accelerating code development workflows. All results, insights, and critical reasoning remain my own, and the use of these tools was limited to supporting clarity and productivity.

Acknowledgments

I would like to thank my supervisor, István Reguly, for his unwavering support throughout this journey – whether debugging code, shaping this dissertation, or offering personal encouragement during difficult times. His patience and understanding made a real difference.

I'm grateful to Gihan R. Mudalige (University of Warwick) and Neil Sandham (University of Southampton) for warmly welcoming me during my time abroad and treating me as a colleague. Thanks also to Pushpender K. Sharma and David Lusher for their insights and help with the physics and coding aspects of my research.

Special thanks to the Robinson family for hosting me during my stay in Southampton – your kindness created a space where I could truly focus.

To my fellow PhD students and friends – especially Dani, Attila, and the "Ebéd délben?" group – thank you for the discussions, support, and good humor. I'm also grateful to my flatmates Dani, Anna, Tomi, and Dalma for their daily companionship.

To my family: your quiet but constant support meant more than I could express. I especially remember my father, who passed away during this journey – I carry his pride with me.

To those who shared life with me during this time, particularly Ágota – thank you for your patience and care. And to all my friends who supported me in ways big and small: thank you.

I also appreciate the administrative staff, especially Tivadarné Vida, and the international relations office for their help throughout. This work was made possible by support from EKÖP, ÚNKP, Erasmus+, OTKA, and Tamás Zsedrovits, whose timely offer of a grant helped me continue when I needed it most.

Thank you all.

Publications of the author

[J1] B. Siklósi, G. R. Mudalige, and I. Z. Reguly, “Enabling bitwise reproducibility for the unstructured computational motif”, Applied Sciences, vol. 14, no. 2, 2024, issn: 2076-3417. doi: 10.3390/app14020639. [Online]. Available: https://www.mdpi.com/2076-3417/14/2/639

[J2] D. J. Lusher, A. Sansica, N. D. Sandham, J. Meng, B. Siklósi, and A. Hashimoto, “Opensbli v3.0: High-fidelity multi-block transonic aerofoil cfd simulations using domain specific languages on gpus”, Computer Physics Communications, vol. 307, p. 109406, 2025, issn: 0010-4655. doi: https://doi.org/10.1016/j.cpc.2024.109406 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465524003291.

[J3] B. Siklosi, P. K. Sharma, D. J. Lusher, I. Z. Reguly, and N. D. Sandham, “Reduced and mixed precision turbulent flow simulations using explicit finite difference schemes”, Future Generation Computer Systemss, 2025, Under review.

[C1] B. Siklósi, I. Z. Reguly, and G. R. Mudalige, “Bitwise reproducible task execution on unstructured mesh applications”, in 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), 2020, pp. 889–892. doi: 10.1109/CCGrid49817.2020.00015.

[C2] B. Siklósi, “Bitwise reproducible execution of unstructured mesh applications”, in PhD Proceedings Annual Issues of the Doctoral School, Faculty of Information Technology and Bionics, vol. 16, 2021, pp. 165–168.

[C3] B. Siklósi, “Bitwise reproducible execution of unstructured mesh applications”, in PhD Proceedings Annual Issues of the Doctoral School, Faculty of Information Technology and Bionics, vol. 15, 2020, pp. 161–164.

[C4] B. Siklósi, I. Z. Reguly, and G. R. Mudalige, “Bitwise reproducible execution of unstructured mesh applications”, Jedlik Laborato-

ries Reports, vol. 9, no. 2, pp. 13–19, 2020.

[C5] B. Siklósi, “Achieving mixed precision computing with the help of domain specific libraries”, in PhD Proceedings Annual Issues of the Doctoral School, Faculty of Information Technology and Bionics, vol. 17, 2022, pp. 187–189.

[C6] B. Siklósi, “Utilizing the op2 domain specific library for adaptive multi-precision computing”, in PhD Proceedings Annual Issues of the Doctoral School, Faculty of Information Technology and Bionics, vol. 18, 2023, pp. 141–144.

[O1] B. Siklosi, I. Z. Reguly, and G. R. Mudalige, “Heterogeneous cpu-gpu execution of stencil applications”, in 2018 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC), 2018, pp. 71–80. doi: 10.1109/P3HPC.2018.00010.

[O2] B. Keömley-Horváth, G. Horváth, P. Polcz, et al., “The design and utilisation of pansim, a portable pandemic simulator”, in 2022 First Combined International Workshop on Interactive Urgent Supercomputing (CIW-IUS), 2022, pp. 1–9. doi: 10.1109/CIW-IUS56691.2022.00006.

References

[1] “Ieee standard for floating-point arithmetic,” *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, 2019.

[2] D. Goldberg, “What every computer scientist should know about floating-point arithmetic,” *ACM Comput. Surv.*, vol. 23, p. 5–48, 3 1991.

[3] O. Villa, D. Chavarría-Miranda, V. Gurumoorthi, A. Marquez, and S. Krishamoorthy, “Effects of floating-point non-associativity on numerical computations on massively multithreaded systems,” in *CUG Proceedings*, 5 2009.

[4] O. Zienkiewicz, R. Taylor, and J. Zhu, *The Finite Element Method: its Basis and Fundamentals (Seventh Edition)*. Oxford: Butterworth-Heinemann, seventh edition ed., 2013.

[5] J. Demmel, P. Ahrens, and H. D. Nguyen, "Efficient reproducible floating point summation and blas," Tech. Rep. UCB/EECS-2016-121, EECS Department, University of California, Berkeley, 06 2016.

[6] G. Mudalige, M. Giles, I. Reguly, C. Bertolli, and P. Kelly, "Op2: An active library framework for solving unstructured mesh-based applications on multi-core and many-core architectures," 2012 Innovative Parallel Computing (InPar), pp. 1–12, 2012.

[7] I. Z. Reguly, G. R. Mudalige, and M. B. Giles, "Loop Tiling in Large-Scale Stencil Codes at Run-Time with OPS," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 4, pp. 873–886, 2018.

[8] D. J. Lusher, S. P. Jammy, and N. D. Sandham, "OpenSBLI: Automated code-generation for heterogeneous computing architectures applied to compressible fluid dynamics on structured grids," *Computer Physics Communications*, vol. 267, p. 108063, 2021.

[9] S. P. Jammy, C. T. Jacobs, and N. D. Sandham, "Performance evaluation of explicit finite difference algorithms with varying amounts of computational and memory intensity," *Journal of Computational Science*, vol. 36, p. 100565, 2019.

[10] D. J. Lusher, A. Sansica, N. D. Sandham, J. Meng, B. Siklósi, and A. Hashimoto, "OpenSBLI v3.0: High-Fidelity Multi-Block Transonic Aerofoil CFD Simulations using Domain Specific Languages on GPUs," *Computer Physics Communications*, p. 109406, 2024.