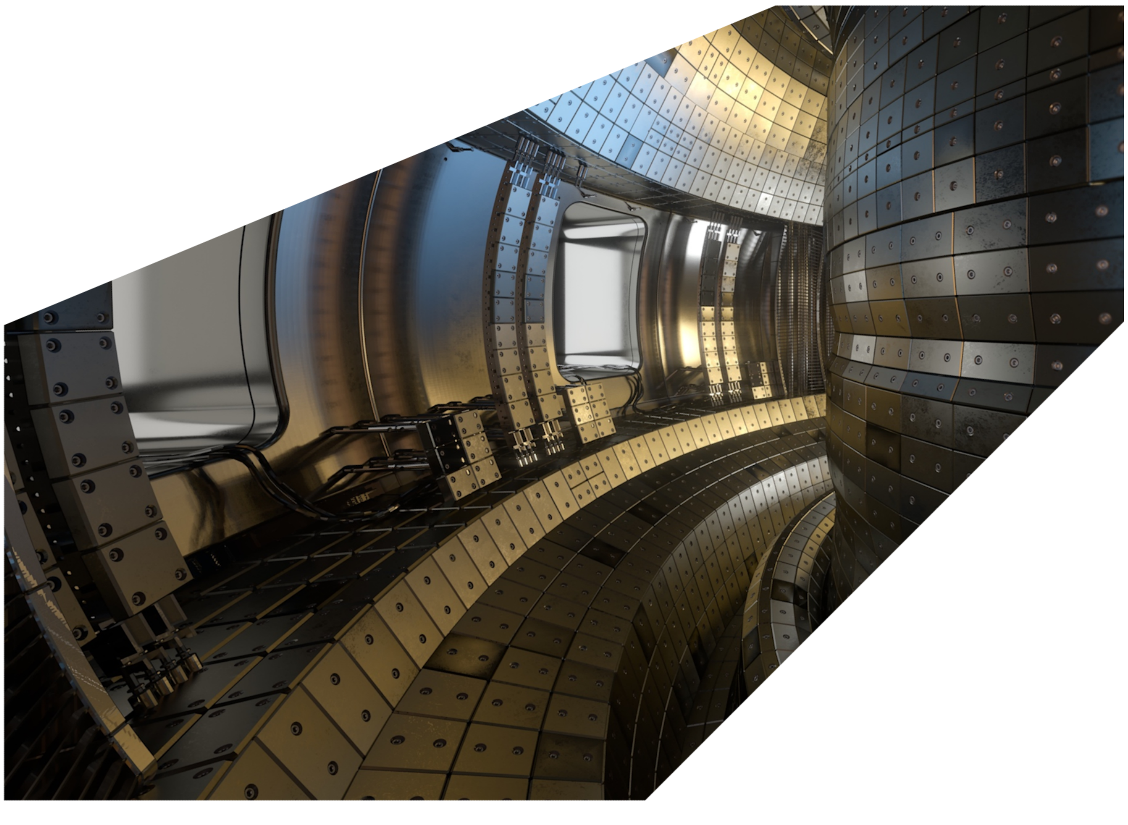
UK Atomic Energy Authority

ExCALIBUR

Finite Element Models Complementary Actions: Code Integration, Acceptance and Operation 1.

M6c.2 Version 1.00

Abstract
The report describes work for ExCALIBUR project NEPTUNE at Milestone 6c.2. It provides material on finite element exterior calculus, techniques which are seen as effectively indispensible in modern numerical models involving coupling of particles to electromagnetic fields viz. particle-in-cell simulations. A small collection of illustrative examples, taken from a key textbook ([1]) in the field, is exhibited, with reference to implementations in the *Firedrake* PDE solver which are available via a public GitHub repository. Also included is a brief overview of recent developments within the *Nektar++* solver framework intended to produce a suitable *Nektar++*-based library for the handling of continuum field aspects of NEPTUNE. This material covers the removal of the need for an XML session file for each simulation, in-progress modifications needed to harmonize the library with a PIC implementation, and also recent work by the holder of the grant T/NA078/20.



UKAEA REFERENCE AND APPROVAL SHEET

	Client Reference:	
	UKAEA Reference:	CD/EXCALIBUR-FMS/0069
	Issue:	1.00
	Date:	27 September 2022
Project Name: ExCALIBUR Fusion Modelling System		

	Name and Department	Signature	Date
Prepared By:	Ed Threlfall Owen Parry Will Saunders BD	N/A N/A N/A	27 September 2022 27 September 2022 27 September 2022
Reviewed By:	James Cook BD	JWSCook	27 September 2022

1 Introduction

The finite element method for solving PDEs is mature and encompasses many different variants, for example there is a wide choice of basis functions and assignments of degrees of freedom, so apparently a selection of methods capable of solving the same problem (see, for example, the table of over fifty finite element types at [2]). In recent years, works such as [3] have provided a unifying theory of the choice of finite element method that is appropriate, or most appropriate, to a particular type of problem. This framework, going under the designation of finite element exterior calculus (hereafter FEEC), has clarified the role of hitherto-discovered discretizations and also has led to the discovery of novel element types; a pleasing distillation of these ideas is the 'periodic table' of the finite elements [4]. Another favourable aspect is that most of the constituent element types have known order-p implementations and thus the path to exponential (spectral) convergence, at least for smooth solutions, is straightforward.

These developments, which have been led by mathematicians and numerical analysts, stem from a deep understanding of the underlying structure of both the continuum PDEs and their discrete form. In particular, the geometrical nature of the dependent variable comes to the fore, with the equations themselves being expressed in the language of exterior calculus, viz. differential forms and the exterior derivative. These structures encode important aspects of the various problems, such as topological invariants and conserved quantities, and, with the correct choice of discretization, these properties are also manifest in the discretized system - for this reason, FEEC is referred to as a 'structure-preserving', 'mimetic', or 'compatible', method. Note that the exterior calculus encompasses the more familiar 'fields' of physics, e.g. a 0-form is simply a scalar field, a 1-form is a (covariant) vector field, and a 2-form is an antisymmetric rank-2 covariant tensor of which a good example is the Maxwell tensor of classical electrodynamics; higher-k forms correspond to fully-antisymmetric tensors of higher rank.

One small cautionary note is that other, non-finite element, structure-preserving discretizations exist, going under the wider banner of discrete exterior calculus (hereafter DEC). These may not have elements satisfying all of the requirements of the Ciarlet definition ([5]) and they usually involve a pair of spatial grids - the primal and the Voronoi dual. The prototypical example is the Yee algorithm widely used in computational electromagnetics, in which the electric degrees of freedom are associated to one spatial grid and the magnetic degrees of freedom the other.

To attempt an overview (which incorporates advice given in [6]), there are two main reasons to use FEEC (or DEC): 1) they give naturally stable discretizations which do not need artificial numerical dissipation (e.g. upwinding or least-squares) to remain stable; 2) they enable the construction of discretizations with conserved quantities. In particular, FEEC / DEC has been used extensively to provide stable discretizations for Maxwell's equations, and structure-preserving methods for electromagnetism coupled to particle methods have been shown to produce effective solvers for the Vlasov-Poisson system. For NEPTUNE it is intended that structure-preserving methods are incorporated into the *Nektar++* PDE solver framework and this is currently under consideration by the main developers of that code in conjunction with UKAEA NEPTUNE team members. The integration of the families of finite elements used by FEEC into the *Nektar++* framework in terms of code structure and optimizations for efficient calculation is an area for further research, though this would build on an existing body of knowledge (see e.g. [7] for efficient implementations of H(div)- and H(curl)-conforming elements, and [8] for an application of the sum-factorization optimization technique to these element types). Some instructive examples of the utility of FEEC methods are presented in Section 2 as a prelude to the implementation of similar techniques in NEPTUNE software.

The spectral element framework *Nektar++* is currently being modified such that it will form a library supplying continuum field capabilities to higher-level NEPTUNE applications. The main work in this direction is summarized in Section 3, and includes the removal of the need for an XML session file, modifications to interface *Nektar++* with the UKAEA in-house PIC code NESO-Particles, preliminary implementation of Nvidia GPU kernels, modifications to the *NekMesh* mesh generator aimed at producing meshes suited to fusion-relevant scenarios, and a comment on the amount of memory used by *Nektar++*.

2 Finite element exterior calculus

Taken from [1] and associated papers, the examples presented here, which illustrate problems where conventional FEM may perform poorly, focus on the Hodge Laplacian - a generalization of the familiar ∇^2 operator to include action on differential forms of order $k = 0, 1, ..., D$ (in D space dimensions). The $k = 1$ Hodge Laplacian is the familiar vector Laplacian and study of this case is motivated by the appearance of this operator in the Navier-Stokes equations (equivalently, the vector vorticity equation), the Maxwell equations in the frequency domain (which can give eigenproblems of the type examined in Subsection 2.4), and the vector induction equation.

The exterior derivative operator d acts to convert differential forms between the space V^k of k-forms and that V^{k+1} of $(k + 1)$-forms; the formal adjoint of d, denoted d^*, is given by $\delta \equiv \pm * d *$, where $*$ is the Hodge star, and called the co-differential. The sign is fixed by the adjoint property (see [1]); for this operator in two dimensions and acting on 0- and 1-forms, the negative sign is appropriate. The Hodge Laplacian is then the unbounded operator from $V^k \to V^k$ given by $L^k = d\delta + \delta d$.

The Hodge Laplace problem

$$L^k u = f - P_{\mathfrak{h}} f, \quad u \perp \mathfrak{H}^k \tag{1}$$

has a unique solution (in the appropriate space; see [1]) for any $f \in W^k$. The possibility of a null space (denoted $\mathfrak{H}^k$) for L^k (i.e. harmonics; see Subsection 2.4) is the reason for the condition $u \perp \mathfrak{H}^k$ and the subtraction of the projection $P_{\mathfrak{h}} f$ of f onto this space.

Two weak formulations (used because they are numerically convenient and allow existence proofs of well-behaved solutions) of this problem are the *primal weak formulation*

$$\langle du, dv\rangle + \langle d^* u, d^* v\rangle = \langle f - P_{\mathfrak{H}} f, v\rangle \tag{2}$$

where u is a trial function and v is a test function (in a Galerkin approximation, taken from the same finite-dimensional function spaces), and the *mixed weak formulation*, in which $\sigma \equiv d^* u$ (note the adjoint) and $p = P_{\mathfrak{H}} f$

$$\begin{aligned} \langle \sigma, \tau \rangle - \langle u, d\tau \rangle &= 0, \\ \langle d\sigma, v \rangle + \langle du, dv \rangle + \langle p, v \rangle &= \langle f, v \rangle, \\ \langle u, q \rangle &= 0. \end{aligned} \tag{3}$$

According to [1], there are four possible ‘good’ choices of (pairs of) discrete spaces for this problem (though some may in practice be equivalent), which are (V_h^k denotes the function space used to discretize the space of k-forms; $\mathcal{T}_h$ represents a simplicial triangulation of the domain; $\mathcal{P}_r\Lambda^k$ denotes the complete space of piecewise polynomial k-forms of polynomial order at most r over the domain and $\mathcal{P}_r^-\Lambda^k$ denotes a similar but slightly smaller space of polynomial k-forms, called the *trimmed* space):

$$V_h^{k-1} = \begin{cases} \mathcal{P}_r\Lambda^{k-1}(\mathcal{T}_h), \\ \quad \text{or} \\ \mathcal{P}_r^-\Lambda^{k-1}(\mathcal{T}_h) \end{cases}$$

and

$$V_h^{k} = \begin{cases} \mathcal{P}_r^-\Lambda^{k}(\mathcal{T}_h), \\ \quad \text{or} \\ \mathcal{P}_{r-1}\Lambda^{k}(\mathcal{T}_h) \;\; (\text{if } r > 1). \end{cases}$$

This is the notation of the ‘periodic table’ of the finite elements [4]. Elements of these types are implemented in the *Firedrake* PDE solver framework ([9], and see Appendix A) and so it is straightforward to explore these methods using that code. It is worth a brief note to say that *Nektar++* has a narrower range of element types and is currently restricted to ‘scalar’ elements (the framework currently treats vector quantities on a scalar-per-component basis). Changes to this situation are currently being considered by the *Nektar++* development team.

2.1 Simple one-dimensional example of an unstable discretization

Section 1.1 of [3] contains a particularly straightforward demonstration of the need for a numerically-stable method in the case of a very simple ODE. A yet more straightforward example is found in the reference in the same paper to work by Brezzi and Bathe ([10], of which p.49 in particular), in which the ODE is

$$-\frac{d^2u}{dx^2} = f \tag{4}$$

on the interval $[-1:1]$ with homogeneous Dirichlet conditions at the endpoints, with forcing term $f = 1$; the solution is trivially $u = \frac{1}{2}(1 - x^2)$.

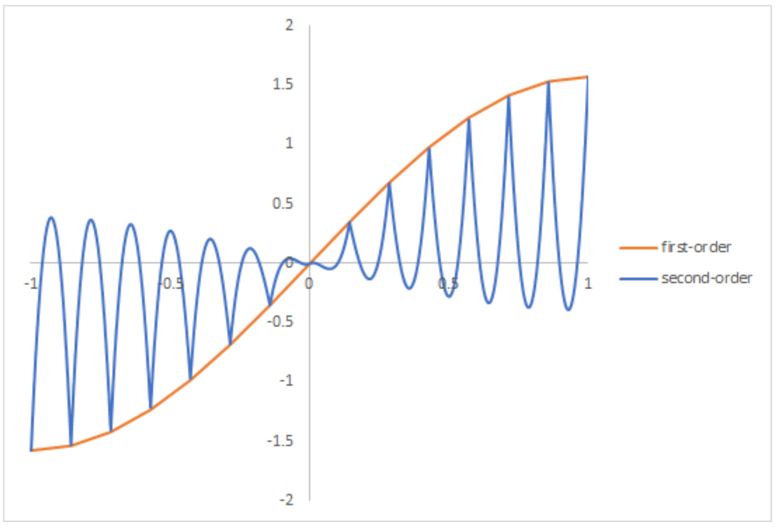


Figure 1: Plots of the derivative for the example from [3] showing sensible output for a linear discretization of the derivative and clear nonsense in the quadratic case.

A mixed formulation is used with the auxiliary variable $\sigma \equiv -u'$, giving the weak form obeyed by the discrete approximation σ_h and u_h:

$$\int_{-1}^{1} \sigma_h \tau \; dx \quad = \quad \int_{-1}^{1} u_h \tau' \; dx, \tag{5}$$

$$\int_{-1}^{1} \sigma_h' v \; dx \quad = \quad \int_{-1}^{1} v \; dx. \tag{6}$$

Consider a discretization into N equal finite elements with u represented by (discontinuous) piecewise constants and σ by continuous piecewise quadratics. It is not difficult to show that the L^2-norm of the error $\sigma_h - \sigma$ is

$$||\sigma_h - \sigma||_0^2 = \frac{5}{9}\left(4 - \frac{1}{N^2}\right). \tag{7}$$

The point here is that this error tends to a constant as $N \to \infty$ i.e. the numerical solution *never* converges to the analytic one.

This example corresponds to the script `unstable_1d_interval_laplacian.py` in the repository [11]. That reference demonstrates that a linear space for σ is stable but that enriching this space to include quadratic functions gives the problem indicated above; the script includes also the example from [3] which has a slightly less trivial cosine forcing term and which produces the output in Fig.1.

2.2 $k = 2$ Hodge Laplacian on square

Section 2.3 of [3] contains an example somewhat similar to the one in the preceding section, but in two dimensions. This is a source problem on the square with homogeneous Dirichlet boundary conditions; the equation is

$$\nabla^2 u = 2y(1-y) + 2x(1-x), \tag{8}$$

with solution $x(1-x)y(1-y)$.

Here u appears to be a scalar but it actually corresponds to a 2-form (which clearly has a single component in two dimensions).

A mixed formulation is used with $\sigma \equiv -\mathrm{grad}\ u$, giving

$$\int \sigma \cdot \tau \ d^2x = \int u \ \mathrm{div}\ \tau \ d^2x, \tag{9}$$

$$\int \mathrm{div}\ \sigma \ v \ d^2x = \int f v \ d^2x. \tag{10}$$

In this case it is demonstrated that the combination of piecewise linear elements for (the components of) σ and piecewise constants for u is not numerically stable. A stable discretization is, however, provided by choosing the lowest order Raviart-Thomas elements to represent σ and a DG Lagrange element to represent u. This choice of elements has the property that the divergence operator maps the Raviart-Thomas basis functions onto the space of piecewise polynomials of one order less ([12]); the Raviart-Thomas element therefore is a natural candidate for the discretization of $H(\mathrm{div})$ i.e. representing vectorial quantities which need to have their divergence taken. The outputs in unstable and stable cases are shown in Fig.2.

This example corresponds to the script `k_equals_2_hodge_laplacian_square.py` in [11].

2.3 $k = 1$ Hodge Laplacian, re-entrant corner

This example is taken from [1] (there Example 5.1). Consider the *L*-shaped domain consisting of the region $[-1:1]$ with the lower left quadrant removed. Further consider the 1-form forced Hodge Laplacian problem on this domain

$$\nabla^2 \underline{u} = (1, 0)\,, \tag{11}$$

with $\underline{f} = (1, 0)$ and $\underline{u} \cdot \underline{n} = \mathrm{curl}\ \underline{u} = 0$ on the boundary.

It is known that a re-entrant corner produces a singularity in the output, of the type $|\underline{u}| \propto r^{-\alpha}$ with $\alpha = 1 - \frac{1}{2 - \frac{\varphi_0}{\pi}}$ for corner internal angle φ_0, so a $r^{-\frac{1}{3}}$ singularity in this case (the worst case, for a zero-degree interior angle, is $\alpha = \frac{1}{2}$). The singularity causes problems for conventional FEM,

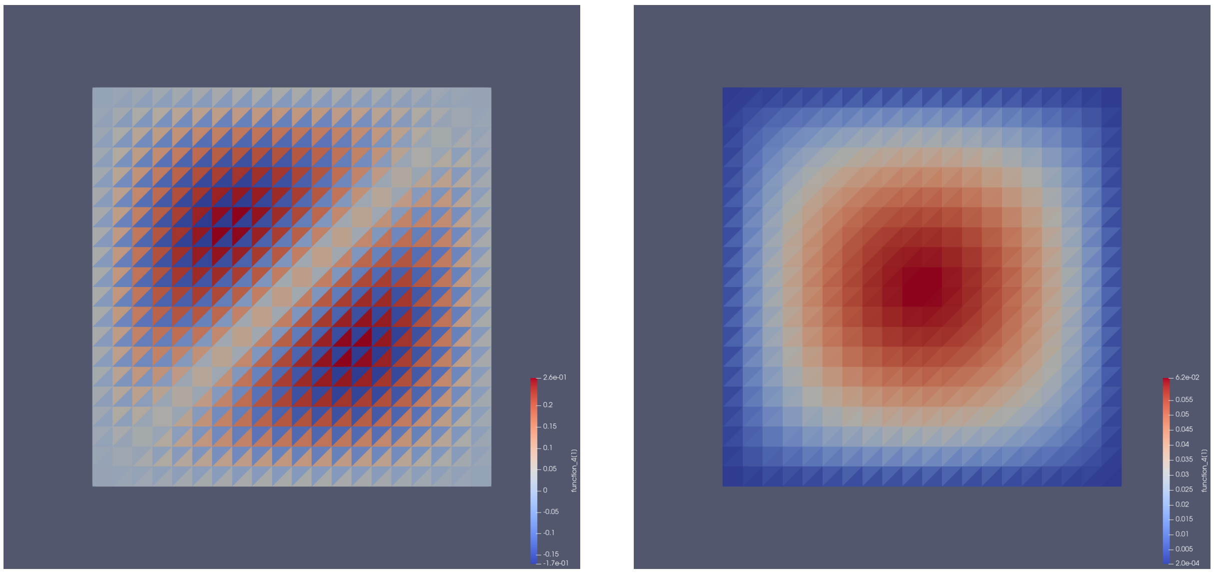


Figure 2: Numerical solutions to Eq.8 obtained using the unstable (left) and stable (right) methods explained in the text.

in particular the conventional FEM basis functions are unable to represent it as it lies without the space of representable functions.

The left-hand-side plots in Fig.3 show output generated using the primal weak formulation using a vector CG discretization (note that this requires the imposition of a Dirichlet condition on the normal component of u, which is straightforward in *Firedrake* only if the boundaries are all aligned to Cartesian axes). The right-hand-side plots were generated using the mixed weak formulation with a combination of elements that is known to give convergence to the correct solution. A lucid explanation of why this primal weak formulation fails in this case is given in Section 5.1 of [1] and a sketch of the argument may serve to enlighten here. The $CG(N)$ elements are used to discretize the space $H^1(\Omega)$ of functions which are themselves square-integrable (L^2) over Ω and whose first derivatives (i.e. gradients) are also L^2 - that is, the discrete subspace spanned by H^1_n contains only functions which are contained within H^1. In the case of the re-entrant corner, the behaviour of the vector function near the corner is as a singularity $r^{-\alpha}$ where α was specified above and the derivative of this function is *not* L^2 - as a result, the output in this representation can *never* converge to the true answer. (Note also that turning to DG elements is unlikely to yield nice results due to the non-uniqueness of fields at element boundaries.) Consider instead discretizing the vector field into a representation that is a subspace of $H(\mathrm{curl})$ and recall that the mathematical definition of a discrete curl is as a line integral over the element boundary divided by the element area. Now the integrals of the singular functions along any given element boundary are actually *finite* (as an illustration, consider $\int_0^1 \frac{dx}{\sqrt{x}} \equiv 2$ despite the singularity of the integrand) and so elements containing functions in $H_n(\mathrm{curl})$ *are* able to represent the corner singularity. Therefore, one is invited to consider for the representation of the 1-form $H(\mathrm{curl})$-conforming elements, for which see the list at [13], and note that the implementations on simplices in *Firedrake* are the Raviart-Thomas 'edge'-type (called `RTE`), also known as Nédélec curl elements of the first kind (`N1curl`) and also Whitney elements; or the Brezzi-Douglas-Marini 'edge'-type (called `BDME`), alternately known as Nédélec

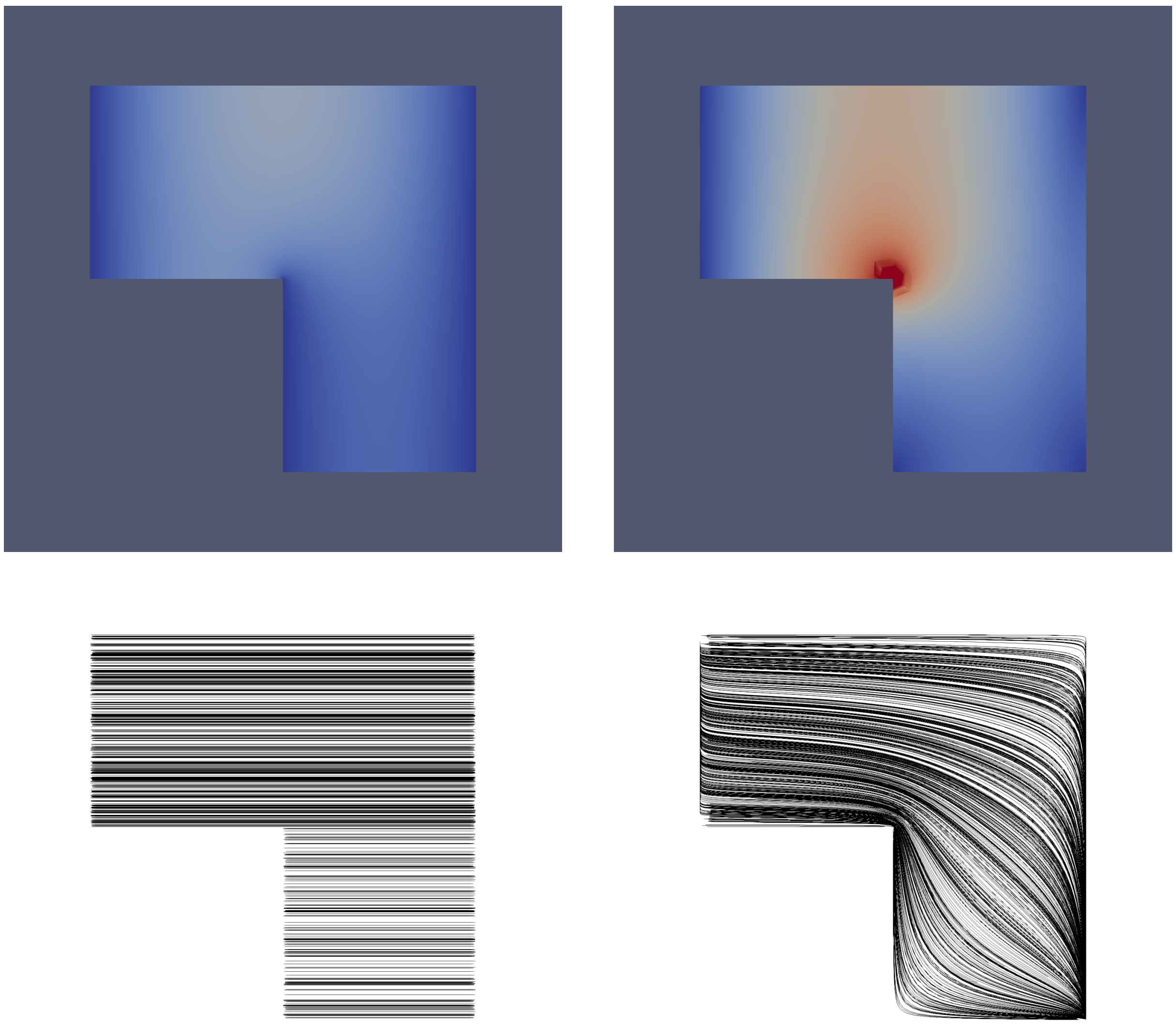


Figure 3: Outputs for the re-entrant corner problem treated using a standard Galerkin discretization (left-hand-side plots) and a combination of Raviart-Thomas ‘edge’ elements for the 1-form and CG elements for the divergence (right-hand-side). The colour maps (top) show the magnitude of the 1-form field and the lower plots are the streamlines. The scale in upper figures is identical (the scale is meaningful because the problem is inhomogeneous). Note that the colour map for the spurious solution on the left *looks* plausible - though the dominance of one Cartesian component of the 1-form might raise suspicions.

curl elements of the second kind (`N2curl`). The mixed variational form (Eq.4) is used (note there are no nontrivial harmonics in this case as the domain is simply-connected).

It is found that clean and consistent-looking plots are obtained for both the above sets of spaces using the default `solver_parameters` of *Firedrake*; see Fig.3. An implementation of the primal formulation (which generates the spurious solution) can be found in the script `re_entrant_corner_primal.py`, and a working FEEC implementation in `re_entrant_corner.py`, both in [11].

2.4 The $k = 1$ Hodge Laplacian in non-trivial topologies

The de Rham cohomology space is that of solutions to the equations

$$\begin{aligned} du &= 0, \\ \delta u &= 0 \end{aligned} \tag{12}$$

and subject to a boundary condition e.g. $u \cdot n = 0$.

In the $k = 1$ case, the Eqs.13 mean respectively $\mathrm{curl}\ u = \mathrm{div}\ u = 0$ with the additional conditions that the trace (here the normal component of u) vanishes on the boundary. It is clear that these conditions together mean that any solutions lie in the kernel of the Hodge Laplacian operator and are solutions to Eqs.4 with $f = 0$ (and no p or q needed). It is shown in [1] that the dimension of the cohomology space is equal to the first Betti number of the manifold, which in two dimensions means simply the number of holes in it; this gives the number of linearly-independent solutions. The textbook [1] also explains that the harmonics give a representation of the $k = 1$ cohomology of the domain and proves that the correct choice of discretization allows the discrete form to retain the same (Theorem 5.1, isomorphism of cohomology).

In a domain with non-zero first Betti number, there are one or more zero-eigenvalue solutions to

$$\nabla^2 \underline{u} = \lambda \underline{u}, \tag{13}$$

because the harmonics are in the kernel of the Hodge Laplacian.

Some additional material on 2D harmonics is included in Appendix B.

2.4.1 Harmonics of doubly-connected regions

On a circular annular domain, there is one harmonic solution (up to multiplication by a constant) which is the usual vortex (here written in familiar vector notation)

$$\underline{u} = \left(\frac{-y}{x^2 + y^2}, \frac{x}{x^2 + y^2} \right). \tag{14}$$

It is not currently straightforward to implement in *Firedrake* a primal formulation of this circular annular vortex, due to the need to apply a Dirichlet boundary condition on the normal component of the vector field at the boundaries (see [14]). It is thus easier to explore numerically the same problem on a square annulus where the essential boundary condition is straightforwardly imposed (note the solution has re-entrant corners but it will be seen at least that the FEEC method chosen handles these correctly).

Consider attempting to use conventional FEM to find the vortex solution as the solution to the eigenvalue problem defined by Eq.13. A standard primal discretization using a vector CG space leads to the solution shown in the left-hand-side plots of Fig.4. Additionally, the primal weak formulation gives the results shown in Table 1 for the lowest eigenvalue, the latter (as with other eigenvalue computations in this section) obtained using the Scalable Library for Eigenvalue Problem Computations (*SLEPc*) developed at the Universitat Politècnica de València, Spain [15], which is called from within *Firedrake*.

element order p	λ_{min}
1	3.8877955
2	3.7499638
3	3.7083853
4	3.6880550
5	3.6761782
6	3.6684848
7	3.6631459
8	3.6592514

Table 1: Lowest eigenvalue for the $k = 1$ Hodge Laplacian problem on the square annulus, using the primal weak formulation with CG Lagrange elements of order p. The result, though appearing to converge, is completely spurious as the true lowest eigenvalue is exactly zero.

It is known that the true lowest eigenmode must have an eigenvalue of exactly zero and hence the eigenmode found here is a numerical artifact. A contrasting implementation that finds the harmonic correctly, using a mixed formulation with Raviart-Thomas 'edge' elements to represent the 1-form and CG elements to represent the divergence, can be found in `eigenvalue_problem_square_vortex.py` at [11]; the primal formulation is `eigenvalue_problem_square_vortex_primal.py`.

Note that it is possible to obtain these solutions by conventional methods (no FEEC) by combining a known vortex solution (e.g. Eq.18) and a scalar Laplacian problem. The trick is to realize that, unless the solution topology changes, two (harmonic!) solutions of the Hodge Laplacian problem on two domains are related by a gradient; the resulting flow will automatically satisfy the zero-curl equation (because the curl of a gradient vanishes) and also the zero-div condition if the scalar obeys the scalar Laplace equation. The boundary conditions are patched up by applying a Neumann condition to remove unwanted normal components at the boundaries. The scalar Laplacian problem avoids the problems associated to harmonics (as long as the null space of constants is dealt with) and also the non-L^2 issue because the scalar problem does not have corner divergences (they appear in the *gradient* of the scalar). This technique is demonstrated in the script `square_vortex_cohomology_alternate.py` in [11].

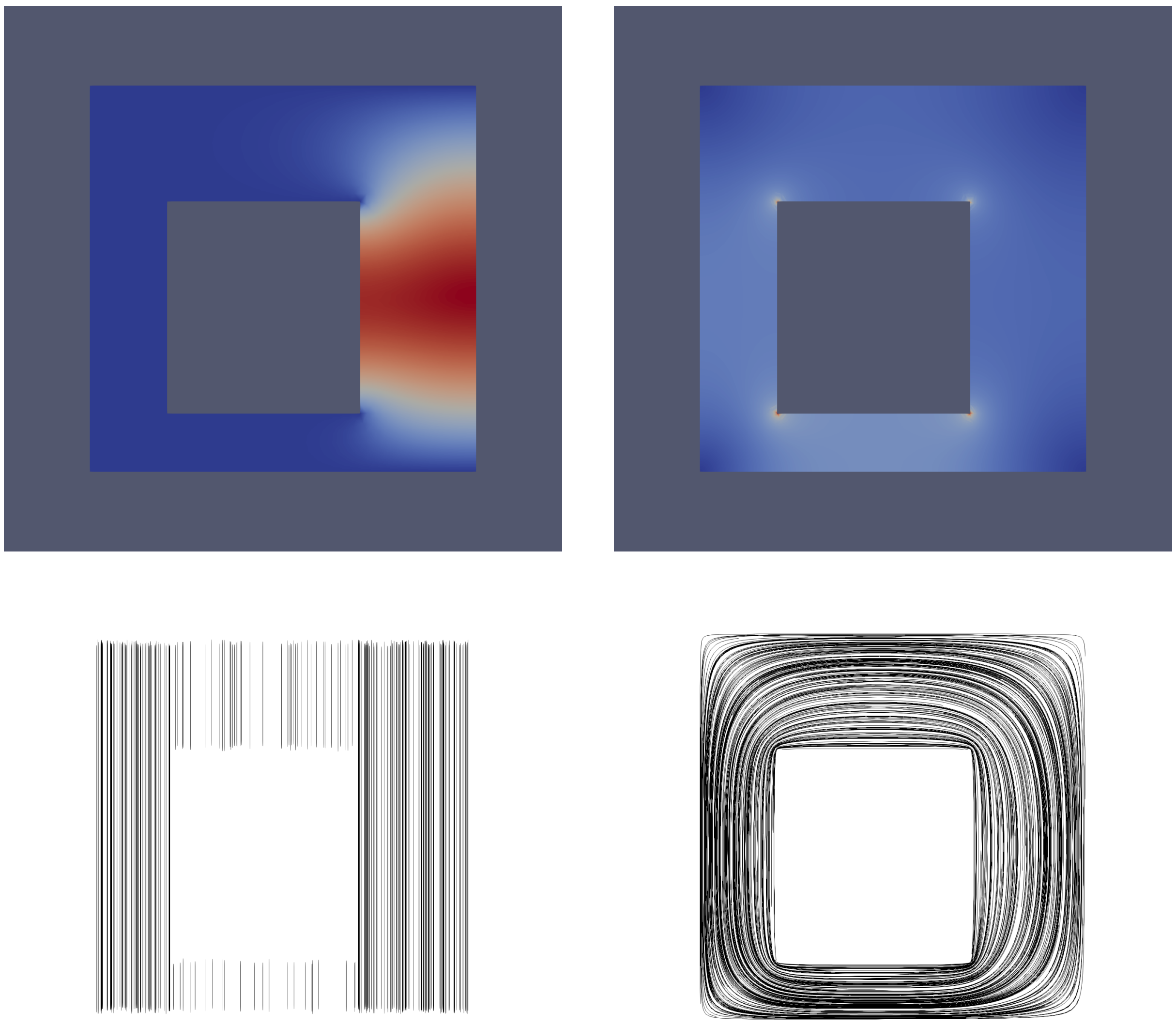


Figure 4: Lowest eigenmode for the square annulus obtained using a primal CG discretization (left) and using a mixed discretization with Raviart-Thomas ‘edge’ elements to represent the 1-form (right). Note that the overall scale is meaningless for this homogeneous problem.

2.4.2 Harmonics of triply- and higher-connected regions

The triply-connected region considered here is defined as a bounded or unbounded region of the plane with two circular discs removed. Because the dimension of the cohomology space is equal to the first Betti number of the manifold, two harmonics are expected. For an unbounded plane domain, one of these is the counter-rotating vortex pair solution

$$\underline{u} = \left(\frac{-2xy}{(x^2+y^2)^2 + a^4 + 2a^2(y^2-x^2)}, \frac{x^2-y^2-a^2}{(x^2+y^2)^2 + a^4 + 2a^2(y^2-x^2)} \right). \tag{15}$$

This is in fact just the gradient of the multiple-valued coordinate σ in the bipolar coordinate system ([16]) (cf. the single circular vortex, which is the gradient of the polar angle).

Cohomology theory means there must be a second solution and a co-rotating vortex solution (possessing non-circular inner boundaries) is easily found by considering the streamfunction for two same-sense vortices, and is given by (see, for example, [17])

$$\underline{u} = \left(\frac{-y(x^2+y^2+a^2)}{(x^2+y^2)^2 + a^4 + 2a^2(y^2-x^2)}, \frac{x(x^2+y^2-a^2)}{(x^2+y^2)^2 + a^4 + 2a^2(y^2-x^2)} \right). \tag{16}$$

It is straightforward to map this co-rotating vortex pair back onto the domain with two circles removed, and also to incorporate an outer boundary, using the ‘adding a gradient’ technique mentioned earlier. These solutions are exhibited in Fig.5 for a domain with a square outer boundary. The necessary gradient can be obtained using *Firedrake*, or alternatively using the advection-diffusion-reaction solver of *Nektar++* and post-processing in *ParaView* (i.e. calculating gradient and adding back the analytic solution) - see the script `double_vortex_cohomology_alternate.py` at [11] for the former. Note also that the higher-order geometry element of *Nektar++* can be used to improve the accuracy of the gradient solution; for this, the second-order element geometry capability of *Gmsh* can be used for the curved boundaries when generating the mesh (meshing command: `gmsh -2 filename.geo -format msh2 -order 2`).

The technique using FEEC is exhibited in the script `eigenvalue_problem_double_vortex.py` at [11]. Note that the eigenstates found by *SLEPc* do not correspond exactly to the counter- and co-rotating vortex solutions discussed above but rather to two mutually-orthogonal linear combinations of them (also, the symmetry of the two-vortex example above invited classification of the states by symmetry but a more general logic is to take a Lagrange-like basis where for each eigenstate the flow circulation vanishes for all but one domain hole). A more complex example, consisting of a square domain with eight holes and cohomology space of corresponding dimension, is exhibited in Fig.6. I n this *Firedrake* example, the *SLEPc* solver finds eight zero-eigenvalue eigenstates in a few seconds and again they seem to be rather arbitrary in terms of basis choice.

2.4.3 Source problems in the presence of harmonics

Source problems on domains with non-trivial cohomologies also encounter problems associated with the presence of harmonics. An example, taken from [1], is

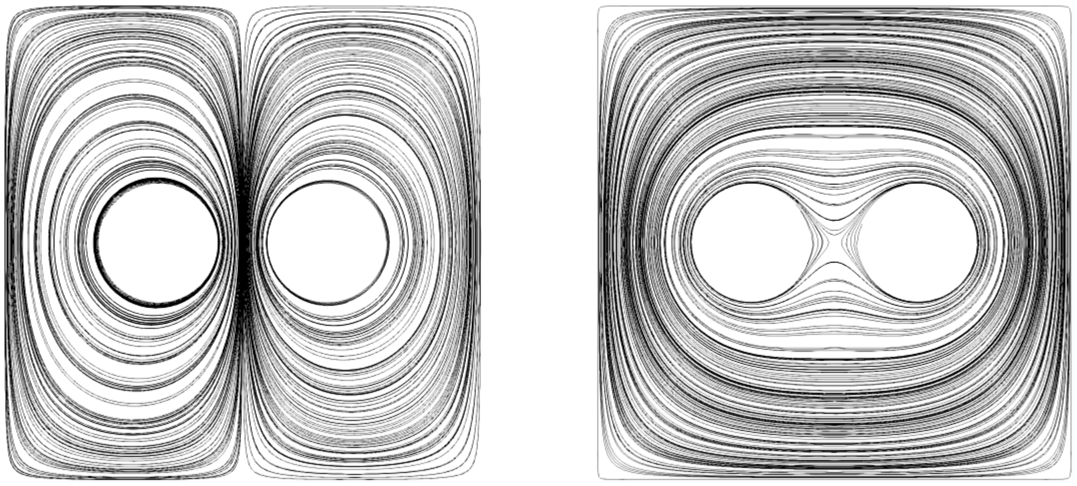


Figure 5: Streamlines of 1-form harmonics on the square with two circles removed. These plots were created using the semi-analytic technique outlined in the text in order to obtain pure counter- (left) and co-rotating (right) vortex pair solutions.

$$\nabla^2 \underline{u} = (0, x) \tag{17}$$

on the annulus $\frac{1}{\sqrt{2}} \leq \sqrt{x^2+y^2} \leq \sqrt{2}$ and subject to the boundary conditions $\underline{u}\cdot\underline{n} = (\mathrm{curl}\ \underline{u})\cdot\underline{n} = 0$.

The obvious problem with this is the ill-posedness: any multiple of the vortex solution can be added to a solution and the problem is still satisfied (indeed, a naive implementation in *Firedrake* produces obviously-nonsensical output). This difficulty can be removed using the `nullspace` feature of *Firedrake* - the software is given the harmonic(s) as the basis for the null space. However, a standard primal Galerkin discretization of the problem still fails, for the reason that the solution obtained is polluted by some multiple of the harmonic, even though the solver has been informed of the harmonic mode. Results from the standard method (using per-component order-6 CG elements) can be seen in the left-hand plots of Fig.7. On the right-hand side of the same figure are the outputs from a FEEC discretization that preserves the cohomology of the continuum problem, in this case order-2 Brezzi-Douglas-Marini 'edge' elements for the 1-form and order-3 CG elements for the divergence. These outputs can be compared with the very similar results in Figs. 5.1 of [1] and 2.4 of [3].

The numerical outputs, as well as an analytic solution derived starting from the scalar vorticity equation $\nabla^2\Omega = 1$, are produced by the script `source_problem_on_annulus.py` in [11].

3 Nektar++ developments

It is an ongoing goal of NEPTUNE to shape a version of *Nektar++* into a library to provide finite-element capabilities tailored to the needs of the project. There have been a number of develop-

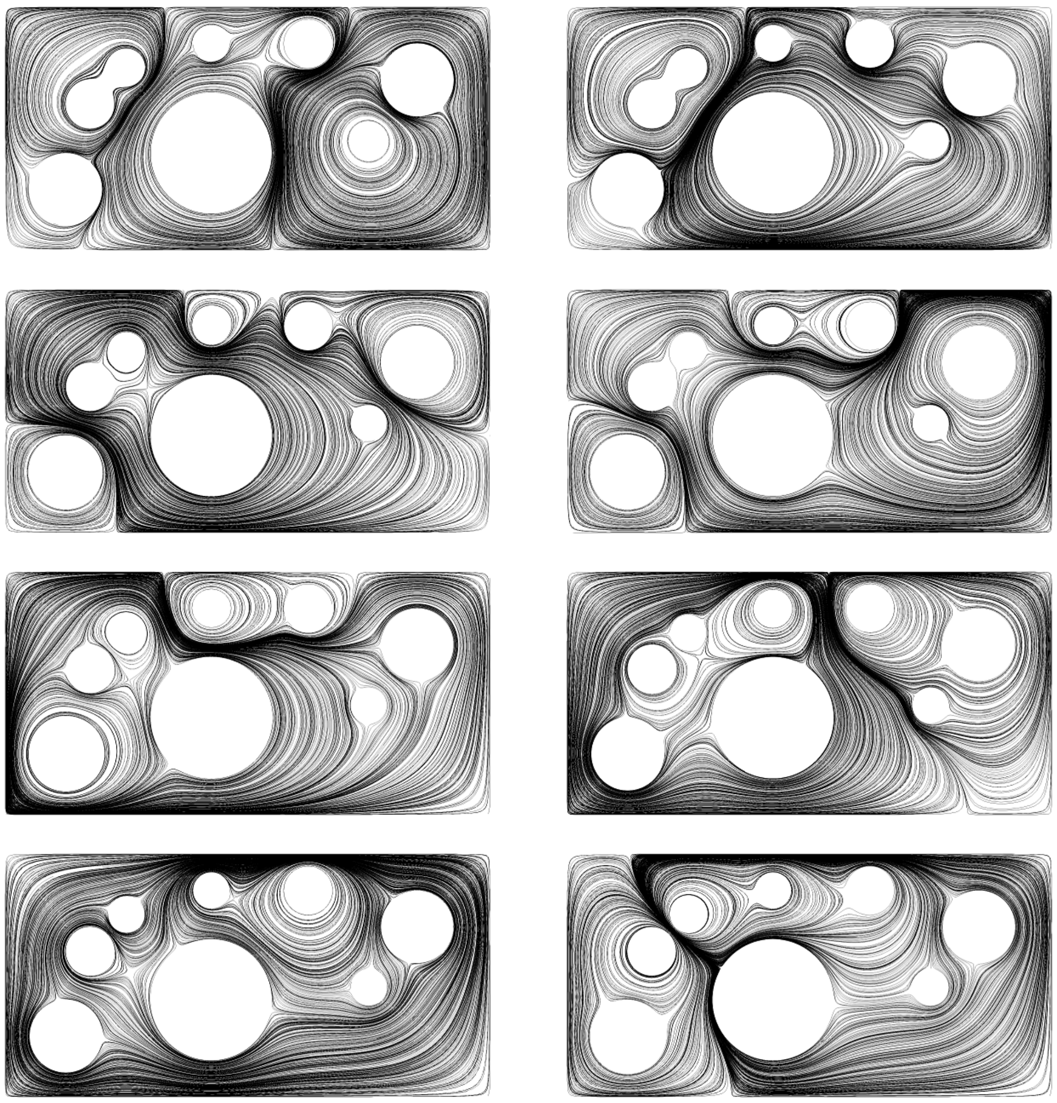


Figure 6: The 1-cohomology of a domain containing eight holes, as calculated by *Firedrake*; plots show flow streamlines. The calculation uses order-3 Raviart-Thomas ‘edge’ elements to represent u and order-3 CG Lagrange elements for the divergence σ and finds all eight representatives in a few seconds on a laptop.

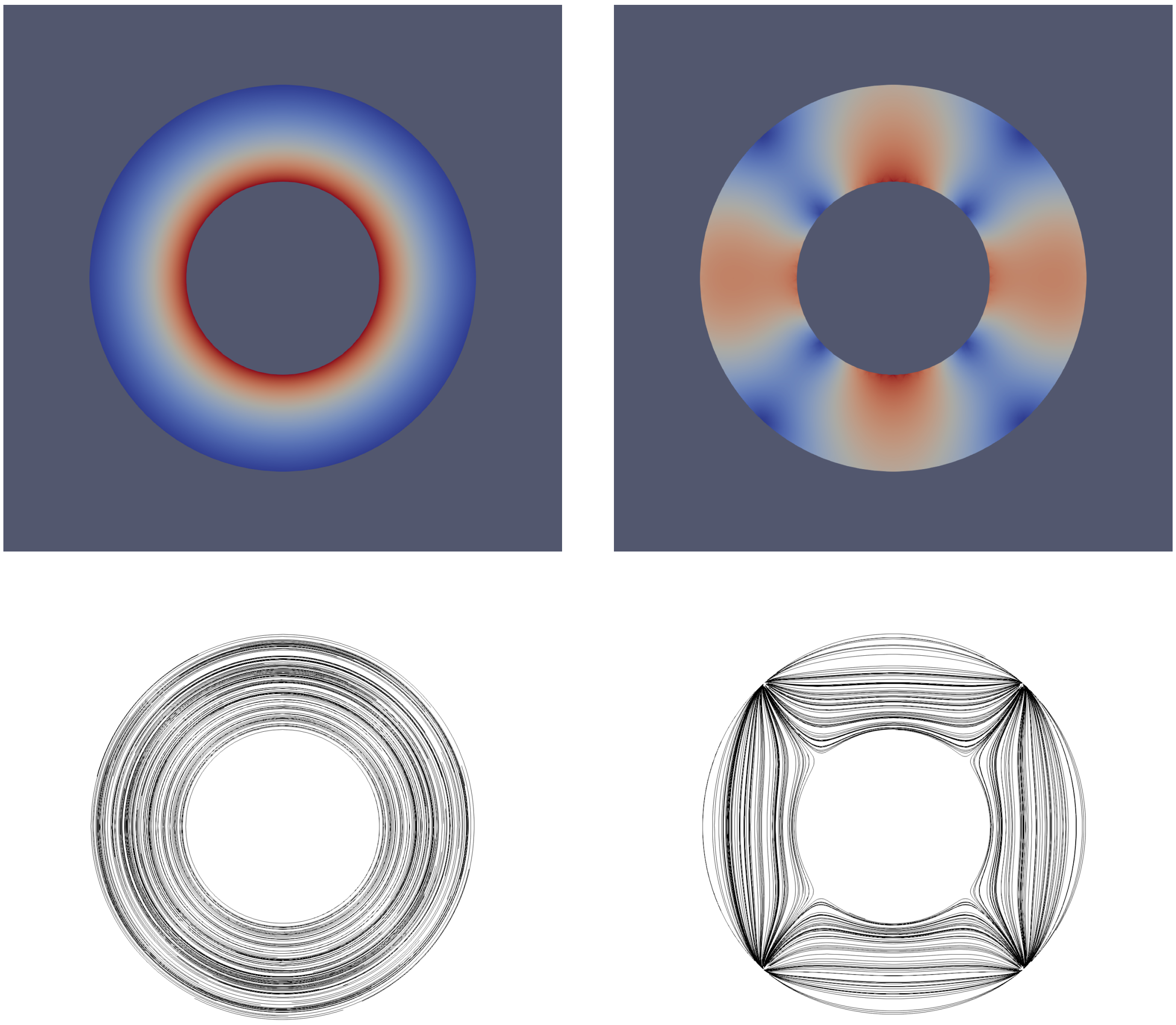


Figure 7: Outputs for the source problem on annulus described in the text using a primal CG discretization (left) and using a mixed discretization with order-2 Brezzi-Douglas-Marini ‘edge’ elements to represent the 1-form and order-3 CG elements for the divergence (right). The primal formulation gives a solution dominated by the harmonic vortex solution (the amplitude is many orders of magnitude larger than that of the true solution). The analytic solution to this inhomogeneous problem is visually identical to and matches in scale the FEEC numerical solution except that it avoids the artifacts at the facet boundaries visible in the numerical solution (curved-sided elements in *Firedrake* are not currently as straightforward as they are in *Nektar++*). Note the existence of four stagnation points on each boundary of the true solution.

ments during the period covered by this report, by UKAEA personnel and by the developers of *Nektar++* and brief descriptions of these follow.

3.1 Removal of XML session file

One factor that complicates the integration of *Nektar++* functionality in NEPTUNE is the requirement that configuration options be specified in XML 'session' files. Running multiple solver instances over the course of a NEPTUNE calculation, therefore, would mean keeping track of several external files and/or manipulating their contents on-the-fly.

To address this, a number of changes to the *Nektar++* core libraries have been made, separating out code that defines a Session object from code that parses XML. This allows runtime options to be supplied programmatically instead, significantly simplifying the process of using *Nektar++* as a library in NEPTUNE. The 'Session' class now contains only core member variables, getter and setter functions, with all XML-based functionality moved into a subclass, called 'SessionXmlReader'. To set up a *Nektar++* solver from calling code, it is now possible to instantiate a 'Session' object using a constructor that accepts various C++ standard template library containers as arguments. For instance, fluid field names can be supplied as a vector of strings and solver options as a string-to-string map. The mesh can either be read from XML, as before, or instantiated programmatically by supplying vertex positions and vectors of vertex indices that determine the shape of each element. Finally, some example code that demonstrates programmatic solver set up for 1D and 2D problems has been added. The two examples double as regression tests for the framework, ensuring that it continues to work as intended as the *Nektar++* code base evolves.

A merge request has been created and is currently under review by the *Nektar++* core development team; the changes are expected to be incorporated into a release of the software in the near future.

3.2 Interfacing with particles

NESO (NEPTUNE Exploratory SOftware) is a UKAEA-developed proxyapp coupling particles and finite element methods. It is expected to form the core of a more developed NEPTUNE code.

The particles aspect of NESO is called NESO-Particles (repository at [18]); it is written in an abstract manner that is independent of the choice of FEM library. The interface layer that sits between *Nektar++* and NESO-Particles has been written. Fundamentally this layer allows particles to logically exist on and traverse 2D linear *Nektar++* meshes. The interface layer uses functionality to copy *Nektar++* geometry objects between MPI ranks. By copying relevant geometry objects between the subdomains that result from the mesh decomposition process, this interface layer builds the halo regions that are required in order to map positions in space to *Nektar++* geometry objects.

The following subsections contain additional relevant details.

3.2.1 Projection

The functionality to convert a particle representation to a continuum representation proceeds via an L^2 Galerkin projection. This projection interface is expected to be on the critical path of the simulation and should be implemented with computational efficiency in mind. Consider a source point cloud of particles where each particle p has a position x_p, quantity (scalar or vector) q_p (e.g. electric charge, current) and a Dirac-delta shape $\delta(x - x_p)$. This point cloud can be represented in the FEM function space via an L^2 Galerkin projection to produce a function $q(x) = c_i\phi_i(x)$, (scalar or vector valued, where ϕ_i is the i^{th} basis function, here with coefficient c_i), which may be used as a source or coefficient term in some other equations. The projection takes the form (using the Einstein convention; i is the free index)

$$\int_V \phi_i\phi_j \, dV c_j \equiv M_{ij}c_j = \sum_p q_p \int_V \delta(x - x_p)\phi_i \, dV \equiv \sum_p q_p\phi_i(x_p).$$

This is an $Ax = b$ problem where A corresponds to the mass matrix with components M_{ij}, x to the vector of DOFs c_j, and b to $\sum_p q_p\phi_i(x_p)$, i.e. the vector where the i^{th} component is the sum of the values of the i^{th} basis function ϕ_i evaluated at each and every particle's position x_p.

The interface and implementation may be written to assume or require that the particles that form the point cloud are already binned into mesh cells, i.e. particles may be stored on a per-cell basis, or a map between cell identifiers to particle indices may exist. Furthermore, as part of the cell binning process, the reference position of the points in the mesh cell may be known by the calling implementation and reused by the *Nektar++* implementation.

For future consideration, note that simulations could involve fast-moving particles that are treated in a time-averaged manner where the particle shape function and weight is a trajectory over a time step, e.g. the contribution from a particle exists over a line through space rather than a single point and quadrature along these trajectories will be required.

3.2.2 Evaluation

Considering a source field in *Nektar++*, the evaluation interface should evaluate the function at a set of arbitrary points. As in the projection case, these arbitrary points may already be binned into mesh cells and the reference coordinates known. Also as in the projection case, this function will be called at every time step for every particle and hence is expected to be performance-critical.

3.2.3 Mesh augmentation (halos)

The NESO/NESO-Particles implementation duplicates mesh objects to enable the transfer of particle data over the globally decomposed mesh - this mesh augmentation is critical to how NESO determines where to send particles. The details of which mesh objects are duplicated and where is currently determined by NESO by using serialization/de-serialization also implemented in NESO. Exactly which mesh elements are duplicated and why is a PIC-specific choice and so this logic may not sit naturally within a native FEM code base. Regardless of who retains ownership of this portion of code, certain functions to handle *Nektar++* mesh objects are needed:

1. (De)Serialization of mesh objects. NESO (or any other implementation) can currently inspect the geometric properties of *Nektar++* mesh objects and use this geometric information to determine which objects are to be communicated between MPI ranks. Helper methods to serialize an existing object (into bytes) and reconstruct the object from bytes would abstract user codes (i.e. NESO) from the exact details of each type of geometry object.

2. Construction of *Nektar++* tree data structures for geometry objects. The primary use of these duplicated geometry objects is to map a point in space to a geometry object (and as a by-product compute the reference coordinates for use in projection/evaluation) which determines the owning MPI rank. If *Nektar++* has data structures that already facilitate mapping points to geometry objects (e.g. R-trees) then it would be advantageous to be able to construct these data structures using the duplicated objects.

3. Future time stepping methods in NEPTUNE / NESO may be (semi-)implicit and iteratively compute new particle positions via a sub-stepping process that proposes new particle positions which may not be final. In this scenario it is advantageous to be able to evaluate *Nektar++* fields from cells in the halo region on each rank. This halo evaluation allows particles to be stepped into the final position for each (full) time step before particles are transferred between MPI ranks.

3.3 GPU kernels

Nektar++ has been augmented with the capability to run selected kernels (e.g. transforms between mode coefficient and spatial value, Helmholtz operator, derivative evaluations) on an NVIDIA GPU, via kernel implementations in CUDA. The implementation includes a diagnostic of the performance achieved (in GFLOPS). One restriction of these kernels is that the polynomial element order is currently capped at a maximum of 11.

This capability is to be seen as a stepping-stone to a longer-term goal of making the entire *Nektar++* framework concordant with modern approaches to performance portability, especially efficient operation on current and forthcoming GPU-dominated systems from AMD, Intel, and Nvidia.

3.4 Python interface and h- and r-adaptation for curves

Deliverable 1.4 of Grant T/NA078/20 specifies a Python interface, documentation and CI for the new additions the the *NekMesh* mesh generator. The code can be found at [19]. Briefly, this new code is capable of local h-refinement in the vicinity of a specified CAD curve (shown in Fig.8) and also anisotropic r-refinement meaning moving mesh nodes closer to a specified CAD curve (Fig.9), the curves here being proxies for magnetic field lines. These features are demonstrated in the figures and can be generated by running the script `run-adaption.py`. Note that the example includes a simple illustration of a higher-order (curved-sided) geometry for the representation of the embedded circle.

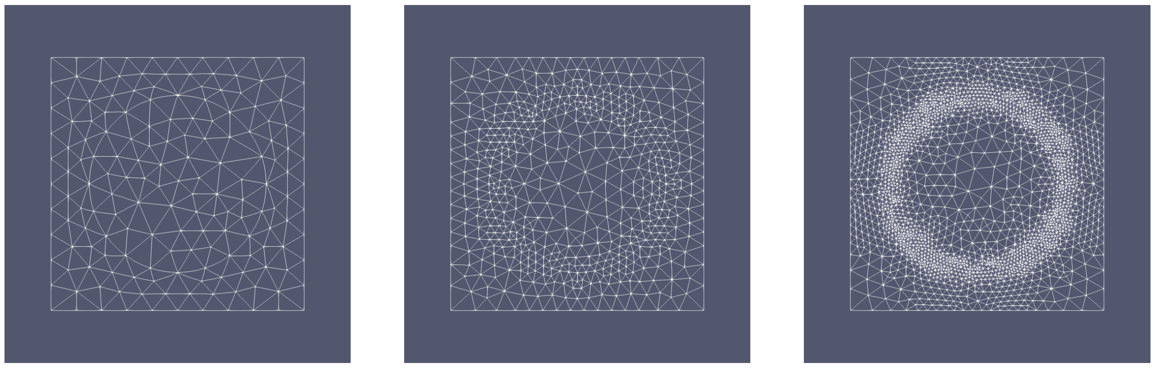


Figure 8: h-refinements of a simple 2D mesh consisting of a circle embedded in a square domain. The left-most plot shows the mesh prior to local h-refinement in the vicinity of the circle. To the right are plots h-refined locally to the circle using δ parameters of 0.0035 and 0.0015 respectively.

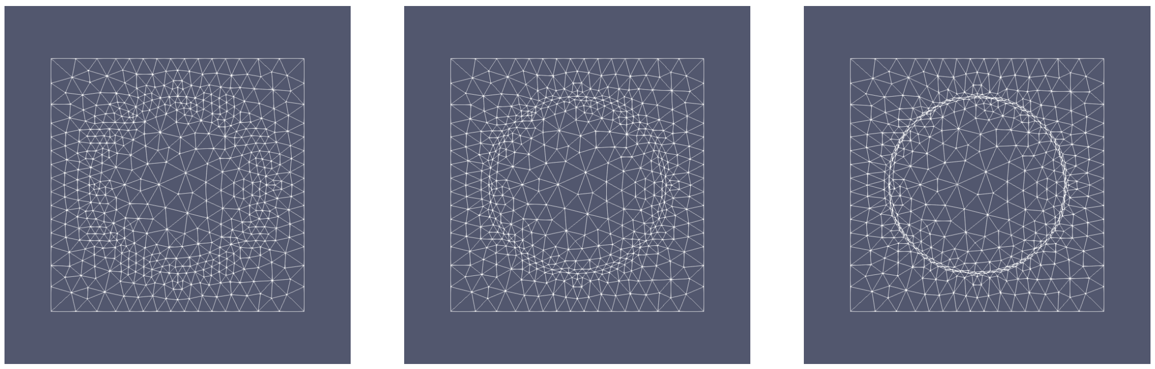


Figure 9: r-refinements of a simple 2D mesh consisting of a circle embedded in a square domain. The left-most plot shows the mesh prior to local r-refinement in the vicinity of the circle. To the right are plots r-refined with `radaptscale` parameters of 0.7 and 0.4 respectively such that the mesh nodes are displaced toward the circle without alteration to the mesh connectivity.

It is also possible to use the Python interface to generate simple structured meshes of quadrilaterals or triangles; see the script `gen_mesh.py` in the repository [20].

3.5 Memory use in *Nektar++*

It has been determined ([21]) that the incompressible Navier-Stokes solver of *Nektar++* retains in memory *two* copies of the system matrix - pre- and post-static condensation versions of the same object. Only one is strictly needed for the computation. This results in larger-than-expected memory use by this solver; modifications are being considered by the developers.

4 Summary

The finite element exterior calculus approach offers advantages for numerical methods in providing guarantees of stability and convergence. This was demonstrated in examples taken from the literature, which were implemented in *Firedrake* due to the wide range of finite element types implemented in that package. The scripts were committed to a public GitHub repository which is intended as a pedagogical resource. The examples in this report focussed on the Hodge Laplacian problem and demonstrated the need for FEEC in cases resistant to traditional approaches, for several different reasons: problems where poorly-chosen combinations of finite elements gave unstable discretizations, problems containing singular fields, and problems on domains with non-trivial topology. For NEPTUNE, the desirable properties of FEEC as applied to coupled particle-continuum systems are likely to be the main area of interest and will be the focus now that some of the foundations of the theory are understood.

Developments to the *Nektar++* code base intended to align it to use as a FEM library for NEPTUNE were exhibited; this is an area of ongoing work. In particular, the coupling to a PIC implementation is progressing, via the UKAEA in-house NESO code, reinforcing the case for the implementation of structure-preserving methods such as FEEC.

ET acknowledges assistance from Colin Cotter, Patrick Farrell, and other members of the *Firedrake* community.

Acknowledgement

The support of the UK Meteorological Office and Strategic Priorities Fund is acknowledged.

A *Firedrake*

Firedrake is a framework for the solution of PDEs. The mathematical problem is specified using the Unified Form Language from the FEniCS Project ([22]), which represents a powerful domain-specific language for expressing PDEs. The solvers themselves leverage PETSc ([23]).

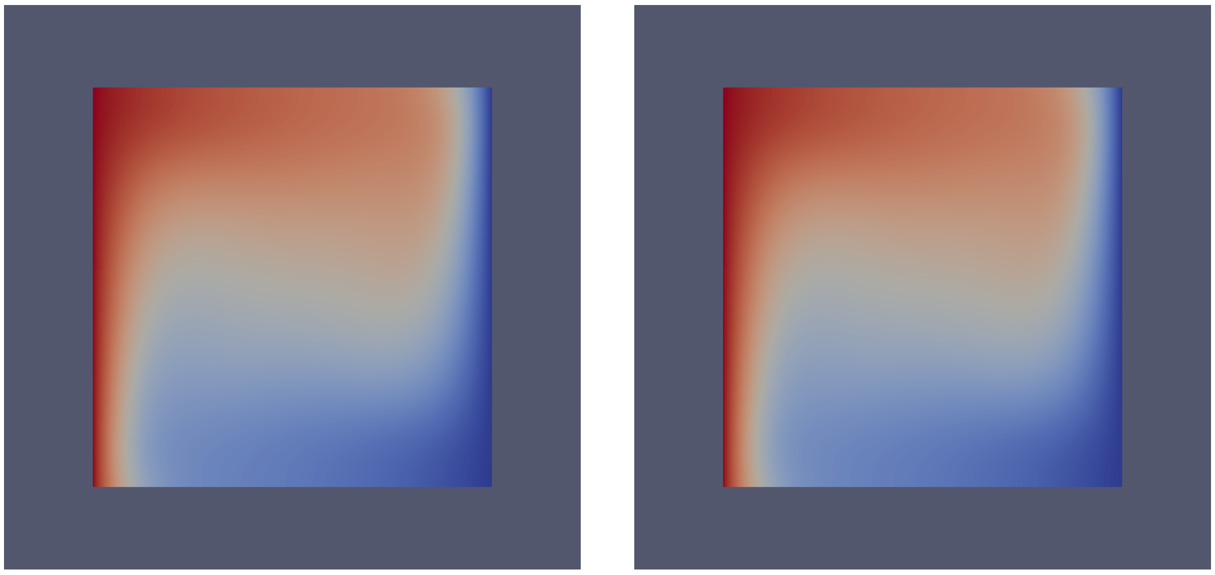


Figure 10: Temperature field of a convection problem in a unit square cavity with the right-hand side maintained at unit temperature and the left at zero temperature, Rayleigh number 5×10^4 and Prandtl number 7.0. The left-hand plot is obtained from a steady-state solution in *Firedrake* and the right-hand one from time-evolving to the steady state in the incompressible Navier-Stokes solver of *Nektar++*. The outputs are visually identical.

Firedrake has major strengths in that the problem specification mirrors very closely the mathematical notation for the weak form, and the approach of using such a DSL in conjunction with behind-the-scenes code generation facilitates the implementation of a wide variety of problems, which can be specified very economically by Python script. This is somewhat in contrast to *Nektar++*, in which new solvers must be implemented in C++. *Firedrake* also implements geometrical constructs such as vectorial function spaces, and supports a wide range of vector and scalar finite element types. Conversely, known relative strengths of *Nektar++* are tools for building curved geometry meshes and the ability to mix element geometries in the same mesh (e.g. hexahedra and tetrahedra) where the hexahedra are often used to mesh boundary layers efficiently.

Another particular strength of *Firedrake* is the ability to handle equations with null spaces, a technique used in the script `source_problem_on_annulus.py` in [11]. For scalar Neumann problems, *Nektar++* deals with this issue by fixing one particular value on the boundary but this approach has in the past been seen to generate problems (see Appendix B of [24]).

Equivalent output generated by *Firedrake* and by *Nektar++* is shown in Fig. 10. The *Nektar++* data is generated by using the incompressible Navier-Stokes solver to time-evolve an initial condition to a laminar convective steady state. The *Firedrake* example is essentially the same as the example at [25] and performs a linear solve to obtain the steady state, using Taylor-Hood elements which are known to provide a convergent solution. Close correspondence is seen in the resulting temperature field outputs.

B More on harmonics and vortices

B.1 Single vortex

The single-vortex solution on a circular annulus is

$$\underline{u} = \left(\frac{-y}{x^2+y^2}, \frac{x}{x^2+y^2} \right). \tag{18}$$

It is a property of this solution, when viewed as a fluid flow, that the differential rotation (with radial distance) causes a paddle-wheel probe to circulate the origin in such a way that its orientation remains constant (this is the zero-curl property). It is worth noting that this is a stationary solution to the Navier-Stokes equations of fluid mechanics that has zero viscous dissipation (even though there is non-uniform velocity) and where the pressure gradient obeys the simple Newtonian law $|\nabla p| = \frac{\rho v^2}{r}$ i.e. it provides the centripetal force. Note also that the zero-curl equation means that the above vortex has zero vorticity except at the origin, where there is a singularity in the fluid velocity, a difficulty that can be avoided by considering only annular domains which do not contain the origin.

The solution can be seen to be the gradient of the polar angle φ, but it is not an exact differential form because φ is not single-valued on the domain. It is also the curl of the streamfunction given by the familiar $\psi = \frac{K}{2\pi} \ln r$, where K is the vortex strength (and recall the Riemann surface for the function $w = \ln z$).

It is also possible to define the same problem with the alternate boundary condition that the angular component of the velocity vanish on the boundaries. The solution to this is, up to overall sign, the Poincaré dual of Eq.18 (apply the Hodge star!), under which the streamlines and equipotentials are swapped, i.e.

$$\underline{u} = \left(\frac{-x}{x^2+y^2}, \frac{-y}{x^2+y^2} \right). \tag{19}$$

This is the radial flow 'draining bathtub' solution (which finds interesting applications in the area of fluid-mechanical black hole analogues); it is clearly curl-free and is divergence-free (radial velocity $\propto \frac{1}{r}$) except at the origin. As an aside, it has not yet proven possible to obtain the 'draining bathtub' solutions directly in *Firedrake* (this problem is technically the 'de Rham complex with boundary conditions'; see [26] for a discussion)

B.2 Double vortex

The triply-connected region considered is defined as a bounded or unbounded region of the plane with two circular discs removed; two harmonics are expected. For an unbounded plane domain, one of these is easily found using the bipolar coordinate system [16] (in this and later, a is the distance of each focus from the origin)

$$ds^2 = \frac{a^2}{(\cosh\tau - \cos\sigma)^2}\left(d\tau^2 + d\sigma^2\right). \tag{20}$$

Writing down the curl and div equations in these coordinates, an obvious solution is (translated to Cartesians) Eq.15, which is in fact just the gradient of the multiple-valued coordinate σ (cf. the single circular vortex).

It turns out that the streamlines and equipotentials of this solution correspond to the Apollonian circles known since antiquity (these are the curves of constant σ or τ as defined in Eq.20). Physically, this solution corresponds to the flow associated to a pair of counter-rotating vortices in uniform translation and possessing zero aggregate angular momentum; note this is also the solution for the magnetic field lines surrounding two infinite wires carrying counter-flowing currents. (A somewhat similar fluid flow with two counter-rotating vortices can be generated by dragging a teaspoon in a cup of fluid but there the non-slip boundary conditions lead to additional dynamics.)

The circles corresponding to the streamlines are found by integrating the expression for the tangent velocity,

$$\frac{dy}{dx} = \frac{-x^2 + y^2 + a^2}{2xy}, \tag{21}$$

a Bernoulli-type ODE with solution $x^2 + y^2 + a^2 = cx$ for constant c.

Cohomology theory means there must be a second solution, though this is not naturally found using the bipolar coordinates (the serendipitous Apollonian-circles property is not shared by this other solution). A co-rotating vortex solution (possessing non-circular inner boundaries) is, however, easily found by considering the streamfunction ψ given by (see, for example, [17])

$$\psi = \ln\sqrt{(x+a)^2 + y^2} + \ln\sqrt{(x-a)^2 + y^2} \tag{22}$$

(which is obviously the superposition of two same-sense vortices in a Green's function approach; note that a similar formula, $\tanh\psi = \frac{2ax}{a^2+x^2+y^2}$, can be used to define the Apollonian circles of the counter-rotating case), giving the velocity field $\underline{u} = \mathrm{curl}\ \psi$ (equivalent to $-\nabla^{\perp}\psi \equiv -e_z \times \nabla\psi$ and often called 'rot ψ' in the literature) corresponding to Eq.16.

That the streamlines are not circles can be seen by solving the equation

$$\frac{dy}{dx} = -\frac{x(x^2 + y^2 - a^2)}{y(x^2 + y^2 + a^2)} \tag{23}$$

to $x^2 + y^2 + a^2 = \sqrt{c + a^4 + 4a^2x^2}$ (in general, not a circle, and note $c = -a^4$ corresponds to the circles of zero radius at the vortex cores). This solution represents two co-rotating vortices and as such has zero linear momentum and a finite angular momentum (it is also the solution for the magnetic field lines surrounding two infinite wires carrying co-flowing currents). The streamlines close to the vortex cores at $x = \pm a, y = 0$ are bound to their respective vortex, which further out, the streamlines merge and encompass the pair; the boundary between the two classes of

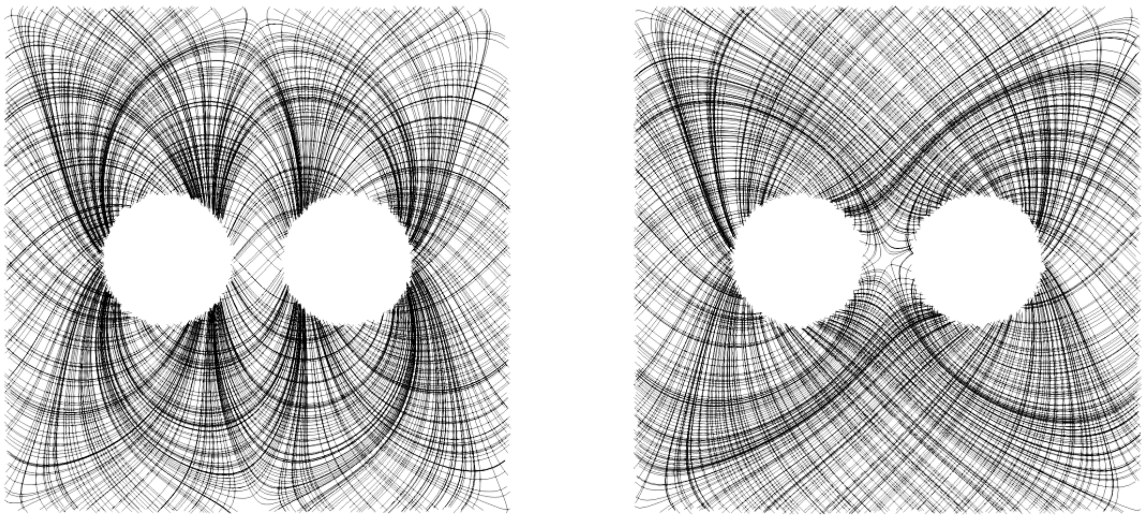


Figure 11: Harmonics as in Fig.5 except for different boundary conditions corresponding to an equal linear combination of the 'vortex'- and 'draining bathtub'-type solutions, showing streamlines and equipotentials (for the left-hand plot derived from the counter-rotating vortex solution, one hole is a source and the other is a sink, hence the existence of streamlines connecting the holes). The choice of boundary condition is basically a choice of angle at which the streamlines meet the boundaries and this angle is the same for all the curves.

streamline is demarcated by a separatrix. As mentioned in the main text, it is straightforward to map the latter solution back to the domain with two removed circles by adding a gradient.

B.3 Generating 2D vector harmonics from functions of a complex variable

In two dimensions, there exists a simple method for generating divergence- and curl-free vector fields. The Cauchy-Riemann equations of complex analysis are, for a holomorphic function $w(z = x + iy) = u + iv$,

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \tag{24}$$

$$\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}. \tag{25}$$

(26)

Note that if the sign of the y-derivative terms are flipped, the equations become respectively the zero-divergence and zero-curl conditions for a 2D vector field with Cartesian components (u, v). It follows that the real and imaginary parts of any antiholomorphic function form the x- and y-components of a divergence- and curl-free two-dimensional vector field in the complex z-plane.

In particular, the single vortex solution is given by $w(\bar{z}) = \frac{i}{\bar{z}}$ and the draining bathtub solution by $w(\bar{z}) = -\frac{1}{\bar{z}}$. The action of Poincaré duality is basically represented by multiplication by i - repeated multiplications switch the solution between anticlockwise vortex, sink, clockwise vortex, drain; more general pure phases $e^{i\varphi}$ give solutions with mixed vortex and drain character as shown in Fig.11 for the two-vortex case. The counter- and (non-circular streamline) co-rotating vortex pair solutions (15, 16) exhibited in the main text are given simply by

$$w(\bar{z}) = \frac{i}{2a}\left(\frac{1}{\bar{z}-a} \mp \frac{1}{\bar{z}+a}\right), \tag{27}$$

in which the factor of $\frac{1}{a}$ means that the solution reduces to a point dipole in the limit $a \to 0$ (this limit represents a double pole in the complex z-plane).

It is apparent that the vorticity corresponds to the residue of $w(\bar{z})$ at its poles; in fact functions with non-zero residues correspond to cohomology representatives of domains with 'holes' localized at poles with nonzero residue. Functions lacking residues in the domain of interest correspond to gradients (equivalently, exact 1-forms).

It is possible to illustrate additional vortex solutions simply by plotting the real and imaginary parts of antiholomorphic functions, for example the function $w(\bar{z}) = i\Gamma(\bar{z})$ represents an infinite sequence of vortices of alternating sign located at $\bar{z} = 0, -1, -2, ...$ and $w(\bar{z}) = i\ \mathrm{cosec}(\bar{z})$ an infinite sequence of vortices of alternating sign at $Re(z) = \pm n\pi$ i.e. along the real axis for integer n; a related solution corresponding to co-rotation is given by $w(\bar{z}) = i\cot(\bar{z})$. A doubly-periodic lattice of vortices can be generated from an elliptic function, e.g. $w(\bar{z}) = i\ \mathrm{sn}(\bar{z}, 2)$ which is a solution to the pendulum equation $w''(z) = -\sin w(z)$. These are shown in Fig.12.

References

[1] D.N. Arnold. *Finite Element Exterior Calculus*. CBMS-NSF regional conference series in applied mathematics **93**, SIAM, 2018.

[2] DefElement. https://defelement.com/elements/index.html, 2022. Accessed: September 2022.

[3] Arnold D.N., Falk R.S., and Winther R. Finite element exterior calculus: from Hodge theory to numerical stability. *Bull. Amer. Math. Soc. (N.S.)* **47** *(2010) no.2, 281-354. MR2594630.*

[4] Periodic table of the finite elements. https://www-users.cse.umn.edu/~arnold/femtable/, 2022. Accessed: September 2022.

[5] P.G. Ciarlet. *The finite element method for elliptic problems*.

[6] Cotter C.J. Private communication, 2022.

[7] Rognes M.E., Kirby R.C., and Logg A. Efficient assembly of H(div) and H(curl) conforming finite elements. *SIAM Journal on Scientific Computing 31, no. 6 (2010): 4130-4151*, 2010.

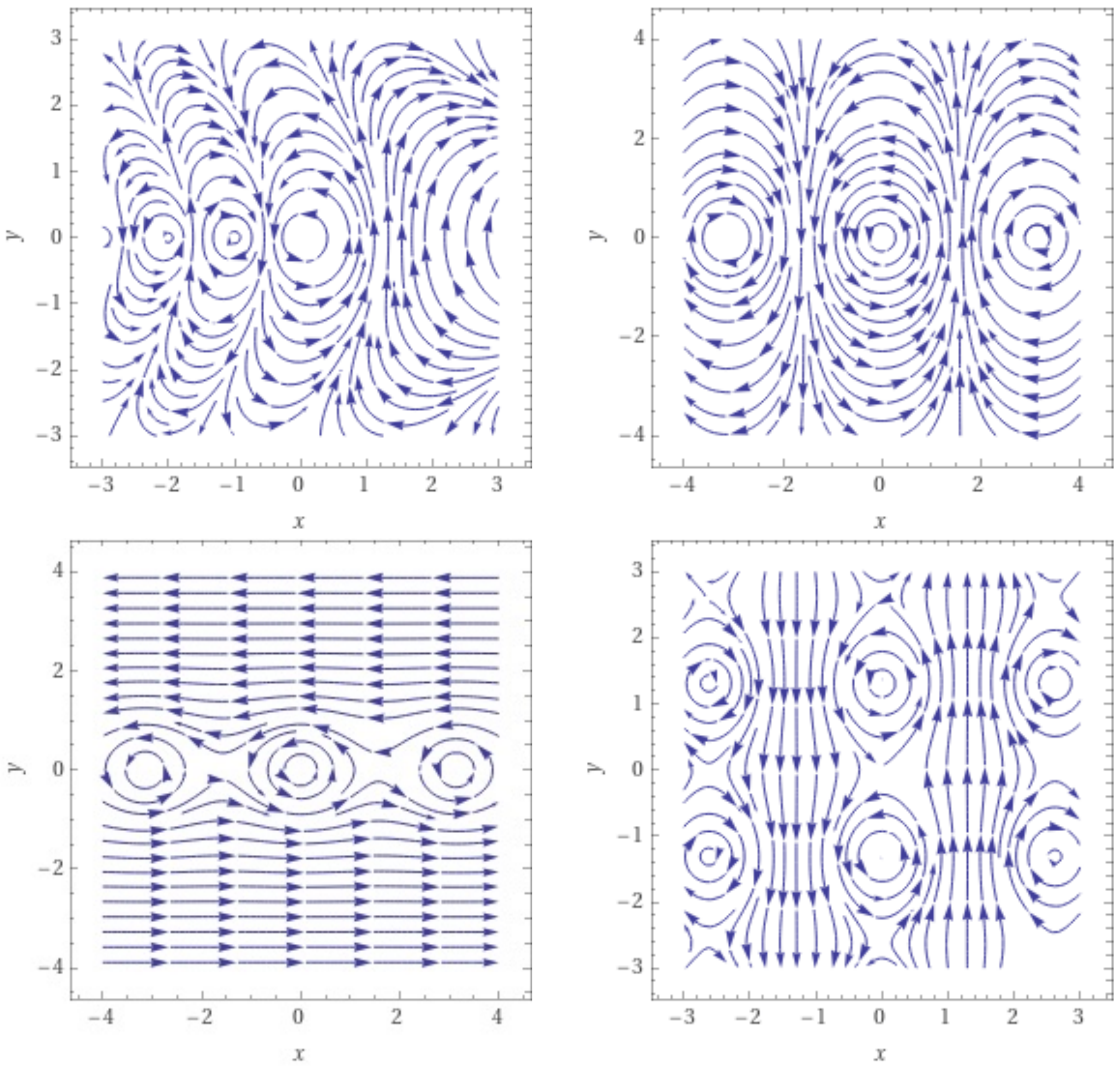


Figure 12: Harmonics generated using the antiholomorphic functions $w(\bar{z}) = i\Gamma(\bar{z})$ (top left), $w(\bar{z}) = i\ \mathrm{cosec}(\bar{z})$ (top right), $w(\bar{z}) = i\ \cot(\bar{z})$ (bottom left), and the Jacobi elliptic function $w(\bar{z}) = i\ \mathrm{sn}(\bar{z}, 2)$ (bottom right) and plotted using *Wolfram Alpha* [27].

[8] Homolya M., Kirby R.C., and Ham D.A. Exposing and exploiting structure: optimal code generation for high-order finite element methods. *submitted to ACM Transactions on Mathematical Software*, 2017.

[9] Florian Rathgeber, David A. Ham, Lawrence Mitchell, Michael Lange, Fabio Luporini, Andrew T. T. Mcrae, Gheorghe-Teodor Bercea, Graham R. Markall, and Paul H. J. Kelly. *Firedrake*: automating the finite element method by composing abstractions. *ACM Trans. Math. Softw., 43(3):24:1–24:27, 2016. URL: http://arxiv.org/abs/1501.01809, arXiv:1501.01809, doi:10.1145/2998441.*, 2015.

[10] Brezzi F. and Bathe K.-J. A discourse on the stability conditions for mixed finite element formulations. *Comput. Methods. Appl. Mech. Eng.* **82** *(1990), 27-57. MR 1077650*, 1990.

[11] Finite element exterior calculus. `https://github.com/ethrefall/Finite-element-exterior-calculus`, 2022. Accessed: September 2022.

[12] Raviart-Thomas finite elements. `https://en.wikipedia.org/wiki/Raviart-Thomas_basis_functions`. Accessed: September 2022.

[13] DefElement H-Curl conforming elements. `https://defelement.com/lists/categories/Hcurl.html`, 2022. Accessed: September 2022.

[14] GitHub discussions thread re homogeneous Dirichlet boundary condition in *Firedrake*. `https://github.com/firedrakeproject/firedrake/discussions/2552`, 2022. Accessed: September 2022.

[15] Scalable Library for Eigenvalue Problem Computations. `https://slepc.upv.es/`. Accessed: September 2022.

[16] Bipolar coordinate system. `https://en.wikipedia.org/wiki/Bipolar_coordinates`. Accessed: September 2022.

[17] H. Lamb. *Hydrodynamics*. Cambridge University Press, 1932.

[18] NESO-Particles code repository. `https://github.com/ExCALIBUR-NEPTUNE/NESO-Particles`. Accessed: September 2022.

[19] Python interface demo for *NekMesh*. `https://github.com/EXCALIBUR-NEPTUNE/nekmesh-python-demo`. Accessed: September 2022.

[20] Sandbox for exploring *NekMesh*. `https://github.com/oparry-ukaea/nekmesh_sandbox`. Accessed: September 2022.

[21] Moxey D. Private communication, 2022.

[22] FEniCS Project. `https://fenicsproject.org/`. Accessed: September 2022.

[23] Portable, Extensible Toolkit for Scientific Computation. `https://petsc.org/release/`. Accessed: September 2022.

[24] E. Threlfall and W. Arter. Finite Element Models: Complementary Activities I. Technical Report CD/EXCALIBUR-FMS/0051-M6.1, UKAEA, 2021. https://github.com/ExCALIBUR-NEPTUNE/Documents/blob/main/reports/ukaea_reports/CD-EXCALIBUR-FMS0051-M6.1.pdf.

[25] Implementation of Rayleigh-Bénard convection in *Firedrake*. https://www.firedrakeproject.org/demos/rayleigh-benard.py.html. Accessed: September 2022.

[26] GitHub Discussions thread re de Rham complex with boundary conditions in *Firedrake*. https://github.com/firedrakeproject/firedrake/discussions/2501, 2022. Accessed: September 2022.

[27] WolframAlpha. www.wolframalpha.com. Accessed: September 2022.