The 9th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2024
3–7 June 2024, Lisboa, Portugal

LOWER-ORDER REFINED PRECONDITIONING FOR SPECTRAL/HP ELEMENT METHOD FOR COMPLEX, 3D GEOMETRIES

Parv Khurana[1], Spencer J. Sherwin[2], Julien Hoessler[3], David Moxey[4] and Athanasios Chatzopoulos[5]

[1,2] Department of Aeronautics, Imperial College London, SW7 2AZ London

[1,3,5] CFD Methodology Group, McLaren Racing Limited, GU21 4YH Woking

[4] Department of Engineering, King's College London, WC2R 2LS London

[1]p.khurana22@imperial.ac.uk

Key words: Spectral hp/element methods, Preconditioning, Lower-order refined, Pressure Poisson, Incompressible Navier-Stokes

Summary. The current work presents the incorporation of the *Lower-Order Refined* (LOR) preconditioner within the incompressible Navier-Stokes equations solver of the open-source spectral/*hp* element method framework Nektar++. This preconditioner is constructed on a low-order (P=1) finite element discretisation spectrally equivalent to a given high-order discretisation. The LOR preconditioner provides advantages like low cost operator evaluations, constant memory requirement per degree of freedom, minimal sensitivity to high aspect-ratio elements and controlled iterative condition number with increasing problem size. The contribution extends to developing a functional LOR preconditioner tailored for mixed-element 3D mesh configurations, encompassing both prismatic and tetrahedral elements, allowing usage for complex geometries requiring unstructured 3D meshes with boundary layers. Notably, the Nektar++ implementation is the first instance of using the LOR preconditioner with a modal expansion basis function. This study presents the iterative performance and scaling of the proposed preconditioner to solve the pressure Poisson system arising from the incompressible Navier-Stokes equations solver for industrial test cases relevant to the context of race-car aerodynamics.

1 INTRODUCTION

Spectral hp/element methods [1] have gained recent prominence in simulating high Reynolds (Re) number incompressible flows around complex, 3D industrial geometries, including multi-element wings [2, 3]. The iterative solution of these methods is more computationally expensive than their lower-order counterparts because of the typically large problem sizes, various length scales involved, anisotropic spatial discretisation and denser linear systems. These factors lead to highly ill-conditioned matrices, and efficient preconditioning techniques are needed to alleviate the computational cost of these methods, facilitating their adoption in industrial applications. Furthermore, the preconditioning technique must scale to leverage modern high-performance computing systems with many CPU cores, adding design considerations. The developments in designing an accurate, robust, efficient, and scalable preconditioner will pave the way towards adopting higher-order (HO) methods in industries characterised by fast turnaround times, such as Formula 1.

The focus of this study is the efficient design and use of preconditioning techniques for the incompressible Navier-Stokes equation (IncNS) solver in the open-source spectral/*hp* methods framework Nektar++

[4]. The solver is implemented using the velocity correction scheme [5], also known as the high-order splitting scheme, which splits the incompressible Navier-Stokes equations into a pressure Poisson system and a Helmholtz problem that includes the viscous terms. Notably, the pressure Poisson system resulting from the velocity correction scheme presents an elliptic partial differential equation (PDE) problem, posing a numerically challenging task for preconditioning due to its stiffness compared to the Helmholtz problem arising from the viscous terms. Currently, available preconditioners in Nektar++ are not well-suited to tackle these issues, necessitating the development of specialised preconditioning strategies for the pressure Poisson system.

A potential candidate is the *Lower-order refined* (LOR) preconditioner, also known as the *SEMFEM* preconditioner, which uses a spectrally equivalent [6] lower-order (P=1) discretisation to precondition the high-order problem [7, 8, 9, 10]. The motivation behind this approach is to achieve a sparser preconditioner by leveraging the sparsity of lower-order operators compared to their higher-order counterparts. The advantages of using a lower-order discretisation to improve conditioning properties for preconditioning the finite element method were observed previously [11, 12, 13].

Another advantage of using a LOR discretisation is its constant memory requirement per degree of freedom, making it less memory-intensive than higher-order discretisation [14]. These methods show minimal sensitivity to high aspect-ratio elements [9], making them suitable for complex geometries. A bounded iterative condition number behaviour with increasing problem size was later established by many works [15, 16, 17]. Finally, many preconditioning methods are readily available in existing software packages, like the algebraic multigrid (AMG), to solve the lower-order problem (P=1) for elliptic PDEs generated from the LOR discretisation.

Recent years have witnessed significant progress in the development of such lower-order preconditioning techniques, especially for meshes with rectangular and hexahedral elements with Lagrange polynomial basis and GLL-based quadrature [8, 9, 10, 18, 19, 20]. However, to the authors' knowledge, there is limited work for triangular elements in 2D and no available LOR preconditioner implementation for prism, tetrahedron, and pyramid elements in 3D. Canuto et al. [10] examined the performance and condition numbers for different $\mathbb{P}1$ and $\mathbb{Q}1$ type finite elements in 2D and 3D. Subsequently, Bello-Maldonado and Fischer [9] demonstrated a lower-order finite element preconditioner for a Poisson solver using a $\mathbb{P}1$ discretisation (i.e. triangles in 2D and tetrahedrons in 3D) in the lower-order space, albeit for rectangular and hexahedral high-order spectral elements. A low-order finite-element approach was also tested by Schöberl et al. [21] for triangular and tetrahedral meshes, albeit in the context of overlapping Schwarz preconditioners, and observed bounded condition number behaviour in both h and P. Chalmers and Warburton [22] attempted the approach on triangular elements and observed that there is no optimum node set that provides the bounded iterative condition behaviour with increasing problem size and predict conditioning challenges if the approach is further extended to prisms, tetrahedrons and pyramids [22, 23].

This work demonstrates the LOR preconditioner for spectral/hp elements in Nektar++ for prismatic and tetrahedron elements in 3D. The quadrilateral and triangle elements implementation in 2D was previously communicated by Khurana et al. [24]. Another research gap addressed in this study is the implementation of the LOR preconditioner for modal or hierarchical expansion basis functions for the higher-order finite element, along with standard higher-order Lagrange-type basis functions. This development allows the application of the LOR preconditioner to mixed element-type meshes necessary for unstructured meshes around complex, 3D geometries. This study investigates the iterative conditioning, performance and scaling of the proposed preconditioner for test cases relevant to the context of race-car aerodynamics.

The structure of the document is as follows: Section 2 provides an overview of the spectral/*hp* method and how it is used to formulate the IncNS solver in Nektar++, followed by the description of

preconditioning with a focus on LOR methodology. The LOR preconditioner's conditioning properties, iterative performance and scalability for the Poisson problem and within the IncNS solver are presented in Section 3. Section 4 summarises the work with a future outlook.

2 Methods

2.1 Spectral/*hp* element method

Using the spectral/hp element method to discretise a partial differential equation $\mathcal{L}(u) = f$ over a domain Ω with appropriate boundary conditions on domain boundaries $\partial\Omega$, the dependent variable is represented using the following expansion in terms of the elemental modes:

$$u^\delta(\mathbf{x}) = \sum_{n=0}^{N_{\text{dof}}-1} \hat{u}_n \Phi_n(\mathbf{x}) = \sum_{e=1}^{N_{\text{el}}} \sum_{n=0}^{N_m^e-1} \hat{u}_n^e \phi_n^e(\mathbf{x}) \tag{1}$$

where the domain has a total of N_{dof} degrees of freedom and $\Phi_n(\mathbf{x})$ are the global basis functions and $\hat{u}_n$ are the expansion coefficients. The domain Ω can be further decomposed into an h-sense by N_{el} non-overlapping elements, where Ω^e signifies an element. For a P-expansion there are N_m^e local polynomial expansion modes within the element Ω^e, with $\phi_n^e(\mathbf{x})$ being the n^{th} local expansion mode within the element Ω^e, and $\hat{u}_n^e$ represents the n^{th} local expansion coefficient within the element Ω^e.

The approximation Eqn. (1) is utilized in a Galerkin formulation. We assume $\mathcal{L}(u)$ is a linear operator. By using the appropriate reduced finite-dimensional function spaces for trial (U^δ) and test (V^δ) functions, we wish to determine an approximate solution $u^\delta \in U^\delta$ that satisfies the integral equation

$$\int_\Omega v^\delta \cdot \mathcal{L}(u^\delta) d\mathbf{x} = \int_\Omega v^\delta f d\mathbf{x} \quad \forall v^\delta \in V^\delta.$$

If $\mathcal{L}(u^\delta)$ contains a second-order differential operator, an integration by parts is performed to obtain the weak form which is denoted by

$$a(v^\delta, u^\delta) = (v^\delta, f).$$

This also allows us to enforce Neumann boundary conditions in a weak form. For a Bubnov-Galerkin projection, where the $U^\delta = V^\delta$, using the approximation (1) and applying integration by parts, this problem can be represented as the following in matrix terms

$$\mathbf{A}\hat{\mathbf{u}} = \hat{\mathbf{f}} \tag{2}$$

where $\mathbf{A}$ is the global matrix (i.e. Helmholtz or Mass matrix) with element $\mathbf{A}[m][n] = a(\Phi_m, \Phi_n)$ and $\hat{\mathbf{f}}$ is the vector of coefficients $\hat{\mathbf{f}}[m] = (\Phi_m, f)$. The choice of basis functions (ϕ_n^e) and the quadrature rule for Eqn. (1) affect the system matrix $\mathbf{A}$. For 3D cases, Ω^e is either one of the following element types: hexahedral, prism, pyramid or tetrahedron.

Within Nektar++, the local expansion or basis functions are built from 1D functions. There are two choices for these functions: a) *modal* or *hierarchical* expansion basis, where the expansion set of order $P-1$ is contained within the expansion set of order P, and b) *nodal* expansion basis which is a non-hierarchical basis consisting polynomials of order P through a pre-decided set of nodal points. In Nektar++, the modal expansion bases are P^{th} order Legendre polynomials ($\psi_p^a(\xi)$) modified at the boundaries to ensure C^0 continuity, where a Gauss-Lobotto-Legendre quadrature is adopted for numerical integration. The nodal expansion bases are P^{th} order Lagrange polynomial ($h_p(\xi)$) using nodal points through the zeros of the Gauss-Legendre-Lobatto integration rule ($\xi_{\mathbf{GLL}}$).

2.2 Incompressible Navier-Stokes equations solver in Nektar++

The IncNS equations solver in Nektar++ is implemented using the spectral/*hp* element method with a continuous Galerkin discretisation. The IncNS equations governing viscous Newtonian fluid flows can be written as follows:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f} \tag{3a}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{3b}$$

where $\mathbf{u}$ is the velocity, p is the specific pressure (including density), $\mathbf{f}$ is the forcing term and ν the kinematic viscosity. The solver uses the Velocity Correction Scheme (VCS) introduced by Karniadakis et al. [5] to decouple the velocity and the pressure system. In the Nektar++ implementation of the IncNS solver, the time is discretised using a backward approximation:

$$\frac{\partial \mathbf{u}}{\partial t}^{n+1} \simeq \frac{\gamma_0 \tilde{\mathbf{u}}^{n+1} - \hat{\mathbf{u}}}{\Delta t} \tag{4}$$

where $\tilde{\mathbf{u}}^{n+1}$ is an intermediate velocity, $\hat{\mathbf{u}}$ is the summation of previous solutions and γ_0 is a constant. The chosen approximation order (J) of the time derivative decides the $\hat{\mathbf{u}}$ and γ_0. To decouple the system, we impose that $\nabla \cdot \tilde{\mathbf{u}}^{n+1} = 0$. The following final decoupled equation systems are obtained:

- Pressure: the strong form equivalent of the pressure equation system is

$$\nabla^2 p^{n+1} = \nabla \cdot \left(\frac{\hat{\mathbf{u}}}{\Delta t} - \mathbf{N}^{*,n+1} \right), \tag{5}$$

  where $\mathbf{N}^{*,n+1}(\mathbf{u}) = [\mathbf{u} \cdot \nabla \mathbf{u}]^{*,n+1}$ is the advection term ($\mathbf{N}$) and the superscript indicates extrapolation from previous solutions according to the order of the time derivative J.

- Velocity:

$$\left(\nabla^2 - \frac{\gamma_0}{\nu \Delta t} \right) \mathbf{u}^{n+1} = -\left(\frac{\gamma_0}{\nu \Delta t} \right) \hat{\mathbf{u}} + \frac{1}{\nu} \nabla p^{n+1} \tag{6}$$

It is useful to note that Eqn (5) and Eqn (6) take the form of a Helmholtz equation, $\nabla^2 u(\mathbf{x}) + \lambda u(\mathbf{x}) = f(\mathbf{x})$, where u is the dependent variable. The pressure equation Eqn.(5) is of the Poisson form (i.e. Helmholtz equation with $\lambda = 0$). Thus, the solution of the Helmholtz equation is the work-horse of the IncNS solver. Using a similar procedure as Section 2.1, The Helmholtz equation can be discretised using the spectral/*hp* element method which results in the following representation in matrix terms,

$$\mathbf{H}\hat{\mathbf{u}} = \mathbf{f} \tag{7}$$

where $\mathbf{H}$ represents the Helmholtz matrix, $\hat{\mathbf{u}}$ are the unknown global coefficients and $\mathbf{f}$ is the inner product of the expansion basis with the forcing function.

2.3 Preconditioning in Nektar++

The system in Eqn. (7) needs to be preconditioned for an iterative solution in the IncNS solver. This is done by applying a preconditioning matrix $\mathbf{M}$ to the system in the following manner:

$$\mathbf{M}^{-1}(\mathbf{H}\hat{\mathbf{u}} - \mathbf{f}) = 0, \tag{8}$$

to achieve an equivalent system with better iterative properties. For the iterative solvers in Nektar++, the preconditioner $\mathbf{M}$ is applied to the residual vector $\hat{\mathbf{r}}_{k+1}$ at every iteration

$$\hat{\mathbf{z}}_{k+1} = \mathbf{M}^{-1}\hat{\mathbf{r}}_{k+1}. \tag{9}$$

where $\hat{\mathbf{r}} = \mathbf{H}\hat{\mathbf{u}} - \mathbf{f}$. For a preconditioner to be effective, the condition number κ of the modified system $\mathbf{M}^{-1}\mathbf{H}$ must be improved and be closer to 1 than $\mathbf{H}$, or $\kappa(\mathbf{M}^{-1}\mathbf{H}) \ll \kappa(\mathbf{H})$.

2.3.1 Lower-order Refined (LOR) preconditioner

For the construction of the LOR preconditioner, the global matrix $\mathbf{H}$ is interpolated from the higher-order spectral/*hp* discretisation to its corresponding low-order (P=1) "refined" finite element discretisation $\mathbf{H}_L$. The conceptual representation of this conversion is seen in Fig. 1, where the LOR mesh on the right is a sub-structured grid of collocated Gauss-Lobatto-Legendre (GLL) nodal points. The application of the preconditioning matrix $\mathbf{M}$ is then formulated as $\mathbf{M}^{-1} = \mathbf{H}_L^{-1}$.

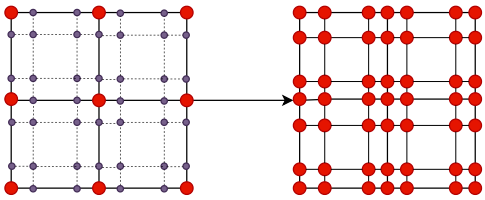


Figure 1: Representation of converting a higher-order finite element grid at P=3 to its lower-order P=1 equivalent for the LOR preconditioner. The red dots represent the points on the linear mesh, and the purple dots represent the quadrature points within a linear element. The resulting LOR mesh only contains linear DOF. The spacing of the linear DOF for the LOR mesh here is done in a GLL manner, the same as that of the higher-order element.

For the high-order discretisation $\mathbf{H}$, every internal DOF is coupled with every other DOF in an element. The conversion to the LOR space $\mathbf{H}_L$ increases the sparsity as now the DOF are only connected to their immediate neighbours. The difference in connectivity between the two discretisations is further accentuated in mathematical operators, even if the number of DOF is the same (as seen in Fig. 1). Thus, the operators on the lower-order discretisation are cheaper to evaluate than their higher-order counterparts. The generalised Vandemonde matrix $\mathcal{V}$ from Eqn. (10) is used for the interpolation of the higher-order polynomial space to the collocated Lagrange basis in the LOR space. In matrix terms, this interpolation is expressed as,

$$\mathcal{V}\hat{\mathbf{f}} = \mathbf{f} \tag{10}$$

where, $\mathbf{f}[i] = f(\xi_\mathbf{i})$ is the known polynomial expansion, $\mathcal{V}[i][j] = \phi_j(\xi_\mathbf{i})$ where ϕ_j is the chosen basis function and $\hat{\mathbf{f}}[j] = \hat{f}(\xi_\mathbf{j})$ are the unknown expansion coefficients. The unknown Lagrange polynomials are then found using $\hat{\mathbf{f}} = \mathcal{V}^{-1}\mathbf{f}$. The interpolation basis works for any basis selection in the HO space, where the matrix $\mathcal{V}$ is appropriately constructed according to the choice of higher-order expansion basis [1]. The interpolation for the case of the nodal expansion corresponds to a collocation projection. In contrast, a modal expansion basis is more involved as the construction of $\mathcal{V}$ has to handle the change of basis. The interpolation step is applied to both the trial and test functions in the higher-order polynomial space, and the interpolated functions are then used to construct $\mathbf{H}_L$.

Fig. 2 gives an overview of the Nektar++ LOR preconditioner algorithm. The input to the LOR preconditioner code is the residual of the iterative solution $\hat{\mathbf{r}}$ at a given iteration. Interpolation of

the solution from the HO discretisation to the LOR space is done using an appropriately constructed generalised Vandemonde matrix $\mathcal{V}$, depending on the choice of the higher-order expansion basis. Post the interpolation to the LOR space; the linear system is solved using an appropriately chosen solution strategy, referred to as the *inner-iteration strategy* from here on. The solution from the LOR mesh is then copied back to the HO space, which is then transformed back onto the original expansion by the inverse of the $\mathcal{V}$ matrix operation. Appropriate scatter-gather transformations are handled using the assembly maps $(\mathcal{A}_H, \mathcal{A}_L)$ in both spaces.

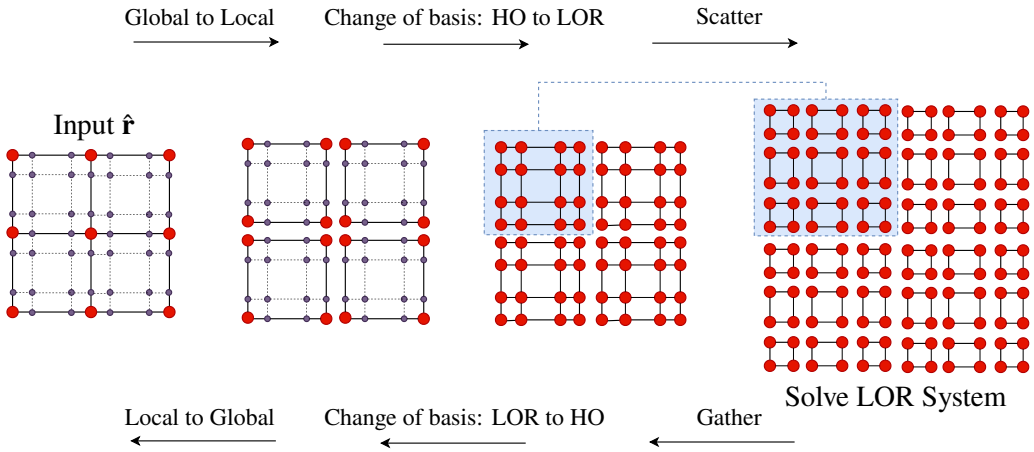
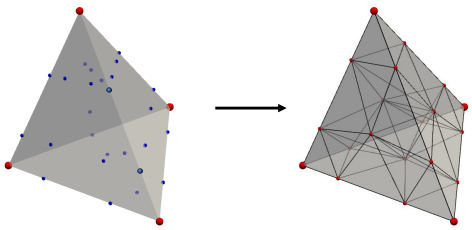


Figure 2: Representation of the current algorithm for the LOR preconditioner. There are three major steps: a) The conversion to the LOR mesh (left to right), b) solving the LOR linear system and c) projecting the solution back to the HO mesh (right to left). This process takes place at every preconditioned iteration.

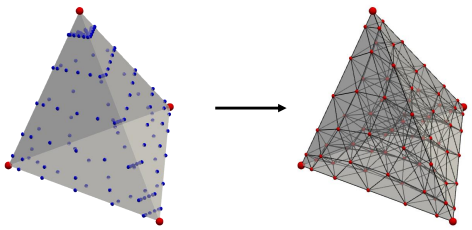
2.3.2 LOR space for simplices

The choice of discretisation in the LOR space affects the conditioning properties of the LOR preconditioner. The default point distribution in the LOR space is the GLL distribution, which is consistent with the spectral equivalence of the HO-LOR spaces. However, the LOR space for simplices like prisms and tetrahedrons requires extra considerations. For a nodal expansion in triangles and tetrahedrons, a closed-form expression for Lagrange polynomial does not exist like in the case of tensorial product of modal expansions for Gauss-Lobatto-Legendre (GLL) nodal points [1]. Thus, the formation of LOR space, which is a grid of collocated nodal points at $P = 1$, can't be created in a straightforward manner with either choice of higher-order expansion basis.

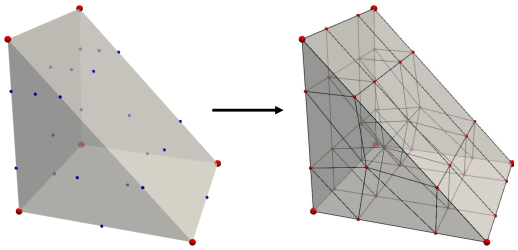
To solve this issue, Nektar++ originally uses a more easily defined polynomial expansion (ϕ_{ele}) through a different set of nodal points of the same dimension, spanning the same space, called the electrostatic points $(\xi_{\mathbf{ele}})$ instead of the chosen GLL nodal points $(\xi_{\mathbf{GLL}})$, based on the work of Hesthaven [25]. For the new set of nodal points, the interior nodal points are determined by the minimisation of electrostatic potential, and all the edge points are constrained to GLL quadrature points. The same approach is translated to the construction of LOR space. The visualisation of this construction is seen in Fig. 3. Fig. 3a and 3b show the LOR space for tetrahedrons at $P = 3$ and 6. Fig. 3c and 3d show the LOR space for prisms at $P = 3$ and 6, where a prismatic domain expansion is achieved by extending the triangular nodal expansion by making a tensor product with the Lagrange polynomial in the third direction. The reader is referred to Khurana et al. [24] for a detailed explanation of the construction of LOR space for triangles.



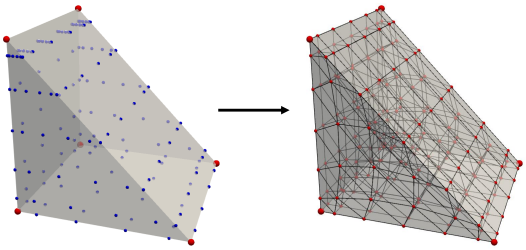
(a) Tetrahedron at $P = 3$



(b) Tetrahedron at $P = 6$



(c) Prism at $P = 3$



(d) Prism at $P = 6$

Figure 3: Visual representation of the creating the LOR space for simplices at $P = 3$ and 6. Within each subfigure, the figure on the right shows the LOR space for a HO element on the left. The LOR space is viewed as a collocated Lagrange basis at P=1 indicated by red dots, with on $\xi_{\mathbf{ele}}$ points in the interior and $\xi_{\mathbf{GLL}}$ points constrained the boundary. One can identify this by looking at any of the triangular faces for the simplices.

3 Results

An efficient solution of the pressure Poisson in Eqn. (5) is the primary application of interest for improving the performance of the IncNS solver. Thus, the Poisson equation (Eqn. 11) is the primary test problem for this study

$$\nabla^2 u(\mathbf{x}) = f(\mathbf{x}) \tag{11}$$

A simple, known forcing function in Eqn. (12) is used for all test cases in further sections, with either a Dirichlet boundary condition from Eqn. (13) or a zero Neumann boundary condition on the boundaries unless specified otherwise.

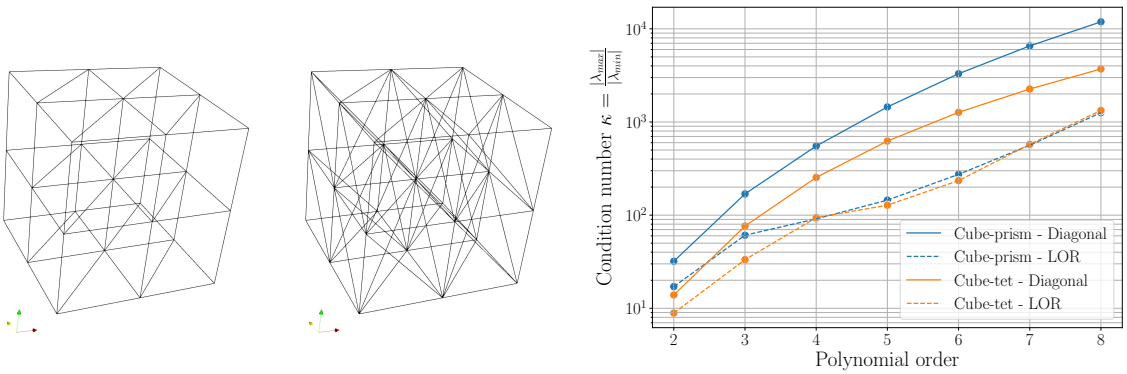
$$\text{Forcing: } f(\mathbf{x}) = -d \prod_{i=1}^{d} \sin(x_i) \tag{12}$$

$$\text{Dirichlet BCs: } u_{\partial\Omega} = \prod_{i=1}^{d} \sin(x_i) \tag{13}$$

To understand the conditioning properties of the preconditioning strategies in this section, an *iterative condition number* (κ) is used, which is evaluated for the action of the preconditioning matrix $\mathbf{M}$ or $\kappa(\mathbf{M}^{-1}\mathbf{H})$. The κ is estimated by the ratio of magnitudes of the max and min eigenvalues λ of the system, i.e., $\frac{|\lambda_{max}|}{|\lambda_{min}|}$. A standard eigensolver based on the QR-method [26] has been used out of the box to evaluate the eigenvalues. The iterative solver settings used in the upcoming sections are detailed in the Appendix.

3.1 Conditioning properties

A unit cube in 3D as seen in Fig. 4, discretised using tetrahedrons and prisms, is the first test case of this study. The LOR preconditioner is evaluated on meshes containing only prisms (Fig. 4a) and only tetrahedrons (Fig. 4b) to understand the conditioning properties for these element types. The Poisson problem is solved with the forcing function in Eqn. (12) for $d = 3$ and Dirichlet BCs from Eqn. (13) on all the domain boundaries. A modal HO basis function is chosen, described in Section 2.1.



(a) Cube-prism mesh (b) Cube-tet mesh (c) κ with P for different preconditioners

Figure 4: Comparison of the condition number (κ) on applying the LOR preconditioner for the cube case with Dirichlet BCs on all boundaries. Two different meshes are investigated to see the effect of different types of higher-order elements.

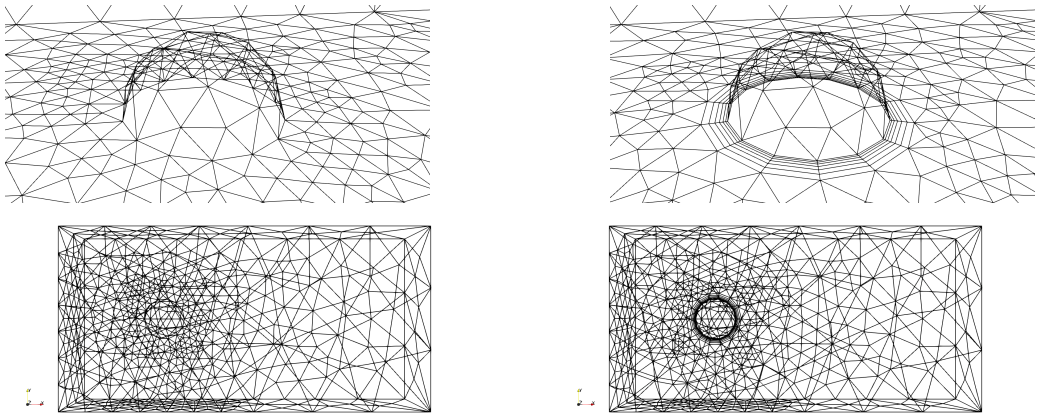
Fig. 4c compares the condition number for a prism mesh (Cube-prism) to the tetrahedron mesh (Cube-tet) for increasing polynomial order. The conditioning of the LOR preconditioner is better than the Diagonal; however, it still shows an increasing trend, especially at higher polynomial orders. However, for the polynomial range of $P \leq 6$, the condition number is controlled, and the hope is that this will be sufficient to improve the conditioning of larger 3D meshes.

3.2 Iterative performance

To further evaluate the iterative properties of the LOR preconditioner, the performance of the preconditioned GMRES solution is evaluated here for a flow past a 3D Hemisphere case seen in Fig. 5. The Poisson problem is solved in the domain with either Dirichlet BCs from Eqn.(13) on all boundaries or zero Neumann BCs on all boundaries but the outlet, which is Dirichlet from Eqn.(13).

Two different meshes are considered for this geometry - one containing only tetrahedron elements (Fig. 5a) and the second mesh contains six additional prism layers on the surface of the hemisphere (Fig. 5b). The second mesh is a step towards meshes typical of aerodynamics simulations, where boundary layers are added to better resolve the near-wall flow.

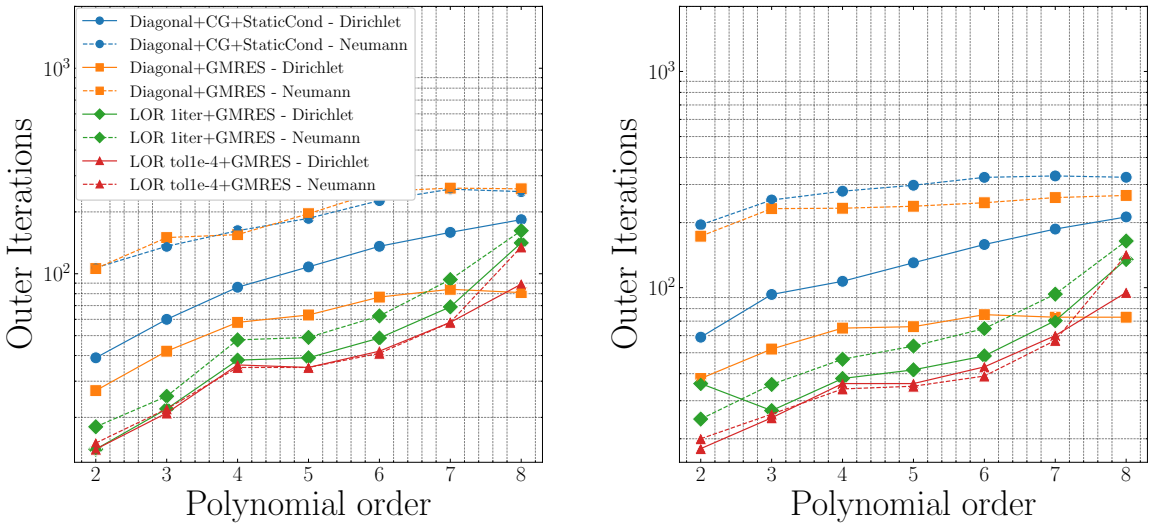
Fig. 6 shows the results for the outer iterations needed to solve the Poisson equation for the 3D hemisphere case with increasing polynomial order. A comparison is made with the default iterative solver configuration for the pressure system in the IncNS solver for large 3D cases, also referred to as "Baseline", moving forward. The default setup applies a static condensation to the system matrix [1] and uses a conjugate gradient solver with the Diagonal preconditioner (Diagonal+CG+StaticCond in Fig. 6).



(a) Only tetrahedrons (5272 elements)

(b) Tetrahedrons and 6 prism layers (5076 elements)

Figure 5: Meshes for the 3D Hemisphere case. The meshes have been generated using the MCF file capability in the mesh generation software *NekMesh* [27], with the same settings across the two meshes. The top row shows the isometric view of the surface mesh inside the domain, and the bottom row shows the bottom view of the domain.



(a) Hemisphere - only tetrahedrons

(b) Hemisphere - tetrahedrons and prisms

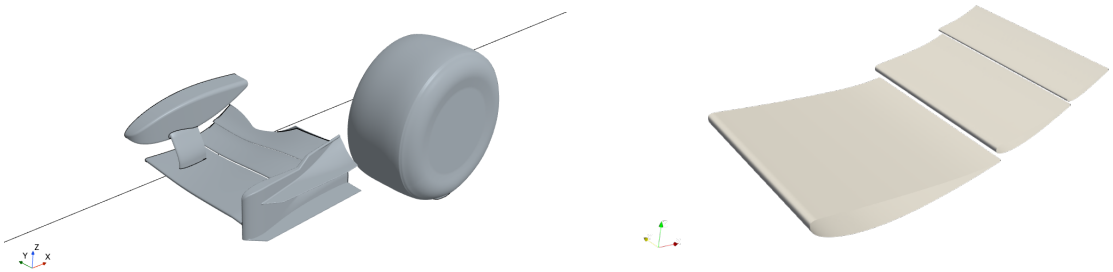
Figure 6: Outer iterations to solve the Poisson equation for the 3D Hemisphere case - polynomial order sweep. The iteration results are an average of three runs on 10 MPI ranks on a single computational node. StaticCond = Static Condensation. LOR/Diagonal: Preconditioner. CG/GMRES: Iterative solver. Dirichlet: Dirichlet BC on all boundaries. Neumann: Zero Neumann BCs on all domain boundaries apart from the outlet, which is Dirichlet. 1iter = 1 V-cycle of AMG for LOR solution. tol1e-4 = V-cycles are applied until relative tolerance drop of 10^{-4}.

The iterative performance of the LOR preconditioner is consistently best across the two meshes and boundary condition choices for the polynomial range of $P \leq 6$. For higher polynomial orders, the Diagonal preconditioner with GMRES performs better than LOR only for the Dirichlet BCs. However, this is not too concerning as the Neumann case is more similar to the BCs applied for the pressure system for an incompressible flow simulation. For complex meshes, such a high polynomial order is rarely used. Interestingly, LOR always performs better than the Baseline configuration for the pressure system.

Two LOR inner iteration strategies are also tested, where at every inner solver iteration, either 1 AMG V-cycle is applied (*1iter*), or V-cycles are applied until a drop of 10^{-4} is seen in the residual (*tol1e-4*). The *tol1e-4* approach shows a lower outer iteration count as the inner solution is solved to a lower residual due to the application of multiple V-cycles. The *1iter* approach also gave variable iteration results across the three runs due to the inaccuracies in solving the LOR space problem. However, the *1iter* approach is not considerably far off iterations-wise from the *tol1e-4*.

3.3 Application on industrial test cases

The Imperial Front Wing (IFW) is a publicly available industrial benchmark [28]. IFW is a multi-element wing operating in ground effect, generating a complex system of interacting vortices downstream, which is a good stress-test case for the IncNS solver. The industrial test cases considered for the current study are: a) IFW with a wheel downstream (IFW-W, Fig. 7a), and b) a slice of the IFW at a spanwise section 250mm from the symmetry plane extruded 50mm in width (eIFW, Fig. 7b). The higher-order mesh for both cases is a mixture of tetrahedrons and prisms. The performance of the LOR preconditioner is compared with the Baseline setup for large 3D simulations in the IncNS solver. The tests run in this section use the HX1 cluster from Imperial College Research Computing Service. Each compute node in the cluster contains Lenovo SD630v2 servers, each with 2 x Intel Xeon Platinum 8358 (Ice Lake) 2.60GHz 32-core processors; resulting in 64 cores per node and 512 GB RAM per node.



(a) IFW with Wheel (2.87×10^6 elements)

(b) extruded IFW slice (4.22×10^5 elements)

Figure 7: Geometry for the test cases in this study. The higher-order meshes are a mixture of tetrahedrons and prisms, with eight boundary layers along the airfoil elements and five boundary layers on the floor profile containing prismatic elements. The rest of the domain is filled with tetrahedron elements.

3.3.1 Poisson Solve performance

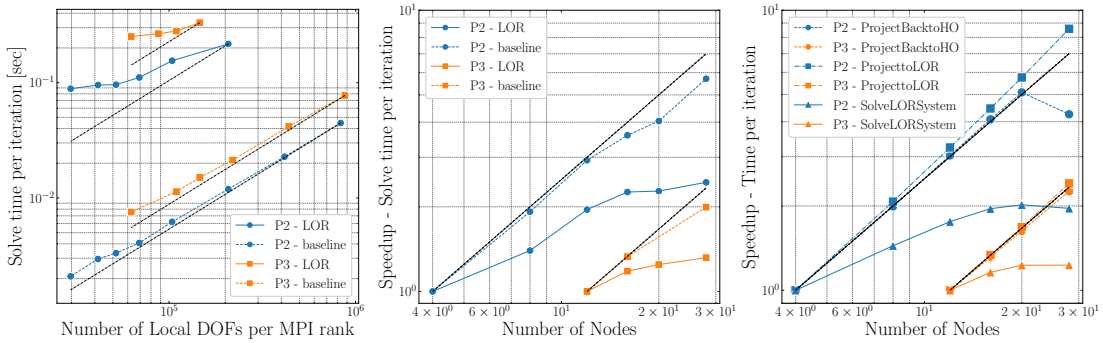
The LOR preconditioner's iterative performance is tested on the IFW-W case. The Poisson problem is solved in the domain with zero Neumann BCs on all boundaries but the outlet, which is specified as zero Dirichlet. Table 1 compares the iterations and timings information of the Poisson solve for two polynomial orders. LOR outperforms the Baseline setup for a problem of this size both for the iterations needed and time to solve. The time to solve for the LOR preconditioner scales up approximately as the local degrees of freedom on increasing the polynomial order. The cost of a LOR iteration (time to solve per iteration) is much higher (about 20x) than that of the Baseline setup. However, the conditioning properties of LOR control the number of iterations well within this margin.

Polynomial order (P)	Degrees of Freedom ($\times 10^6$)		Outer iterations to 1e-4 tolerance		Time to Solve [sec]		Time to Solve per iteration [sec]	
	Global	Local	LOR	Baseline	LOR	Baseline	LOR	Baseline
2	9.59	53.1	67	3504	6.578	39.097	0.0981	0.0046
3	32.2	112.1	124	4142	16.313	62.414	0.3178	0.0151

Table 1: Comparison of LOR and Baseline configurations for solving the Poisson equation for the IFW with Wheel. The tests are run on 12 computational nodes on the HX1 cluster.

A strong scaling study is further conducted to understand the scaling performance of the LOR preconditioner. Fig. 8 shows solve time per iteration costs and compares LOR vs Baseline configurations. There is an evident loss in scaling seen in Fig. 8a and 8b for the time to solve the Poisson equation, as the deviation from ideal scaling is much larger for LOR compared to the Baseline. To attempt to understand this loss of scaling, the three major steps in the LOR preconditioner, as explained previously in Fig. 2, are timed. Fig. 8c indicates that most of the loss in scaling comes from the SolveLORSystem routine, which uses the AMG routines to solve the LOR space. The solution transfer to and from the HO-LOR spaces scales up ideally.

Furthermore, for a given polynomial order, Fig. 8a has some missing points towards higher local DOF per rank for the LOR compared to the baseline. This is due to the memory footprint of the LOR being much larger than the Baseline setup, which does not allow the simulation to run when few computational resources are available. The LOR preconditioner requires the generation of the LOR space, which becomes a memory-intensive task in the current implementation for a complex geometry simulation.



(a) Time to solve: Poisson equation (b) Speedup: Time to solve (c) Speedup: LOR subroutines

Figure 8: Strong scaling study - Time to solve for Poisson equation for the IFW with Wheel. The black dotted line indicates ideal scaling. The tests are run on 1, 2, 4, 8, 12, 16, 20, and 28 computational nodes on HX1 cluster.

3.3.2 Performance within the IncNS solver

This section evaluates the performance of the LOR preconditioner for the pressure system within the IncNS solver of Nektar++. The tests are done for an unsteady flow at a $Re = 2 \times 10^5$ around the eIFW test case. The tests done here are done by varying the configuration of the pressure system only. The setup of the simulations in this section is detailed in the Appendix.

The simulation is run at a polynomial order of *P=3* for pressure for 0.5 seconds, or two convective time units (CTU) at a constant timestep of 10^{-5} seconds. Table 2 shows the average iteration numbers and timing information from CTU = 1 to CTU = 2. The LOR provides an improvement of 31% over the Baseline setup for the pressure solution. The average outer iterations needed for the pressure solution are

9 GMRES iterations for LOR compared to 518 for Baseline, albeit cheaper, conjugate gradient iterations. The velocity system is unaffected by the changes in the pressure system setup, as the average iterations remain the same.

	Avg time per Timestep [sec]	Time to CTU [hrs]	Avg. Pressure iters	Avg. Velocity iters (U,V,W)
LOR	2.745	19.06	9	(45,38,44)
Baseline	3.979	27.63	518	(45,38,44)

Table 2: Timings and Iterative performance for the extruded IFW case with the IncNS solver. The numbers are averaged from CTU = 1 to CTU = 2, resulting in 25000 solver steps. The simulations are run on 4 compute nodes.

Fig. 9 shows the evolution of various properties for the simulation of the LOR vs Baseline setup for the pressure system. The pressure iterations differ by one order of magnitude, and LOR converges to approximately 10 iterations. The variance in pressure iterations is also lower for the LOR, which testifies to the conditioning properties of this preconditioner. The evolution of force coefficients and Courant-Friedrichs-Lewy (CFL) number is similar for the two setups, validating the LOR preconditioner implementation.

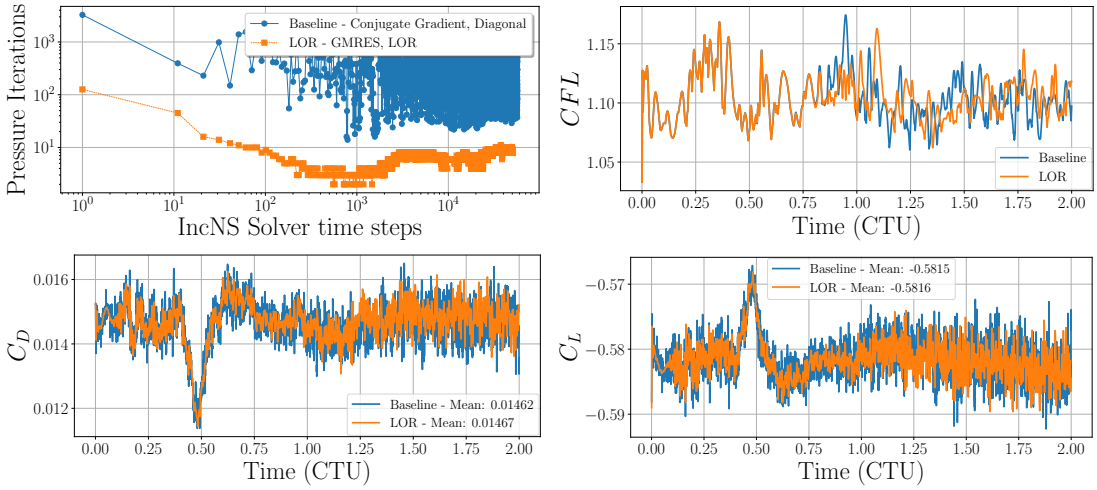


Figure 9: Time stepping evolution of parameters for the extruded IFW case with the IncNS solver.

4 Conclusion and Future work

The current study presents the LOR preconditioner for mixed-element 3D mesh configurations containing simplex elements within the incompressible Navier-Stokes (IncNS) equations solver of Nektar++. LOR preconditioner provides improved conditioning properties over the existing preconditioners for the pressure Poisson system, which translates to better iterative performance. These advantages are seen upon testing the LOR preconditioner for selected industrial geometries based on the Imperial Front Wing. However, two issues in the current implementation are also exposed: scalability and memory footprint, which limit the application envelope of the LOR preconditioner. Nevertheless, using the LOR preconditioner within the IncNS solver provides substantial computational gains for incompressible flow simulations. Future work for this preconditioner involves improving the scalability of the LOR algorithm by focusing on the solution strategy of the LOR space. Efforts will also be undertaken to improve memory consumption by debugging mesh graph routines and looking into matrix-free approaches.

Acknowledgement

This project received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sk lodowska-Curie grant agreement No 955923.

The authors acknowledge computational resources and support provided by the Imperial College Research Computing Service (`http://doi.org/10.14469/hpc/2232`).

Appendix

Iterative solver settings: All the tests in this study which use an iterative solver have been run until a convergence tolerance 10^{-4} is reached. The solver used most frequently is GMRES, with a restart after the GMRES projection reaches 100 vectors.

- *BoomerAMG* from Hypre [29] is used via its interface from PETSc [30] for solving the LOR space. Solver settings for LOR inner iteration strategy:
  - 1 V-cycle for AMG, or until a tolerance, if specified
  - HMIS coarsening
  - Strong threshold = 0.70 for 3D cases
  - Coarse grid size = 10 to control the levels of AMG cycle
  - Extended+i interpolation with a Pmax = 2 and truncation of 0.3
  - SOR/Jacobi smoothener with two sweeps going up and down

IncNS solver simulation setup: The characteristic length of the Imperial Front Wing geometry and its variants is described by the chord length of the main element and is equal to 0.25 m. The computational domain emulates the extruded IFW inside a wind tunnel. The freestream air enters the domain and is enforced as a uniform Dirichlet boundary condition at the inlet. The ceiling and the side walls are treated as slip walls. A high-order stable form of zero Neumann boundary condition at the outlet is used for velocity, and the pressure is fixed to the ambient pressure.

- Solver settings:
  - Taylor-Hood approximation: The polynomial order for the velocity is $P+1$, pressure is P.
  - A second-order time-integration scheme with a constant time-step of 10^{-5} sec.
  - Initialisation from a velocity field obtained from a RANS solution, and the pressure is 0.
  - Linear solvers use an absolute tolerance strategy with a convergence tolerance 10^{-4}.
  - Stabilisation: Spectral Vanishing Viscosity DG Kernel [31]

References

[1] George Karniadakis and Spencer Sherwin. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press on Demand, 2005.

[2] Filipe F. Buscariolo, Julien Hoessler, David Moxey, Ayad Jassim, Kevin Gouder, Jeremy Basley, Yushi Murai, Gustavo R.S. Assi, and Spencer J. Sherwin. Spectral/hp element simulation of flow past a formula one front wing: Validation against experiments. *Journal of Wind Engineering and Industrial Aerodynamics*, 221, 2 2022. ISSN 01676105. doi: 10.1016/j.jweia.2021.104832.

[3] James Slaughter, David Moxey, and Spencer Sherwin. Large eddy simulation of an inverted multi-element wing in ground effect. 2022. doi: 10.21203/rs.3.rs-2068909/v1. URL https://doi.org/10.21203/rs.3.rs-2068909/v1.

[4] C. D. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo, D. De Grazia, S. Yakovlev, J. E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R. M. Kirby, and S. J. Sherwin. Nektar++: An open-source spectral/hp element framework. *Computer Physics Communications*, 192:205–219, 7 2015. ISSN 00104655. doi: 10.1016/j.cpc.2015.02.008.

[5] George Em Karniadakis, Moshe Israeli, and Steven A Orszag. High-order splitting methods for the incompressible navier-stokes equations. *Journal of Computational Physics*, 97(2):414–443, 1991. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(91)90007-8. URL https://www.sciencedirect.com/science/article/pii/0021999191900078.

[6] Claudio Canuto, Alfio Quarteroni, M. Yousuff Hussaini, and Thomas A. Zang. *Spectral Methods*. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-30727-3. doi: 10.1007/978-3-540-30728-0. URL http://link.springer.com/10.1007/978-3-540-30728-0.

[7] Steven A Orszag. Spectral methods for problems in complex geometrics. In *Numerical methods for partial differential equations*, pages 273–305. Elsevier, 1979.

[8] Luke Olson. Algebraic multigrid preconditioning of high-order spectral elements for elliptic problems on a simplicial mesh. volume 29, pages 2189–2209. Society for Industrial and Applied Mathematics Publications, 2007. doi: 10.1137/060663465.

[9] Pedro D. Bello-Maldonado and Paul F. Fischer. Scalable low-order finite element preconditioners for high-order spectral element poisson solvers. *SIAM Journal on Scientific Computing*, 41:S2–S18, 1 2019. ISSN 1064-8275. doi: 10.1137/18M1194997. URL https://epubs.siam.org/doi/10.1137/18M1194997.

[10] Claudio Canuto, Paola Gervasio, and Alfio Quarteroni. Finite-element preconditioning of g-ni spectral methods. *SIAM Journal on Scientific Computing*, 31:4422–4451, 2009. ISSN 10957200. doi: 10.1137/090746367.

[11] Michel Deville and Ernest Mund. Chebyshev pseudospectral solution of second-order elliptic equations with finite element preconditioning. *Journal of Computational Physics*, 60(3):517–533, 1985.

[12] Michel O Deville and Ernest H Mund. Finite-element preconditioning for pseudospectral solutions of elliptic problems. *SIAM Journal on Scientific and Statistical Computing*, 11(2):311–342, 1990.

[13] Michel O Deville and Ernest H Mund. Fourier analysis of finite element preconditioned collocation schemes. *SIAM journal on scientific and statistical computing*, 13(2):596–610, 1992.

[14] Will Pazner. Efficient low-order refined preconditioners for high-order matrix-free continuous and discontinuous galerkin methods. *SIAM Journal on Scientific Computing*, 42:A3055–A3083, 10 2020. ISSN 10957197. doi: 10.1137/19M1282052.

[15] Claudio Canuto and Alfio Quarteroni. Preconditioned minimal residual methods for chebyshev spectral calculations. *Journal of Computational Physics*, 60(2):315–337, 1985. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(85)90010-5. URL https://www.sciencedirect.com/science/article/pii/0021999185900105.

[16] Sang Dong Kim and Seymour V. Parter. Preconditioning chebyshev spectral collocation method for elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 33(6):2375–2400, 1996. doi: 10.1137/S0036142994275998. URL https://doi.org/10.1137/S0036142994275998.

[17] Sang Dong Kim and Seymour V. Parter. Preconditioning chebyshev spectral collocation by finite-difference operators. *SIAM Journal on Numerical Analysis*, 34(3):939–958, 1997. doi: 10.1137/S0036142995285034. URL https://doi.org/10.1137/S0036142995285034.

[18] Will Pazner, Tzanio Kolev, and Clark Dohrmann. Low-order preconditioning for the high-order finite element de rham complex. 3 2022. URL http://arxiv.org/abs/2203.02465.

[19] Malachi Phillips, Stefan Kerkemeier, and Paul Fischer. Tuning spectral element preconditioners for parallel scalability on gpus. 10 2021. URL http://arxiv.org/abs/2110.07663.

[20] Will Pazner, Tzanio Kolev, and Jean-Sylvain Camier. End-to-end gpu acceleration of low-order-refined preconditioning for high-order finite element discretizations. 10 2022. URL http://arxiv.org/abs/2210.12253.

[21] Joachim Schöberl, Jens M Melenk, Clemens Pechstein, and Sabine Zaglmayr. Additive schwarz preconditioning for p-version triangular and tetrahedral finite elements. *IMA Journal of Numerical Analysis*, 28(1):1–24, 2008.

[22] Noel Chalmers and T. Warburton. Low-order preconditioning of high-order triangular finite elements. *SIAM Journal on Scientific Computing*, 40:A4040–A4059, 2018. ISSN 10957197. doi: 10.1137/17M1149444.

[23] T. Warburton, L.F. Pavarino, and J.S. Hesthaven. A pseudo-spectral scheme for the incompressible navier–stokes equations using unstructured nodal elements. *Journal of Computational Physics*, 164(1):1–21, 2000. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.2000.6587. URL https://www.sciencedirect.com/science/article/pii/S0021999100965872.

[24] Parv Khurana, Spencer J. Sherwin, Julien Hoessler, Francesco Bottone, and David Moxey. Comparison of preconditioning techniques using spectral/hp element methods for complex geometries. In *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2023: Selected Papers from the ICOSAHOM Conference, Yonsei University, Seoul, Korea, August 14-18, 2023*. Springer International Publishing, 2024. Accepted.

[25] J. S. Hesthaven. From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex. *SIAM Journal on Numerical Analysis*, 35(2):655–676, 1998. doi: 10.1137/S003614299630587X. URL https://doi.org/10.1137/S003614299630587X.

[26] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999. ISBN 0-89871-447-8 (paperback).

[27] M.D. Green, K.S. Kirilov, M. Turner, J. Marcon, J. Eichstädt, E. Laughton, C.D. Cantwell, S.J. Sherwin, J. Peiró, and D. Moxey. Nekmesh: An open-source high-order mesh generation framework. *Computer Physics Communications*, 298:109089, 2024. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2024.109089. URL https://www.sciencedirect.com/science/article/pii/S0010465524000122.

[28] Jonathan Mark Pegrum. Experimental study of the vortex system generated by a formula 1 front wing, 2006.

[29] Ulrike Meier Yang et al. Boomeramg: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(1):155–177, 2002.

[30] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil M. Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, Jacob Faibussowitsch, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc Web page. https://petsc.org/, 2023. URL https://petsc.org/.

[31] Robert M Kirby and Spencer J Sherwin. Stabilisation of spectral/hp element methods through spectral vanishing viscosity: Application to fluid mechanics modelling. *Computer methods in applied mechanics and engineering*, 195(23):3128–3144, 2006.