The 8th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2022
5-–9 June 2022, Oslo, Norway

VECTORISED SPECTRAL/HP ELEMENT MATRIX-FREE OPERATOR FOR ANISOTROPIC HEAT TRANSPORT IN TOKAMAK EDGE PLASMA

Bin Liu[1], **Chris D. Cantwell**[1], **David Moxey**[2], **Mashy Green**[2] **and Spencer J. Sherwin**[1]

[1] Department of Aeronautics, Faculty of Engineering
Imperial College London, London, UK

[2] Department of Engineering, King's College London
London, UK

Key words: spectral/hp element method, matrix-free operator, SIMD vectorisation, anisotropic heat transport of plasma, tokamak

Abstract. A highly efficient matrix-free Helmholtz operator with single-instruction multiple-data (SIMD) vectorisation is implemented in Nektar++ [1] and applied to the simulation of anisotropic heat transport in tokamak edge plasma. A tokamak is currently the leading candidate for a practical fusion reactor using the magnetic confinement approach to produce electricity through controlled thermonuclear fusion. Predicting the transport of heat in magnetized plasma is important to designing a safe tokamak design. Due to the ionized nature of plasma, the heat conduction of the magnetized plasma is highly anisotropic along the magnetic field lines. In this study, a variational form is proposed to simulate the anisotropic heat transport in magnetized plasma and the details of its mathematical derivation and implementation are presented. To accurately approximate the thermal load of plasma deposition on the wall of tokamak chamber, highly scalable and efficient algorithms are crucial. To achieve this, a matrix-free Helmholtz operator is implemented in the Nektar++ framework, utilising sum-factorisation to reduce the operation count and increase arithmetic intensity, and leveraging SIMD vectorisation to accelerate the computation on modern hardware. The performance of the implementation is assessed by measuring throughput and speed-up of the operators using deformed and regular quadrilateral and triangular elements.

1 INTRODUCTION

Plasma is hot, ionized gas with its constituent atoms split up into electrons and ions, which can move independently. Due to their charged nature, the movement of plasma is strongly influenced by electrostatic and electromagnetic forces, lead to complex behaviour. The primary motivation for this work relates to the prospect of controlled nuclear fusion, which has been under development for over 60 years [2]. A tokamak is currently the leading candidate for a practical fusion reactor based on the magnetic confinement approach to produce controlled thermonuclear fusion. The tokamak must be carefully designed to safely control edge plasma and avoid an excessive thermal load on the plasma-facing components (PFC). The contamination of core plasma and the deterioration of PFC in the vacuum chamber can significantly shorten the global confinement time. In these systems, characteristic rates of heat conduction parallel and perpendicular to the local direction of magnetic field can differ by as many as ten orders of magnitude. Understanding the transport of heat in magnetized plasmas is therefore imperative to furthering efforts towards controlled thermonuclear fusion.

The hot and high pressure core plasma in tokamak is D-shaped and confined by the poloidal and toroidal magnetic fields in the vacuum chamber. Due to the plasma turbulence, particles can escape from the core plasma in this configuration. To regulate the flow of the ionized plasma flux, a magnetic field is employed in a tokamak to create an internal divertor that filters the heavier elements out of the fuel, typically towards the bottom of the reactor. The magnetic field drives the plasma at the lower edge and forms the outer edge of the plasma, the "Scrape-Off-Layer" (SOL), that hits the lithium metal pool. A tokamak featuring a divertor is known as a *divertor tokamak*. In this configuration, the particles escape through a magnetic "gap" (determined by the separatrix), which allows the energy absorbing part of the divertor to be placed outside the plasma. On the other hand, the SOL is defined as the outer edge of the plasma. It is characterised by the open magnetic field lines driving the ionized particles, which transports the energy from the core plasma into the divertor. In divertor plasmas, the SOL absorbs most of the plasma exhaust (particles and heat) and transports it along the magnetic field lines to the divertor plates [13].

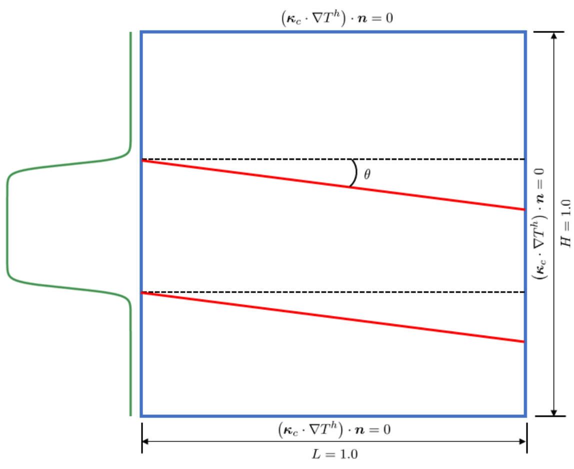

$(\boldsymbol{\kappa}_c \cdot \nabla T^h) \cdot \boldsymbol{n} = 0$
θ
$(\boldsymbol{\kappa}_c \cdot \nabla T^h) \cdot \boldsymbol{n} = 0$
$H = 1.0$
$(\boldsymbol{\kappa}_c \cdot \nabla T^h) \cdot \boldsymbol{n} = 0$
$L = 1.0$

Figure 1: Schematic diagram of computational domain

The study of SOL is very important to the estimation of thermal load on the PFC and the stability of the thermonuclear fusion process, because it is the place where the hot plasma flux gets very close to the cool PFC of the vacuum chamber. The possible contamination of the core plasma and the deterioration of PFCs, due to the excessive thermal load around SOL, can significantly shorten the global confinement time. In order to accurately simulate the anisotropic nature of plasma in the high fidelity plasma edge simulations in tokamak, a simple two-dimensional case of anisotropic thermal conduction in plasma is taken as a test-case to assess the performance of the implemented Helmholtz operator. The schematic diagram of the

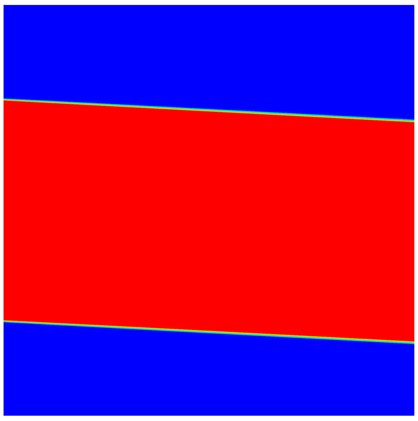


Figure 2: tokamak edge simulation in a unit square

computational domain, Figure 1, represents a small segment of the SOL section in a tokamak, a square domain of unit characteristic length, $L = H = 1.0$, as shown in Figure 2. The traction-free boundary condition is imposed along the top, bottom and right boundaries. A hyperbolic tangent function, illustrated by the green line in Figure 1, is imposed along the left boundary to represent the profile of the SOL. The magnetic field direction $\boldsymbol{B}$ is shown as the red solid line, which has an angle θ with respect to the x-axis.

The process of heat transport can be modelled as anisotropic thermal diffusion. In numerical approximations of the heat transport, any spurious leakage of parallel heat flow into the perpendicular direction would destroy the simulations' predictive capabilities [3]. Modelling anisotropic behaviour in simulation benefits from high-order numerical methods, due to the need to spatially resolve the strong gradient jumps and the lower numerical diffusion of these methods. High-order spectral element methods also have advantageous computational properties due to their high arithmetic intensity, better cache usage and their ability to exploit performance improvements on modern processors. Vectorisation is one such enhancement. Traditional processor registers and math units only hold and operate on one value. However, modern CPUs are able to process multiple values simultaneously by leveraging instruction set extensions. These types of operations are known as single-instruction multiple-data (SIMD) operations, which leads to a fine-grained type of parallelism. As modern hardware now relies on the wide SIMD instructions to boost the computational power, attaining peak performance relies on making efficient use of such instructions. This may be done either through compiler auto-vectorisation, by adjusting the data layout to align more closely with the vectorised nature of SIMD instructions, or explicitly using the intrinsic instructions for the processor.

Algorithmic approaches can be used to further speed up the computation. The construction of spectral/hp element bases using a tensor-product of 1D polynomials, enables the use of sum-factorisation [10] to reduce operation count and, importantly, the amount of data movement.

The volume of data required to be brought from main memory can be reduced further by avoiding the construction of matrices explicitly and calculating the elements of the matrix *on-the-fly* when needed. This *matrix-free* approach increases the arithmetic intensity of the algorithm, reducing memory bottlenecks and improving overall performance. Such optimisations are essential for computational tools to effectively and efficiently leverage exascale platforms.

In this article, a CPU-based Helmholtz matrix-free operator together with sum-factorisation technique and single-instruction, multiple-data (SIMD) vectorisation is proposed and implemented in Nektar++ [1] in order to optimize the computational cost of modelling anisotropic heat diffusion. The derivation of the governing equations, its variational form and the construction of matrix-free operators are described in Section 2. The performance of the implemented CPU-based Helmholtz matrix-free operator together with SIMD vectorisation is evaluated, using a test case for anisotropic thermal conduction in plasma, in Section 3. Finally, conclusions are drawn in Section 4.

2 ANISOTROPIC THERMAL CONDUCTION

2.1 Governing equation

The general form for the heat conduction in plasma is

$$\frac{3}{2}n\frac{dT}{dt} = \nabla\cdot(\boldsymbol{\kappa}_s\nabla T) + Q \tag{1}$$

Due to the ionized nature of plasma and the presence of a strong magnetic field, the dynamics of particles of plasma and the associated energy dissipation is highly anisotropic. The charged particles move rapidly in tight spiral orbits, known as gyro-orbits, along the magnetic field lines, but tremendously slow along the normal direction, e.g., $\kappa_\parallel \gg \kappa_\perp$, typically ten orders of magnitude difference. The thermal conduction tensor therefore becomes a non-diagonal matrix. The temperature gradient is decomposed into three components/auxiliary vectors [5], $\nabla_\parallel T$, $\nabla_\perp T$ and $\nabla_\wedge T$, defined as

$$\nabla_\parallel T = \boldsymbol{b}(\boldsymbol{b}\cdot\nabla T), \qquad \nabla_\perp T = (\boldsymbol{b}\times\nabla T)\times\boldsymbol{b}, \qquad \nabla_\wedge T = \boldsymbol{b}\times\nabla T, \tag{2}$$

with respect to the unit direction of magnetic field $\boldsymbol{b} = \boldsymbol{B}/|\boldsymbol{B}|$, where $\nabla T = \nabla_\parallel T + \nabla_\perp T$. $\nabla_\parallel T$ and $\nabla_\perp T$ are the auxiliary vectors along, and normal, to the magnetic field on the $\boldsymbol{b}-\nabla T$ plane, respectively. The auxiliary vector $\nabla_\wedge T$ accounts for the direction of electromagnetic induction and is normal to the $\boldsymbol{b}-\nabla T$ plane.

It is shown by Goedbloed & Poedts [5] that if a second rank tensor $\boldsymbol{\kappa}_s$ representing the anisotropic transport coefficients is symmetric with respect to rotations about the magnetic field $\boldsymbol{b}$, it implies that $\boldsymbol{\kappa}_s$ can only have three independent elements, e.g., $\kappa_\parallel$, $\kappa_\perp$ and $\kappa_\wedge$, similar to the rotational tensor and have the form below

$$\boldsymbol{\kappa}_s = \begin{bmatrix} \kappa_\perp & -\kappa_\wedge & 0 \\ \kappa_\wedge & \kappa_\perp & 0 \\ 0 & 0 & \kappa_\parallel \end{bmatrix} \tag{3}$$

Assuming the magnetic field is parallel to the 3rd axis of ∇T and the temperature gradient can be expressed as $\nabla T = (\partial_1 T)\hat{\boldsymbol{e}}_1 + (\partial_2 T)\hat{\boldsymbol{e}}_2 + (\partial_{\|} T)\hat{\boldsymbol{e}}_{\|}$ and

$$\begin{aligned} \nabla_{\|} T &= (\partial_{\|} T)\hat{\boldsymbol{e}}_{\|};\ \nabla_{\wedge} T = -(\partial_2 T)\hat{\boldsymbol{e}}_1 + (\partial_1 T)\hat{\boldsymbol{e}}_2; \\ \nabla_{\perp} T &= (\partial_1 T)\hat{\boldsymbol{e}}_1 + (\partial_2 T)\hat{\boldsymbol{e}}_2 \end{aligned} \tag{4}$$

Subsequently the tensor and vector product can be written as

$$\begin{aligned} \boldsymbol{\kappa}_s \cdot \nabla T &= \begin{bmatrix} \kappa_\perp & -\kappa_\wedge & 0 \\ \kappa_\wedge & \kappa_\perp & 0 \\ 0 & 0 & \kappa_{\|} \end{bmatrix} \cdot \begin{bmatrix} \partial_1 T \\ \partial_2 T \\ \partial_{\|} T \end{bmatrix} \\ &= \big((\kappa_\perp \partial_1 T) - (\kappa_\wedge \partial_2 T)\big)\hat{\boldsymbol{e}}_1 + \big((\kappa_\wedge \partial_1 T) + (\kappa_\perp \partial_2 T)\big)\hat{\boldsymbol{e}}_2 \\ &\quad (\kappa_{\|}\partial_{\|} T)\hat{\boldsymbol{e}}_{\|} \\ &= \kappa_{\|}\nabla_{\|} T + \kappa_\wedge \nabla_\wedge T + \kappa_\perp \nabla_\perp T \\ &= \kappa_{\|}\boldsymbol{b}(\boldsymbol{b}\cdot\nabla T) + \kappa_\perp\big(\nabla T - \boldsymbol{b}(\boldsymbol{b}\cdot\nabla T)\big) + \kappa_\wedge \boldsymbol{b}\times\nabla T \end{aligned} \tag{5}$$

Therefore the general form of anisotropic thermal conduction of magnetized plasma in Equation (1) can be recast into the form below

$$\begin{aligned} \frac{3}{2}n\frac{dT}{dt} &= \nabla\cdot\Big[\kappa_{\|}\boldsymbol{b}(\boldsymbol{b}\cdot\nabla T) + \kappa_\perp\big(\nabla T - \boldsymbol{b}(\boldsymbol{b}\cdot\nabla T)\big) + \kappa_\wedge\boldsymbol{b}\times\nabla T\Big] + Q \\ &= \nabla\cdot\Big[\kappa_{\|}(\boldsymbol{b}\otimes\boldsymbol{b})\cdot\nabla T + \kappa_\perp(\boldsymbol{I} - \boldsymbol{b}\otimes\boldsymbol{b})\cdot\nabla T + \kappa_\wedge\boldsymbol{b}\times\nabla T\Big] + Q \\ &= \nabla\cdot\Big[\big((\kappa_{\|} - \kappa_\perp)(\boldsymbol{b}\otimes\boldsymbol{b}) + \kappa_\perp\boldsymbol{I}\big)\cdot\nabla T\Big] + \nabla\cdot[\kappa_\wedge\boldsymbol{b}\times\nabla T] + Q \\ &= \nabla\cdot\big(\boldsymbol{\kappa}_c\cdot\nabla T\big) + \nabla\cdot[\kappa_\wedge\boldsymbol{b}\times\nabla T] + Q \end{aligned} \tag{6}$$

where $\boldsymbol{I}$ is an identity matrix. $\boldsymbol{\kappa}_c$ is defined as the thermal conductivity tensor

$$\begin{aligned} \boldsymbol{\kappa}_c &= (\kappa_{\|} - \kappa_\perp)(\boldsymbol{b}\otimes\boldsymbol{b}) + \kappa_\perp\boldsymbol{I} \\ &= (\kappa_{\|} - \kappa_\perp)\begin{bmatrix} b_x^2 & b_x b_y \\ b_x b_y & b_y^2 \end{bmatrix} + \begin{bmatrix} \kappa_\perp & 0 \\ 0 & \kappa_\perp \end{bmatrix} \\ &= \begin{bmatrix} (\kappa_{\|} - \kappa_\perp)b_x^2 + \kappa_\perp & (\kappa_{\|} - \kappa_\perp)b_x b_y \\ (\kappa_{\|} - \kappa_\perp)b_x b_y & (\kappa_{\|} - \kappa_\perp)b_y^2 + \kappa_\perp \end{bmatrix} \end{aligned} \tag{7}$$

If there is an angle θ defining the direction of the 2D magnetic field, $\boldsymbol{b}$ can be defined as $\boldsymbol{b} = [b_x, b_y]' = [cos(\theta), sin(\theta)]' = [cs, ss]'$, where the superscript ($'$) denotes the transpose operator. Consequently, $b_x^2 = cs^2$, $b_x b_y = cs\ ss$ and $b_y^2 = ss^2$.

In this study, since it is assumed that the anisotropic thermal conduction in magnetized plasma is two-dimensional, the induction direction $\kappa_\wedge$ in the 3rd dimension (the term $\nabla\cdot[\kappa_\wedge\boldsymbol{b}\times\nabla T]$ in Equation (6)) is neglected. Therefore, the implemented strong form of two-dimensional anisotropic thermal conduction becomes

$$\frac{3}{2}n\frac{dT}{dt} = \nabla\cdot\begin{bmatrix} \big((\kappa_{\|} - \kappa_\perp)cs^2 + \kappa_\perp\big)\partial_x T + \big((\kappa_{\|} - \kappa_\perp)cs\ ss\big)\partial_y T \\ \big((\kappa_{\|} - \kappa_\perp)cs\ ss\big)\partial_x T + \big((\kappa_{\|} - \kappa_\perp)ss^2 + \kappa_\perp\big)\partial_y T \end{bmatrix} + Q \tag{8}$$

2.2 Key parameters

Braginskii's transport coefficients are widely used in tokamak edge modelling. The Braginskii transport coefficients [6, 7, 8] for ions (i) and electrons (e) are defined in Equation (9).

$$\kappa_{\parallel}^{e} = 3.2\frac{nk_BT_e\tau_e}{m_e};\ \kappa_{\perp}^{e} = 4.7\frac{nk_BT_e}{m_e\omega_{ce}^2\tau_e};\ \kappa_{\wedge}^{e} = \frac{5}{2}\frac{nk_BT_e}{m_e\omega_{ce}} \tag{9a}$$

$$\kappa_{\parallel}^{i} = 3.9\frac{nk_BT_i\tau_i}{m_i};\ \kappa_{\perp}^{i} = 2.0\frac{nk_BT_i}{m_i\omega_{ci}^2\tau_i};\ \kappa_{\wedge}^{i} = \frac{5}{2}\frac{nk_BT_i}{m_i\omega_{ci}} \tag{9b}$$

where T_α, $\omega_{c\alpha}$, τ_α and m_α respectively are the temperature, the cyclotron frequency (or gyro-frequency), the collision time and the mass of electrons ($\alpha = e$) and ions ($\alpha = i$). k_BT [K] $= |e|T$ [eV], where k_B is the Boltzmann's constant and $|e|$ is the absolute value of the charge on the electron. 1.0 [eV] $\approx 1.6\times10^{-19}$ [J] $\approx 1.16\times10^{4}$ [K]. The cyclotron frequency of the electron and the ion are defined as

$$\omega_{ce} = \frac{eB}{m_e};\quad \omega_{ci} = \frac{eBZ}{m_i} = \frac{eBZ}{m_pA} \tag{10}$$

where e, m_p and B are the positive elementary charge, the mass of proton and the magnitude of magnetic field. $A = m_i/m_p$. Z is the ion charge state. The above definitions of cyclotron frequency is in SI unit. In Gaussian units, the Lorentz force in Equation (10) differs by a factor of $1/c$, where c is the speed of light. The collision time of the electrons and ions are defined as

$$\tau_e = \frac{3\sqrt{m_e}(k_BT_e)^{3/2}}{4n\sqrt{2\pi}\lambda e^4} = 6\sqrt{2\pi^3}\frac{\epsilon_0^2\sqrt{m_e}}{e^4}\frac{(k_BT_e)^{3/2}}{Z^2n\lambda} \tag{11a}$$

$$\tau_i = \frac{3\sqrt{m_i}(k_BT_i)^{3/2}}{4n\sqrt{\pi}\lambda e^4} = 12\sqrt{\pi^3}\frac{\epsilon_0^2\sqrt{m_p}}{e^4}\frac{(k_BT_i)^{3/2}\sqrt{A}}{Z^4n\lambda} \tag{11b}$$

where $\lambda = ln(\Lambda)$ is the Coulomb logarithm. Noted that λ depends on the type of collisions, and is typically between 10 and 20 in a fusion plasma [11]. ϵ_0 is the permittivity of free space.

Due to the anisotropic nature of the magnetized plasma, the magnitudes of diffusion coefficients for ions and electrons are very disparate, so one or other might be neglected [9]. Assuming B is of order unity (in Tesla) and $T_e \approx T_i$, so $\kappa_\parallel \gg \kappa_\perp$. Hence the thermal conductivity coefficients in Equation (8) are chosen as $\kappa_\parallel \approx \kappa_\parallel^e$ and $\kappa_\perp \approx \kappa_\perp^i$. Since the two-dimensional anisotropic thermal conduction is considered in this study, $\kappa_\wedge^e = \kappa_\wedge^i = 0$. By substituting Equation (10) and Equation (11) into $\kappa_\parallel^e$ and $\kappa_\perp^i$ in Equation (9), the $\kappa_\parallel$ and $\kappa_\perp$ can be written as

$$\kappa_\parallel = 19.2\sqrt{2\pi^3}\frac{1}{\sqrt{m_e}}\frac{\epsilon_0^2}{e^4}\frac{(k_BT_e)^{5/2}}{Z^2\lambda}\qquad \kappa_\perp = \frac{1}{6\sqrt{\pi^3}}\frac{1}{m_i}\Big(\frac{nZe}{B\epsilon_0}\Big)^2\frac{(m_pA)^{3/2}\lambda}{\sqrt{k_BT_i}} \tag{12}$$

2.3 Variational formulation

The anisotropic steady diffusion equation can be formulated as the inhomogeneous Helmholtz equation with the scalar constant $\lambda = 0$, given by

$$\nabla\cdot(\boldsymbol{\kappa}_c\cdot\nabla T) - \lambda T = Q \tag{13}$$

where $\boldsymbol{\kappa}_c$, λ and Q are the anisotropic diffusion tensor, reaction coefficient and a forcing term in the field, respectively. Applying the Galerkin method of weighted residuals to Equation (13) and the divergence theorem on the Laplacian operator, its variational form satisfies

$$\begin{aligned}\int_\Omega [\nabla v^h(\boldsymbol{\kappa}_c \cdot \nabla T^h)]d\Omega + \lambda \int_\Omega [v^h T^h] d\Omega &= -\int_\Omega [v^h Q] d\Omega \\ &+ \int_\Gamma [v^h(\boldsymbol{\kappa}_c \cdot \nabla T^h)\cdot \boldsymbol{n}]d\Gamma \qquad (14)\end{aligned}$$

for all test functions v^h. Now dropping the h superscript for clarify, we can expand T^h and v^h in terms of the basis functions on each element,

$$T^e = \sum_i \hat{T}^e \psi_i^e,$$

where the superscript ("e") refers to elemental quantities. Expressing as a matrix system, we arrive at

$$\begin{aligned} & \boldsymbol{L}^e\hat{\boldsymbol{T}}^e + \lambda \boldsymbol{M}^e \hat{\boldsymbol{T}}^e = -\boldsymbol{B}\boldsymbol{W}^e\boldsymbol{Q}^e + \boldsymbol{\Gamma} \qquad (15) \\ \implies \quad & \boldsymbol{H}^e = \boldsymbol{L}^e + \lambda \boldsymbol{M}^e \end{aligned}$$

where the matrices $\boldsymbol{L}^e$, $\boldsymbol{M}^e$ and $\boldsymbol{B}^e$ respectively are the discrete representations of Laplacian, mass and basis evaluation operators. $\boldsymbol{W}^e$ is a diagonal matrix of quadrature weights and $\hat{\boldsymbol{T}}^e$ is the matrix of elemental basis functions ϕ_i evaluated at the quadrature points, ξ_j. For the non-zero surface flux, the vector $\boldsymbol{\Gamma}$ is introduced to represent the boundary integral term in equation (14). Therefore, the Helmholtz operator for each element can be defined as $\boldsymbol{H}^e$ in equation (15). The detailed formulation of $\boldsymbol{L}^e$ and $\boldsymbol{M}^e$ can be written as

$$\boldsymbol{L}^e = (\boldsymbol{D}^e\boldsymbol{B})\boldsymbol{g}^e(\boldsymbol{D}^e\boldsymbol{B})^T\boldsymbol{W}^e \qquad (16a)$$

$$\boldsymbol{M}^e = \boldsymbol{B}\boldsymbol{W}^e(\boldsymbol{B})^T \qquad (16b)$$

where $\boldsymbol{D}$, $\boldsymbol{B}$ and $\boldsymbol{W}$ respectively are the one-dimensional derivative matrix, basis matrix and diagonal weight of numerical integration in the standard region element $\Xi \subset [-1,1]^d$ (uniquely defined for each element type). The metric, $\boldsymbol{g}^e$, incorporates the diffusion tensor $\boldsymbol{\kappa}_c$ and the inverse of Jacobian matrix $\boldsymbol{J}$, as

$$\begin{aligned}\boldsymbol{g}^e &= (\boldsymbol{J}^e)^{-1}\boldsymbol{\kappa}_c(\boldsymbol{J}^e)^{-T} = \begin{bmatrix} g_{00} & g_{01} \\ g_{10} & g_{11}\end{bmatrix} \qquad (17) \\ \boldsymbol{\kappa}_c &= \begin{bmatrix} \kappa_{00} & \kappa_{01} \\ \kappa_{10} & \kappa_{11}\end{bmatrix}; \quad (\boldsymbol{J}^e)^{-1} = \begin{bmatrix} j_{00} & j_{01} \\ j_{10} & j_{11}\end{bmatrix} \\ g_{00} &= \kappa_{00}j_{00}^2 + 2\kappa_{01}j_{00}j_{01} + \kappa_{11}j_{01}^2 \\ g_{01} &= g_{10} = \kappa_{00}j_{00}j_{10} + \kappa_{01}(j_{01}j_{10} + j_{00}j_{11}) + \kappa_{11}j_{01}j_{11} \\ g_{11} &= \kappa_{00}j_{10}^2 + 2\kappa_{01}j_{10}j_{11} + \kappa_{11}j_{11}^2\end{aligned}$$

2.4 Sum-factorisation and matrix-free approach

Recalling that the basis expansions are tensor products of one-dimensional functions for the quadrilateral elements in spectral element method, operators constructed from the tensor product of basis expansions can be decomposed into smaller matrices to perform the matrix-vector operation as a sequence of matrix-matrix operations, thereby reducing the number of total floating-point operations required. This is the sum-factorisation technique. For example, considering the backward transformation that maps basis coefficients to their physical representation, which in two dimensions is expressed as

$$T^e(\xi_i, \xi_j) = \sum_{i,j}^{pq} \hat{T}^e_{ij} \psi^e_{ij}(\xi_i, \xi_j) = \sum_{i}^{p} \sum_{j}^{q} \hat{T}^e_{ij} \phi^e_i(\xi_i) \phi^e_j(\xi_j), \tag{18}$$

where $\hat{T}$ represents the coefficients of basis expansion and i and j index rows and columns respectively. Therefore, this matrix-vector operation can be performed by recasting Equation (18) as

$$T^e(\xi_i, \xi_j) = \sum_{i}^{p} \phi^e_i(\xi_i) \Big[\sum_{j}^{q} \hat{T}^e_{ij} \phi^e_j(\xi_j) \Big] \tag{19}$$

where the $\hat{T}^e_{ij}$ column-vector is rearranged row-by-row into a matrix form. The summation term in the square bracket is evaluated in the form of a matrix-matrix multiplication, with the 1D basis matrix in the first coordinate direction acting on $\hat{T}^e_{ij}$. Subsequently, the result is transposed and the outer summation is evaluated using the 1D basis matrix in the second coordinate direction.

The backward transformation of a quadrilateral element using 2nd order polynomial as basis expansion in each dimension is taken as an example and shown in Figure 3. There are 3 modes in each dimension (a total 9 modes in 2 dimensions) and 9 Gauss quadrature points for exact numerical integration, as shown by the matrix (9×9)-vector (9×1) product in the top-left corner in Figure 3. Applying the sum-factorisation technique, this 9×9 tensor product can be decomposed into two smaller 3×3 matrices and the 9×1 vector is rearranged as a 3×3 matrix. Therefore, the original matrix-vector product is now recast into two matrix-matrix products, as shown in the top-right corner in Figure 3. Consequently, the efficiency of linear algebra operations is significantly improved since the original $\mathcal{O}(P^4)$ operation has been replaced with two $\mathcal{O}(P^3)$ operations.

In the matrix-free approach, only the core finite element operators – the evaluation of the basis functions at the quadrature points, the Gaussian weights and the derivative of the basis functions on the standard reference region – are explicitly computed. More complex operators, such as the Helmholtz operator, and the incorporation of elemental geometric factors are not explicitly constructed, but are computed during evaluation of the operation itself. For the further details on matrix-free operators, please refer to Moxey (2020) [16].

2.5 SIMD vectorisation

In SIMD vectorisation, the same operation can be applied to multiple values during the same clock cycle. For example, Figure 4 shows the comparison of a basic arithmetic operation

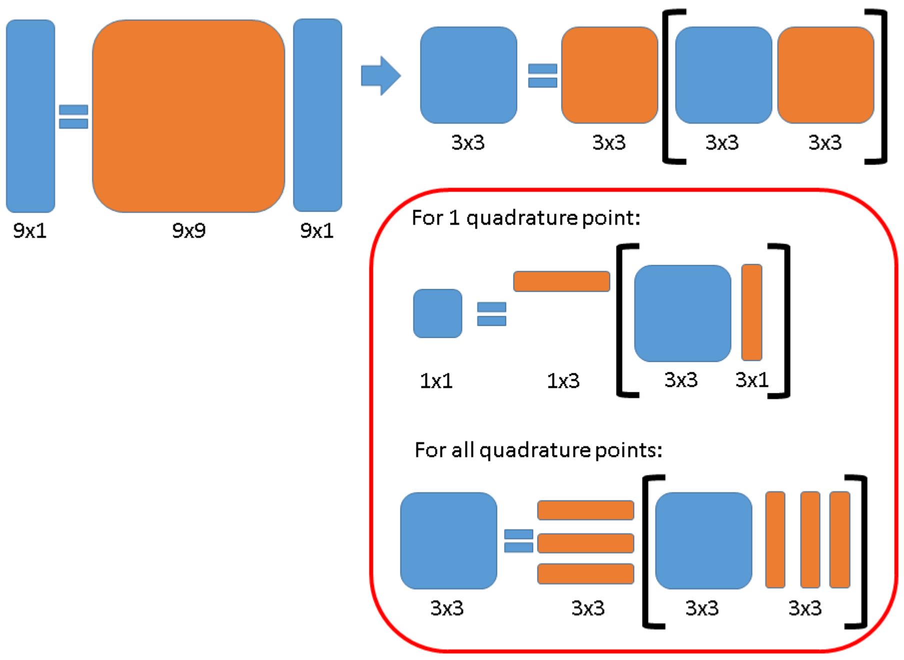


Figure 3: Illustration diagram of sum-factorisation technique using a quadrilateral element with 2nd order polynomial basis functions and 9 Gauss quadrature points

(addition) between the traditional scalar mode (without SIMD) and SIMD vectorisation. The AVX2 (Advanced Vector Extensions 2) instruction set was introduced in 2013 and includes a 256 bit vector register, thereby holding four double-precision floating-point numbers. AVX2 instructions can therefore be used to perform operations on these four values simultaneously. Furthermore, the AVX2 instruction set also support the fused multiply-add (FMA) operations of the form $a \times b + c$, in which the addition and multiplication operation is completed in one cycle. The usage of FMA in the matrix-free operator significantly improves the efficiency of the arithmetic operations involving accumulation of products, e.g., dot products and matrix multiplication.

In Nektar++, to explicitly exploit the SIMD vectorisation, groups of elemental data are interleaved in memory according to the vector width of the architecture. It is assumed that all elements are contiguous in memory. For AVX2, this results in groups of four elements being interleaved, as shown in Figure 5 for second-order quadrilateral elements, where *nElmt* and *nq* refer to the total number of elements and the number of Gauss quadrature points in each element, respectively. If the total number of elements is not divisible by the vector width, additional padding of the array is needed.

2.6 Implementation in Nektar++

Nektar++ is a cross-platform spectral/hp element framework, which aims to improve access to high-order finite element methods for the broader community while being able to solve a

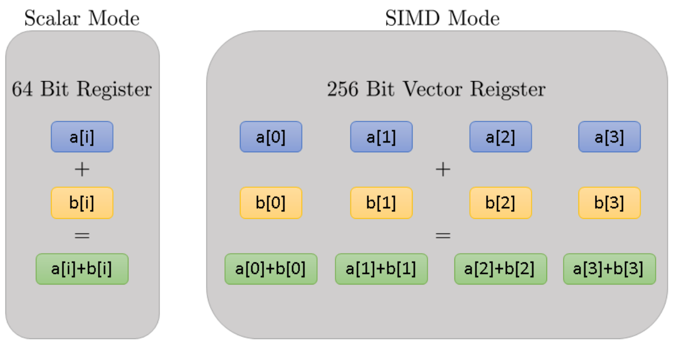


Figure 4: Comparison of basic arithmetic operation between traditional scalar mode and SIMD vectorisation

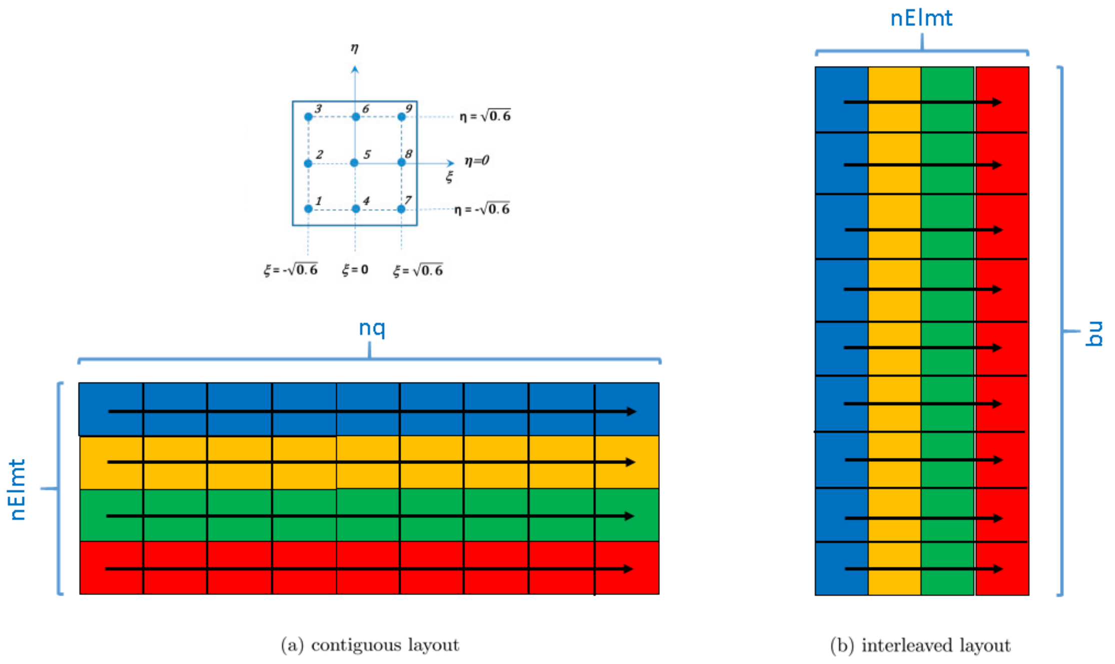


Figure 5: Contiguous and interleaved memory layouts for a cluster of 4 elements. The arrows indicate the memory storage direction

range of problems in high-fidelity scientific computations [1]. It consists of a tiered collection of libraries, each of which implements a different aspect of the formulation, as shown in Figure 6 and described below.

- **LibUtilities** is the most fundamental level in Nektar++ framework. It provides the necessary functions and variables to prepare the element data for spectral element meth-

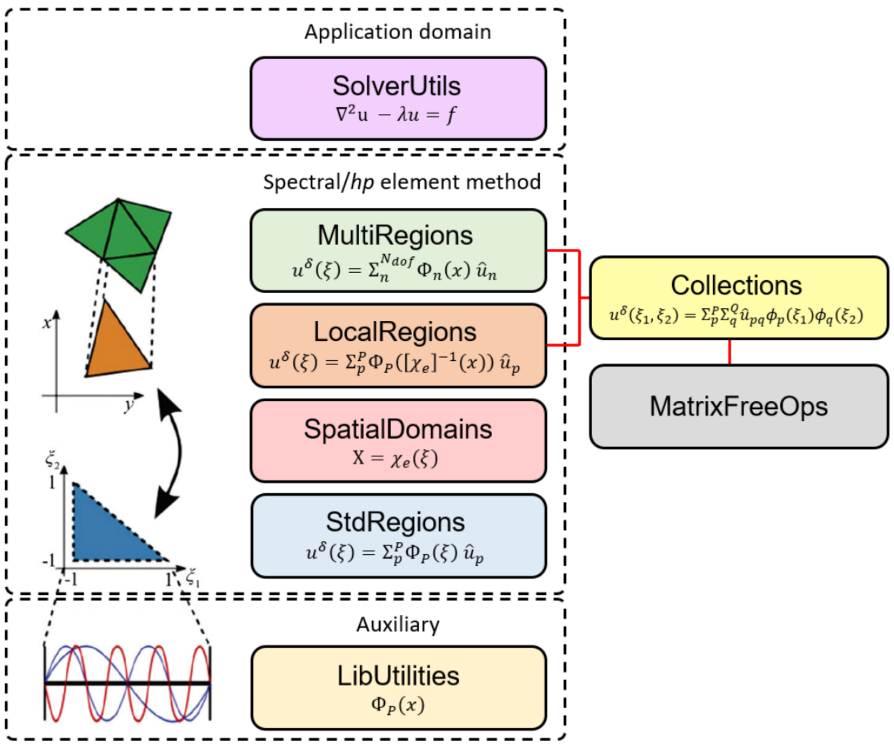

Application domain
SolverUtils
$\nabla^2 u - \lambda u = f$
Spectral/*hp* element method
MultiRegions
$u^\delta(\xi) = \Sigma_n^{N_{dof}} \Phi_n(x)\, \hat{u}_n$
LocalRegions
$u^\delta(\xi) = \Sigma_p^P \Phi_P([\chi_e]^{-1}(x))\, \hat{u}_p$
SpatialDomains
$X = \chi_e(\xi)$
StdRegions
$u^\delta(\xi) = \Sigma_p^P \Phi_P(\xi)\, \hat{u}_p$
Collections
$u^\delta(\xi_1, \xi_2) = \Sigma_p^P \Sigma_q^Q \hat{u}_{pq} \phi_p(\xi_1) \phi_q(\xi_2)$
MatrixFreeOps
Auxiliary
LibUtilities
$\Phi_P(x)$
x
y
ξ_2
ξ_1
1
-1
-1
1

Figure 6: Illustration diagram of libraries and matrix-free operators in Nektar++

ods, which includes the definitions of basis functions, default constants scalars, low-level typedefs, enumerations, mesh partitioning, distributed memory communication, output formats and so on.

- **StdRegions** contains the definitions of reference elements in 1D, 2D and 3D domains. The core finite element operators for each of these elements are also defined, for instance the backward transform, inner product and derivative operators.

- **SpatialDomains** library holds the elemental geometric information, e.g., vertex information and curve information, the mapping between the reference to physical domains and facility reading in and writing out geometry-related information. These geometric information can then be used to extend the finite element operators that were defined for the standard reference regions in *StdRegions* to the physical elements.

- **LocalRegions** includes the elements occupying the physical domain, which is mapped from the elements in the reference region. In Nektar++, each local region is a standard region and has a spatial domain data structure. The local region inherits common expansion methods from its standard region parent, and it uses its spatial domain information to specialize its operators to its local coordinate system.

- **MultiRegions** the various data structures and methods associated with standard regions, spatial domains and local regions are not specifically dictated by any particular numerical methods. It is at the level of *MultiRegions* that we now combine two fundamental concepts: the idea of a collection of elements together to form a "global" expansion and the idea of how these (local) elements communicate (it is the first place at which we can connect to a specific numerical PDF approximation methods of choice, e.g., continuous/discontinuous Galerkin finite element methods). It prepares the assembly operators and form the global linear system based on a particular numerical method.

- **Collections** library groups similar elements together and performs elemental operations collectively. Various element-level operators are defined in *Collections* library, including the matrix-free operators.
- **MatrixFreeOps** implements the matrix-free version of operators, including backward transform (BwdTrans), physical derivatives (PhysDeriv), inner product (IProduct) and Helmholtz operator.

To leverage the vectorisation in the matrix-free operator in the anisotropic thermal conduction formulation, the Helmholtz operator in Nektar++ is re-formulated as shown in Algorithm 1.

Algorithm 1 Overview of matrix-free evaluation of Helmholtz operator

```
1: Helmholtz($\hat{T}, \omega, \boldsymbol{B}, \nabla\boldsymbol{B}, \boldsymbol{J}, \boldsymbol{\kappa}_c$)
2: for each element group do
3:     $T^h \leftarrow$ BwdTrans($\hat{T}, \boldsymbol{B}$)
4:     $\boldsymbol{out} \leftarrow \lambda\cdot$ IProduct($\hat{T}, \boldsymbol{B}, \omega, |\boldsymbol{J}|$)
5:     $\boldsymbol{D}T^h \leftarrow$ PhysDeriv($T^h, \boldsymbol{D}, (\boldsymbol{J})^{-1}$)
6:     if element is deformed then
7:         for each quadrature point $\eta$ do
8:             $\boldsymbol{g} \leftarrow (\boldsymbol{J}(\eta))^{-1}\boldsymbol{\kappa}_c(\eta)(\boldsymbol{J}(\eta))^{-T}$
9:             $\boldsymbol{D}T^h(\eta) \leftarrow \boldsymbol{g}\boldsymbol{D}T^h(\eta)$
10:        end for
11:    else
12:        $\boldsymbol{g} \leftarrow (\boldsymbol{J})^{-1}\boldsymbol{\kappa}_c(\boldsymbol{J})^{-T}$
13:        for each quadrature point $\eta$ do
14:            $\boldsymbol{D}T^h(\eta) \leftarrow \boldsymbol{g}\boldsymbol{D}T^h(\eta)$
15:        end for
16:    end if
17: end for
18: for each dimension d do
19:     $\boldsymbol{out} \leftarrow \boldsymbol{out}$ + IProduct($\boldsymbol{D}T^h, \partial_d\boldsymbol{B}, \omega, |\boldsymbol{J}|$)
20: end for
21: return $\boldsymbol{out}$
```

In Algorithm 1, *deformed* refers to a non-linear reference-to-world mapping, for which we require the calculation of the metric $\boldsymbol{g}$ for each quadrature point. **BwdTrans**, **PhysDeriv** and **IProduct** are the backward transformation, partial derivative of the physical solution and inner product operations, respectively. The functionality of these operations are,

- **BwdTrans**, $\boldsymbol{B}$: evaluate the solution from the spectral/hp element coefficient space to the physical space based on the given elemental coefficients $\hat{\boldsymbol{u}}$ and the standard basis functions evaluated at the quadrature points $\phi_i(\xi_j)$.
- **PhysDeriv**, $\boldsymbol{D}$: compute the partial derivative of the basis functions ϕ_i at the quadrature points ξ_j in physical space.
- **IProduct**, $\boldsymbol{W}$: compute the inner product of a function with respect to the basis functions with given quadrature weights ω_j.

Therefore, the Helmholtz operator can be re-written as,

$$\begin{aligned}\boldsymbol{H}^e &= \sum_{n=0}^{qp} \Big[\nabla\psi^h(\xi^n)(\boldsymbol{J}^e(\xi^n))^{-1}\Big[\boldsymbol{\kappa}_c(\xi^n)\cdot(\boldsymbol{J}^e(\xi^n))^{-T}\nabla\psi^h(\xi^n)\Big] \\ &\qquad + \lambda\psi^h(\xi^n)\psi^h(\xi^n)\Big]\,|\boldsymbol{J}^e(\xi^n)|\omega^h(\xi^n)\end{aligned} \tag{20}$$

Here, the symbol ξ refers to the location of Gauss points in the quadrature rule and qp is the total number of Gauss quadrature points. Line 9 and 14 in Algorithm 1 refer to the sum-factorisation operation in the matrix-free operator, similar to the term in the square bracket in Equation (19).

3 PERFORMANCE EVALUATION

In this section, the performance of the implemented Helmholtz matrix-free operator with SIMD vectorisation is evaluated using the anisotropic thermal conduction problem. The computational domain is a two-dimensional unit square. Two types of elements (regular/deformed quadrilateral and triangular elements) of different polynomial orders (2 to 8) are used. GCC version 10.2.1 was used to compile Nektar++ v5.1.0 and simulations were run using OpenMPI 4.1.0. To access the performance of implemented SIMD instruction, the following CMake options are turned on during configuration:

- -DCMAKE_CXX_FLAGS="-mavx2 -mfma"
- -DNEKTAR_ENABLE_SIMD_AVX2=ON

The specifications of the CPU used for the study are listed in Table 1.

The following simulation configuration is used to execute the testing cases in parallel using 24 processors.

```
<NEKTAR>
  <COLLECTIONS DEFAULT="MatrixFree"/>
  <CONDITIONS>
    <PARAMETERS>
      <P> m_e          = 9.1096e-31        </P>
      <P> m_p          = 1.6726e-27        </P>
      <P> A            = 1.0               </P>
      <P> Z            = 1.0               </P>
      <P> m_i          = A*m_p             </P>
      <P> T_e          = 116050            </P>
```

Table 1: Specifications of intel CPU used for performance evaluation

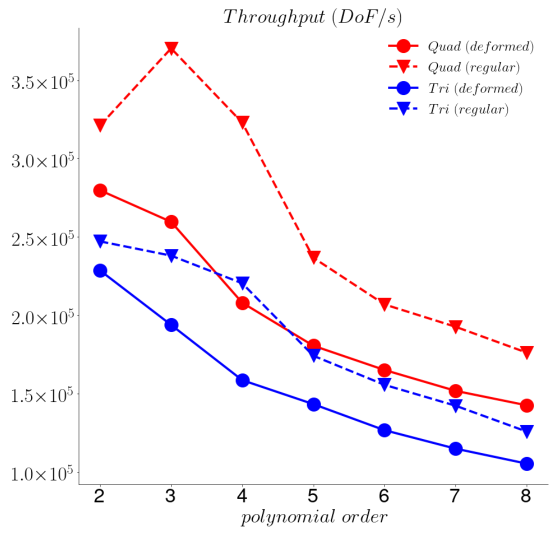
Model	Xeon(R) CPU E5-2670 v3
Architecture	Haswell
SIMD width	256 bit
AVX2 clock speed	2.3 GHz
L3 cache size	30720 KB
Cores per socket	12
Sockets per node	2
Max node GFLOPS/s (AVX2)	883.2
Peak memory bandwidth	136 GB/s

```
    <P> T_i              = T_e                </P>
    <P> epsilon_0        = 8.8542e-12         </P>
    <P> e                = 1.6022e-19         </P>
    <P> k_B              = 1.3807e-23         </P>
    <P> lambda           = 13.0               </P>
    <P> theta            = 0.0                </P>
    <P> n                = 1e18               </P>
    <P> B                = 1.0                </P>
    <P> k_par                      = 19.2 * sqrt(2.0 * PI^3.0) * (1.0/sqrt(m_e
        )) * (epsilon_0^2.0 / e^4.0) * (((k_B * T_e)^(5.0/2.0))/((Z^2.0) *
        lambda)) </P>
    <P> k_perp           = 0.0                </P>
    <P> a                = 500.0              </P>
    <P> NonlinIterTolRelativeL2 = 1e-3        </P>
    <P> NekNonlinSysMaxIterations = 10        </P>
    <P> LinSysRelativeTolInNonlin = 5.0e-2</P>
    <P> NekLinSysMaxIterations = 5000         </P>
    <P> LinSysMaxStorage = 30                 </P>
    <P> PreconMatFreezNumb = 200              </P>
    <P> PreconItsStep = 7                     </P>
  </PARAMETERS>

  <SOLVERINFO>
    <I PROPERTY="EQTYPE"              VALUE="SteadyDiffusion"/>
    <I PROPERTY="Projection"          VALUE="Continuous"      />
  </SOLVERINFO>

  <GLOBALSYSSOLNINFO>
    <V VAR="u">
      <I PROPERTY="GlobalSysSoln"               VALUE="IterativeFull"/>
      <I PROPERTY="LinSysIterSolver"            VALUE="GMRES"        />
      <I PROPERTY="Preconditioner"              VALUE="Diagonal"     />
      <I PROPERTY="MaxIterations"               VALUE="5000"         />
      <I PROPERTY="IterativeSolverTolerance"    VALUE="1e-9"         />
```

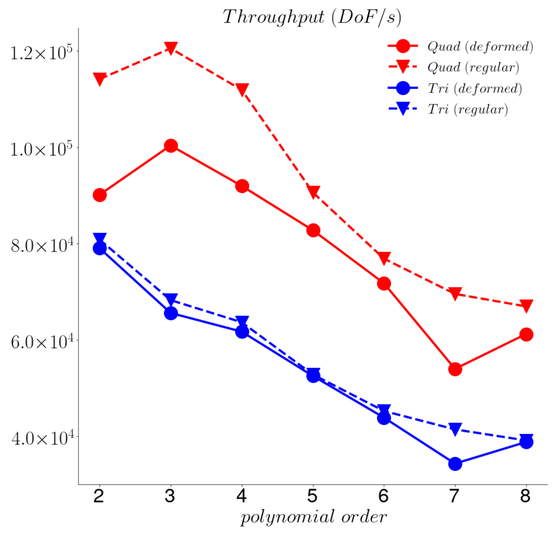
```
            </V>
        </GLOBALSYSSOLNINFO>
    </CONDITIONS>
</NEKTAR>
```

where the first 16 parameters, from m_e to k_perp, are defined for the anisotropic thermal conduction formulation. The last 7 parameters, from NonlinIterTolRelativeL2 to PreconItsStep, are defined to control the convergence and accuracy of the chosen iterative solver in this study, GMRES. For detailed description of these parameters, please refer to the user guide of Nektar++-v5.1.0 [14].

3.1 Throughput analysis



(a)



(b)

Figure 7: Throughput comparison of the matrix-free Helmholtz operator: (a) matrix-free operator with SIMD vectorisation and (b) matrix-free operator without SIMD vectorisation

In this section, we consider the performance of the matrix-free Helmholtz operator from the perspective of throughout, e.g., the number of degrees of freedom processed by the operator per second of execution. It is an indication of the overhead in the implemented matrix-free operators. In this study, both regular and deformed quadrilateral and triangular elements are considered.

In other words, the increase in polynomial order is matched by a corresponding increase in computational work and thus a decrease in throughput. On the other hand, the deformed curvilinear elements are more complex computations, e.g., the computation of deformation gradient tensor $\boldsymbol{J}$ and curvilinear mapping $\mathcal{X}$.

The throughput analyses for the matrix-free operators with/without SIMD vectorisation are shown in Figure 7. For the regular elements, they have the least memory requirement and a clear hierarchy of performance between dimensions and element types. It can be seen that the throughput (Degree-of-freedoms per second) of both quadrilateral and triangular elements proportionally decrease, as the polynomial order of the basis function increases. This means that the operators are performant mostly in a FLOPS-bound regime. Overall, the throughput of the quadrilateral element is almost twice of the triangular element. With the help of SIMD vectorisation, it is found that the throughputs of all types of elements in Figure 7a are about 3 to 3.5 times higher than their counterparts without SIMD vectorisation using the same polynomial order in Figure 7b. The larger value of throughput means that the matrix-free operator with SIMD vectorisation can work more efficiently. This improvement of performance is expected, since the AVX2 (256 bits vector width) can complete four floating point operations simultaneously with one instruction.

Furthermore, the results in Figure 7 also show that the throughput of regular quadrilateral/triangular elements are always higher than those of their deformed counterparts for all polynomial orders. For the deformed curvilinear elements, the significant decrease of throughput is due to the presence of curvature of the elements, which requires the computations of the curvilinear mapping $\mathcal{X}$ and its deformation gradient tensor $\boldsymbol{J}$. This conclusion is also drawn in Kronbichler & Wall (2018) [15] and Moxey et. al. (2020) [16]. However, this difference becomes less obvious for the triangular elements without the help of SIMD vectorisation, as shown by the blue lines in Figure 7b.

3.2 Speed-up analysis and roofline model

To assess the efficiency of the implemented matrix-free operator and SIMD vectorisation, the execution time of Helmholtz matrix-free operator and its corresponding speed-up (the ratio between the execution time of matrix-based operator and the execution time of matrix-free operator) are plotted in Figure 8 for the quadrilateral and triangular elements, respectively. Figure 8a shows that the matrix-free operators with SIMD vectorisation is approximately two order of magnitude faster than the matrix-based operator for low polynomial orders and one order of magnitude faster for higher polynomial orders. On the other hand, without SIMD vectorisation, a relative increase of execution time, more than one order of magnitude with respect to the matrix approach, can still be observed for polynomial orders above six, shown by the blue line in Figure 8.

The calculated speed-up of the Helmholtz matrix-free operators with and without vectorisation is shown in Figure 9. By comparing the performance of matrix-free operators in Figure 9a and Figure 9b, it is observed that the application of SIMD vectorisation effectively speeds up the execution of matrix-free operators by approximately 3 to 3.5 times for both quadrilateral and triangular elements, compared with the non-SIMD case.

To quantify the performance of the implemented matrix-free operators with SIMD instruction in relation to the theoretical peak performance of the hardware, the roofline performance model is employed to assess their computational performance. In this analysis, two performance bottlenecks are considered: floating point operation per second (FLOPS) and memory bandwidth, as shown in Figure 10, where arithmetic intensity (α) is defined as $\alpha =$ GFLOPS/memory

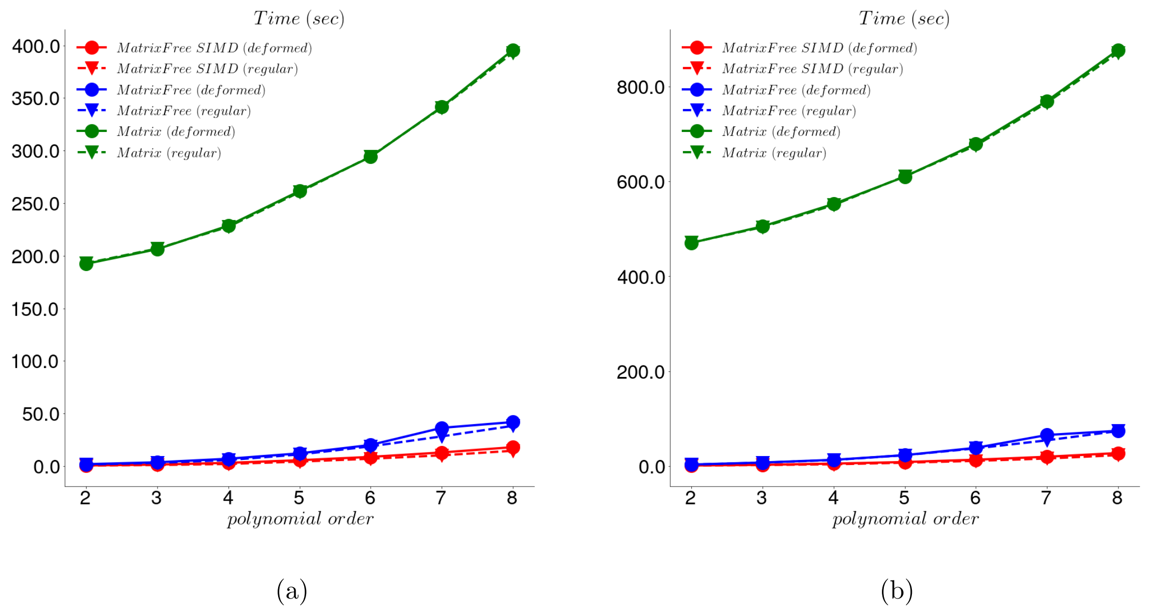


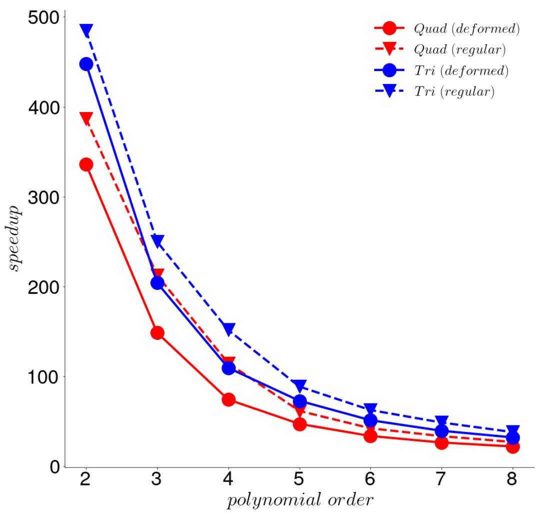
Figure 8: Execution time comparison of the matrix-free and matrix-based Helmholtz operators: (a) quadrilateral element and (b) triangular element

bandwidth and the corresponding maximum GFLOPS is defined as

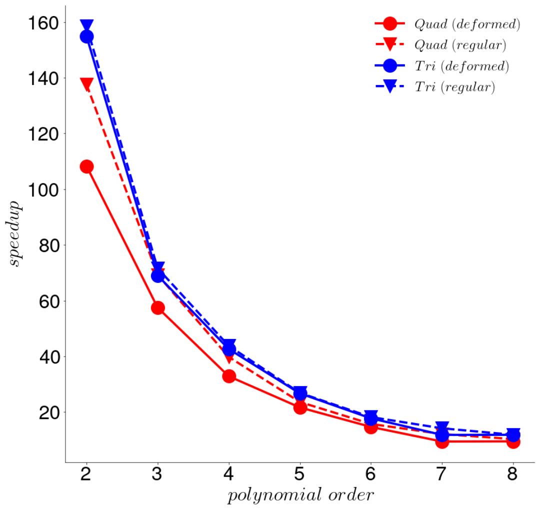
$$\text{Max GFLOPS} = \min(\text{peak GFLOPS}, \text{peak memory bandwidth} \times \alpha) \tag{21}$$

To evaluate the GFLOPS and memory bandwidth of each simulation, the Likwid performance monitoring and benchmarking suite [17] is employed to determine key performance characteristics. The tests were done using Likwid version 4.3.0 and the MEM_DP performance group to record memory bandwidth and GFLOPS/s attained using the likwid-mpirun utility for parallel execution. On the other hand, the peak memory bandwidth is recorded by Likwid using likwid-bench with the stream_mem_avx_fma test.

In the results plotted in Figure 10, the arithmetic intensity increases proportionally with respect to the polynomial orders of the basis function, for each of the different element types considered. This is a beneficial property of high-order methods, enabling them to more fully utilize the CPU and being less constrained by the memory bandwith. It is evident that the regular quadrilateral/triangular elements are FLOPS-bound, whereas the deformed elements are predominantly memory-bound. This observation agrees well with the discussion presented for the throughput analyses. In other words, the performance of implemented matrix-free operator is limited by the memory bandwidth/FLOPS for the deformed/regular elements. Furthermore, it is also noticed that the cases with SIMD vectorisation and FMA instruction set have a GFLOPS value which is again approximately four times higher than without SIMD vectorisation and FMA.



(a)



(b)

Figure 9: Speed-up comparison of matrix-free and matrix-based Helmholtz operators: (a) operator with SIMD vectorisation and (b) operator without SIMD vectorisation

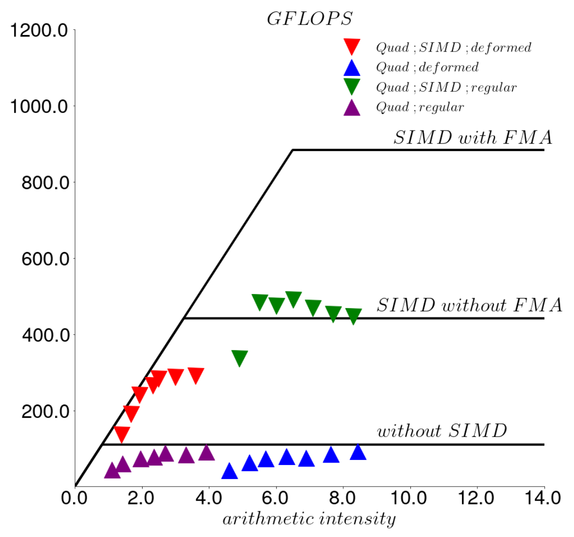
4 CONCLUSIONS AND OUTLOOK

The implementation of a CPU-based Helmholtz matrix-free operator has been described and its performance with/without SIMD vectorisation was assessed in the context of the anisotropic thermal conduction problem. Both regular/deformed quadrilateral/triangular elements of different polynomial orders (2 to 8) were considered. It was found that the matrix-free approach leads to a substantially reduced computation time of between one and two orders of magnitude, compared with a matrix-based approach, due to a reduced effect of memory-bandwidth. SIMD vectorisation effectively speed up the execution of the simulations and it was found that the throughput of the matrix-free operators with SIMD was at least three times higher than their counterparts without SIMD. Although the speed-up reduced with polynomial order, the maximum speed-up and the minimum speed-up observed in this study are approximately 500 times and 10 times, respectively, compared to the matrix-based Helmholtz operator. It was observed that the implemented matrix-free Helmholtz operator is predominantly FLOPS-bound for the regular elements, while its performance is mainly memory-bound for the deformed elements due to the additional data movement.

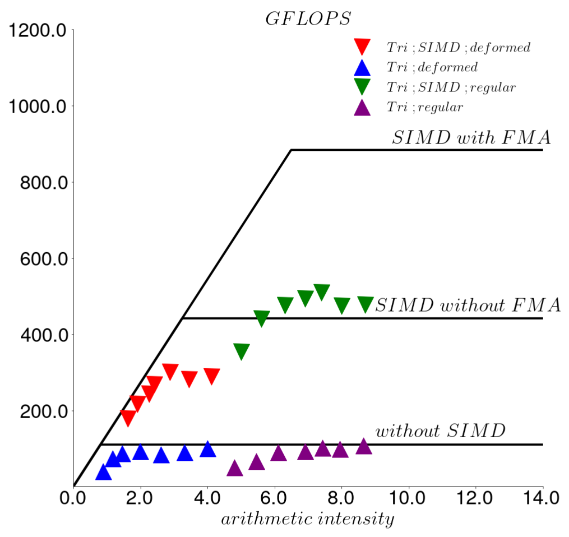
REFERENCES

[1] Cantwell, Chris D., et al. "Nektar++: An open-source spectral/hp element framework." *Computer physics communications.* (2015) **192**: 205-219

[2] Shafranov, Vitalii D. "The initial period in the history of nuclear fusion research at the



(a)



(b)

Figure 10: Roofline analysis the matrix-free Helmholtz operators with/without SIMD vectorisation: (a) quadrilateral element and (b) triangular element

Kurchatov Institute.” *Physics-Uspekhi* (2001) **44.8**: 835.

[3] Sovinec, C. R., et al. ”Nonlinear magnetohydrodynamics simulation using high-order finite elements.” *Journal of Computational Physics* (2004) **195.1**: 355-386.

[4] Dendy, Richard O., ed. ”Plasma physics: an introductory course”. *Cambridge University Press*, 1995.

[5] Goedbloed, JP Hans, J. P. Goedbloed, and Stefaan Poedts. ”Principles of magnetohydrodynamics: with applications to laboratory and astrophysical plasmas”. *Cambridge university press*, 2004.

[6] Braginskii, S. I. ”Transport processes in a plasma, MA Leontovich.” *Reviews of Plasma Physics*, Consultants Bureau, New York (1965).

[7] Edward, Morse. ”Nuclear Fusion”. *Springer International Publishing*, (2018)

[8] Andrew S. Richardson. ”NRL Plasma Formulary”. *US Naval Research Laboratory* (2019)

[9] W. Arter and R. Akers. ”Excalibur equations for neptune proxyapps -draft”. *UK Atomic Energy Authority.* (2020)

[10] Karniadakis, George Em, George Karniadakis, and Spencer Sherwin. ”Spectral/hp element methods for computational fluid dynamics”. *Oxford University Press*, 2013.

[11] Spatschek, Karl-Heinz. "High temperature plasmas: theory and mathematical tools for laser and fusion plasmas". *John Wiley & Sons*, 2013.

[12] Moxey, David, Roman Amici, and Mike Kirby. "Efficient matrix-free high-order finite element evaluation for simplicial elements." *SIAM Journal on Scientific Computing* (2020) **42.3**: C97-C123.

[13] Reinders, Lodewijk Johannes. "The Fairy Tale of Nuclear Fusion". *Springer*, 2021.

[14] Robert M. Kirby, Spencer J. Sherwin, Chris Cantwell and David Moxey. "Nektar++: Spectral/hp Element Framework: User Guide". *Version 5.1.0*, 2021.

[15] Kronbichler, Martin, and Wolfgang A. Wall. "A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers." *SIAM Journal on Scientific Computing* **40.5** (2018): A3423-A3448.

[16] Moxey, David, Roman Amici, and Mike Kirby. "Efficient matrix-free high-order finite element evaluation for simplicial elements." *SIAM Journal on Scientific Computing* **42.3** (2020): C97-C123.

[17] Treibig, Jan, Georg Hager, and Gerhard Wellein. "Likwid: A lightweight performance-oriented tool suite for x86 multicore environments." 2010 39th international conference on parallel processing workshops. *IEEE*, 2010.