Mixed Finite Element Formulation of the Poisson Equation

Kazi Ahmed and Brian Cornille

May 16, 2017

1 Background

In finite element analysis differential equations are translated to equivalent variational forms. The problem domain is then partitioned and a finite set of functions is used to approximate the Hilbert space of test functions in the variational form. This produces a system of equations that can be solved to produce an approximate solution to the problem statement. In many cases it is preferable to begin with a differential equation that has been reduced as much as possible analytically. The Mixed finite element method (FEM) does not use this *irreducible* form and introduces auxiliary variables into the differential system. This may be done because the auxiliary variables are of physical or scientific interest, or in hopes of producing a better conditioned system of equations.

The goal of this paper is to gather some practical knowledge for implementing a mixed FEM and compare it with its classical or irreducible counterpart. We will consider the Poisson equation with zero boundary condition as our test problem. The differential form is traditionally given by

$$\begin{cases} -\Delta p = f & \text{in } \Omega \\ p|_{\partial\Omega} = 0 \end{cases} \tag{D}$$

To get a mixed FEM the auxiliary variable $\vec{u}$ is introduced and an alternative form of eq. (D) is obtained.

$$\begin{cases} \vec{u} = -\nabla p & \text{in } \Omega \\ \nabla \cdot \vec{u} = f & \text{in } \Omega \\ p|_{\partial\Omega} = 0 \end{cases} \tag{D^*}$$

Equations (D) and (D^*) will be used to construct the variational forms of the Poisson equation that we will present.

1.1 Standard FEM formulation of the Poisson equation

We start with eq. (D), which will lead to what we call the *Standard* formulation. To begin, we multiply the differential equation by a test function q and integrate over the

domain. Using the usual notation of the scalar function inner product on the domain, this gives

$$-\langle \Delta p, q\rangle_\Omega = \langle f, q\rangle_\Omega \tag{1}$$

Once the first term is integrated by parts, we have

$$\langle \nabla p, \nabla q\rangle_\Omega - \int_{\partial\Omega} q\nabla p \cdot \hat{n}\mathrm{d}S = \langle f, q\rangle_\Omega \tag{2}$$

From here, the appropriate Hilbert space, H_0^1, is chosen for p and q. Due to the Dirichlet boundary condition, the second term in the left-hand side of eq. (2) drops out. Thus the Standard variational problem is given by: Look for $p \in H_0^1(\Omega)$ such that

$$\langle \nabla p, \nabla q\rangle_\Omega = \langle f, q\rangle_\Omega \qquad \forall q \in H_0^1(\Omega) \tag{V^{S}}$$

1.2 The Hilbert spaces $H(\nabla\cdot)$ and $H(\nabla\times)$

In order to develop the mixed FEM based on eq. (D^*), we will need to introduce some less frequently used Hilbert spaces. $H(\nabla\cdot)$ is defined for $\Omega \subset \mathbb{R}^n$ as [1]

$$H(\nabla\cdot;\Omega) \coloneqq \left\{\vec{w} : \vec{w} \in \left[L^2(\Omega)\right]^n, \nabla\cdot\vec{w} \in L^2(\Omega)\right\} \tag{3}$$

with the norm

$$\|\vec{w}\|^2_{\nabla\cdot,\Omega} \coloneqq |\vec{w}|^2_{0,\Omega} + |\nabla\cdot\vec{w}|^2_{0,\Omega} \tag{4}$$

Similarly, $H(\nabla\times)$ is defined for $\Omega \subset \mathbb{R}^3$ as [1]

$$H(\nabla\times;\Omega) \coloneqq \left\{\vec{w} : \vec{w} \in \left[L^2(\Omega)\right]^3, \nabla\times\vec{w} \in \left[L^2(\Omega)\right]^3\right\} \tag{5}$$

with the norm

$$\|\vec{w}\|^2_{\nabla\times,\Omega} \coloneqq |\vec{w}|^2_{0,\Omega} + |\nabla\times\vec{w}|^2_{0,\Omega} \tag{6}$$

Although the usual $\nabla\times$ operator is defined for $\mathbb{R}^3$, a similar operator $\mathrm{curl} : \mathbb{R}^2 \to \mathbb{R}$ can be defined for $\Omega \subset \mathbb{R}^2$. It should be noted that $H(\mathrm{curl};\Omega)$ is isomorphic to $H(\nabla\cdot;\Omega)$.

1.3 Mixed FEM formulation of the Poisson equation

We start with eq. (D^*) to produce what we will denote the *Mixed* formulation. The first equation is multiply by the test function $\vec{v}$ and the second by the test function q. Both are integrated over the domain.

$$\begin{cases} \langle \vec{u}, \vec{v}\rangle_\Omega = -\langle \nabla p, \vec{v}\rangle_\Omega \\ \langle \nabla\cdot\vec{u}, q\rangle_\Omega = \langle f, q\rangle_\Omega \end{cases} \tag{7}$$

We then integrate the $\langle \nabla p, \vec{v}\rangle_\Omega$ term by parts and rearrange to achieve some symmetry in the system.

$$\begin{cases} -\langle \vec{u}, \vec{v}\rangle_\Omega + \langle p, \nabla\cdot\vec{v}\rangle_\Omega = \displaystyle\int_{\partial\Omega} p\vec{v}\cdot\hat{n}\mathrm{d}S \\ \langle \nabla\cdot\vec{u}, q\rangle_\Omega = \langle f, q\rangle_\Omega \end{cases} \tag{8}$$

Note that here the Dirichlet boundary condition on p appears as a *natural* boundary condition and a Neumann boundary condition would appear as a *essential* boundary condition on $\vec{v}$. This is opposite of the Standard formulation. Again we will enforce the boundary condition to eliminate the surface integral term. Thus the Mixed variational problem is given by: Look for $\vec{u} \in H(\nabla\cdot;\Omega)$ and $p \in L^2(\Omega)$ such that

$$\begin{cases} -\langle \vec{u}, \vec{v} \rangle_\Omega + \langle p, \nabla \cdot \vec{v} \rangle_\Omega = 0 & \forall \vec{v} \in H(\nabla\cdot;\Omega) \\ \langle \nabla \cdot \vec{u}, q \rangle_\Omega = \langle f, q \rangle_\Omega & \forall q \in L^2(\Omega) \end{cases} \tag{V^{M}}$$

1.4 Dual-Mixed FEM formulation of the Poisson equation

An alternative mixed FEM formulation, that we will refer to as *Dual-Mixed*, can be derived from eq. (7) by instead integrating the $\langle \nabla \cdot u, q \rangle_\Omega$ by parts. In this case you end up with the system

$$\begin{cases} \langle \vec{u}, \vec{v} \rangle_\Omega + \langle \nabla p, \vec{v} \rangle_\Omega = 0 \\ \langle \vec{u}, \nabla q \rangle_\Omega = -\langle f, q \rangle_\Omega - \int_{\partial\Omega} q\vec{u} \cdot \hat{n} \mathrm{d}S \end{cases} \tag{9}$$

The Dual-Mixed variational problem is given by: Look for $\vec{u} \in H(\nabla\times;\Omega)$ and $p \in H_0^1(\Omega)$ such that

$$\begin{cases} \langle \vec{u}, \vec{v} \rangle_\Omega + \langle \nabla p, \vec{v} \rangle_\Omega = 0 & \forall \vec{v} \in H(\nabla\times;\Omega) \\ \langle \vec{u}, \nabla q \rangle_\Omega = -\langle f, q \rangle_\Omega & \forall q \in H_0^1(\Omega) \end{cases} \tag{V^{DM}}$$

2 One-Dimensional Results

In the one-dimensional case we do not have to worry about the spaces $H(\nabla\cdot)$ and $H(\nabla\times)$. The vector $\vec{u}$ can be treated as a scalar so $H(\nabla\cdot) \to H^1$ and $H(\nabla\times) \to L^2$ in Mixed and Dual-Mixed formulations.

2.1 Approximating H^1

Simply put, the approximate expansion of the H^1 space should be C^0 continuous functions. To accomplish this on a mesh that is partitioned into multiple elements requires some degrees of freedom (DoF) to be shared across element boundaries. We accomplished this by expanding using Lagrange cardinal functions based on the Gauss-Legendre-Lobatto (GLL) quadrature nodes, which will be labeled P_k^{GLL}. This choice provides the necessary regularity and achieves spectral convergence with increasing polynomial degree k.

2.2 Approximating L^2

The approximate expansion of the L^2 space only requires piecewise continuous functions. Thus the expansion on each element is independent of other elements. This could allow for any number of local expansions, but we chose the Lagrange cardinal functions based

on the Gauss-Legendre (GL) quadrature nodes, which will be labeled P_k^{GL}. Again, this choice achieves spectral convergence with increasing polynomial degree k.

2.3 1D implementation

The implementation of a 1D solver in C++ utilizes the Eigen linear algebra library to handle matrices and solve the assembled system. We create the quadrature points, reference elements, and mesh in our own routines, then assemble the problem matrix and right-hand-side based on input parameters provided through a JSON [2] input file. The Eigen [3] library can then be used to solve this system efficiently, since it has various capabilities for sparse matrices and solving through decomposition, iterative methods, etc. Python scripts facilitate convergence studies and output visualization.

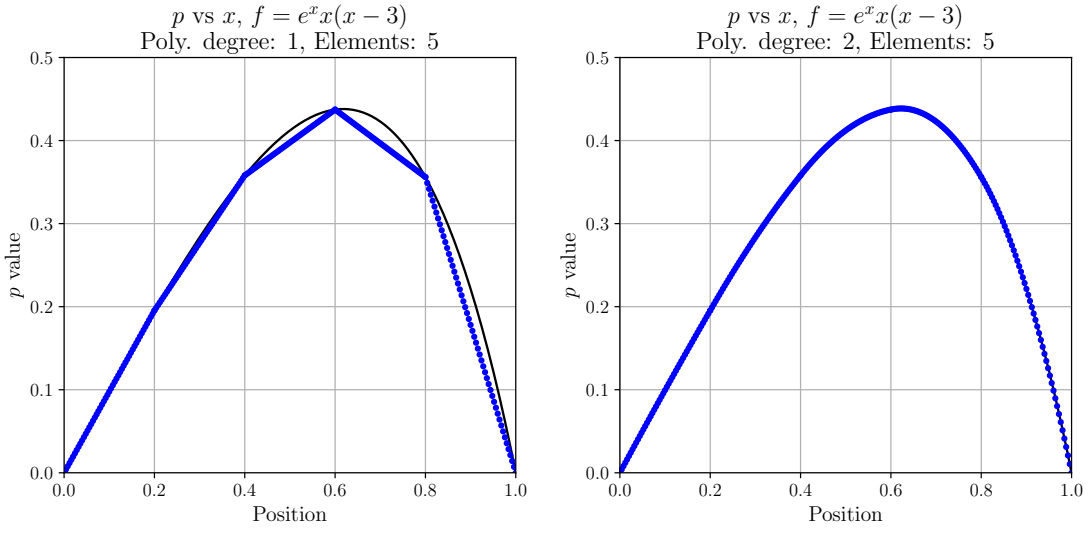
2.4 Standard formulation

In 1D, the Standard formulation uses the approximation space

$$Q_h \coloneqq \left\{ q_h : q_h \in H_0^1(\Omega),\, q_h|_K \in P_k^{GLL}(K) \right\} \tag{10}$$

to find the solution p_h.

Example results from the 1D implementation of the solver for the standard formulation are shown in fig. 1 for elements of 1st and 2nd polynomial degree, and forcing function f. Notice that with the GLL nodes, for the 1D problem we produce the interpolation of the exact solution, which is shown with the black line. The solution is already highly accurate for quadratic basis functions and only 5 elements.



(a) Linear elements

(b) Quadratic elements

Figure 1: P vs. x, Standard Formulation Result

2.5 Mixed formulation

For the Mixed formulation in 1D we need the approximation spaces

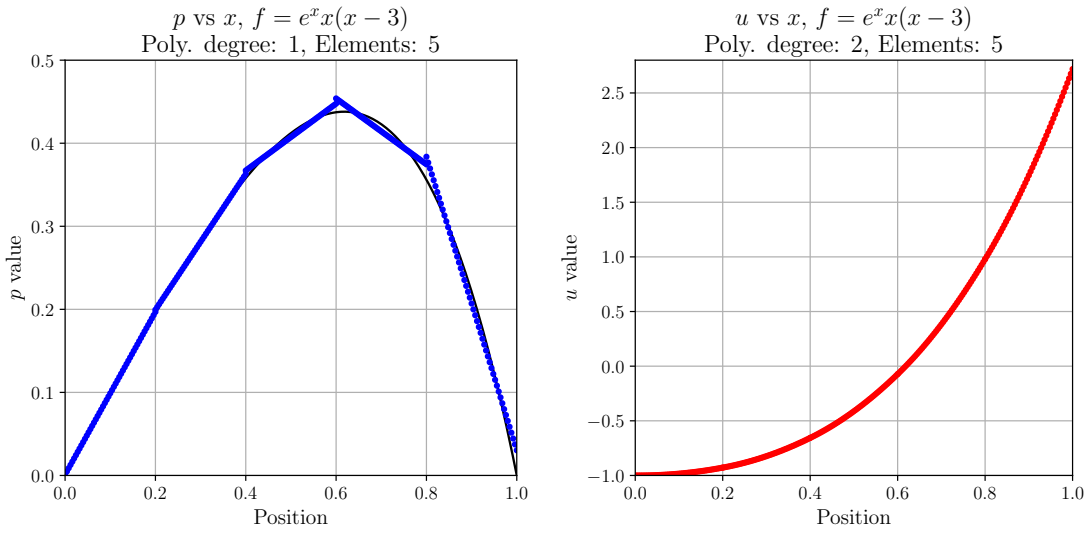
$$W_h^{(k)} := \left\{ w_h : w_h \in L^2(\Omega),\, w_h|_K \in P_k^{GL} \right\} \tag{11}$$

and

$$Q_h^{(k)} := \left\{ q_h : q_h \in H^1(\Omega),\, q_h|_K \in P_k^{GLL}(K) \right\} \tag{12}$$

We look for $u_h \in Q_h^{(k+1)}$ and $p_h \in W_h^{(k)}$ in this case.

The 1D solver produces the p and corresponding u results plotted in fig. 2, for a piecewise-linear p and the same forcing function used to illustrate the standard results. Note that u here is of polynomial degree 1 higher than p. The result for p still yields the interpolation of the exact solution, but on the GL nodes instead of the GLL nodes, so p is only piecewise continuous.



(a) p: linear elements on GL nodes

(b) u: quadratic elements on GLL nodes

Figure 2: P vs. x and u vs. x, Mixed Formulation Result

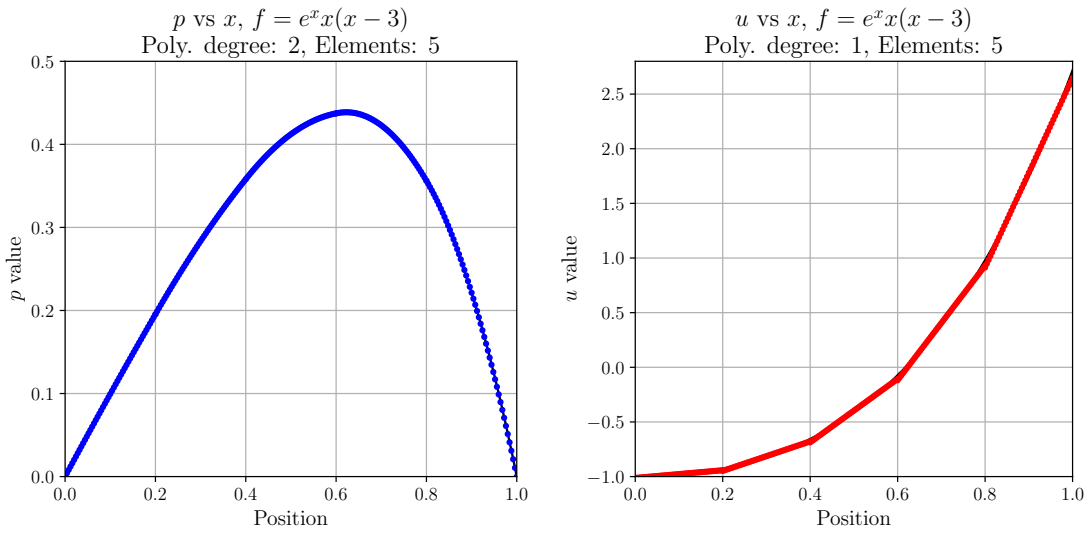
2.6 Dual-Mixed formulation

In 1D we have nearly the same approximation spaces as the Mixed formulation except eq. (12) is instead

$$Q_h^{(k)} := \left\{ q_h : q_h \in H_0^1(\Omega),\, q_h|_K \in P_k^{GLL}(K) \right\} \tag{13}$$

and we look for $u_h \in W_h^{(k-1)}$ and $p_h \in Q_h^{(k)}$.

The 1D solver produces the p and corresponding u results plotted in fig. 3, for a piecewise-quadratic p and the same forcing function used to illustrate the standard results. Note that u here is of polynomial degree 1 lower than p. The result for p yields the interpolation of the exact solution on the GLL nodes, and is identical to the result from the standard formulation. In this formulation u is only piecewise continuous.



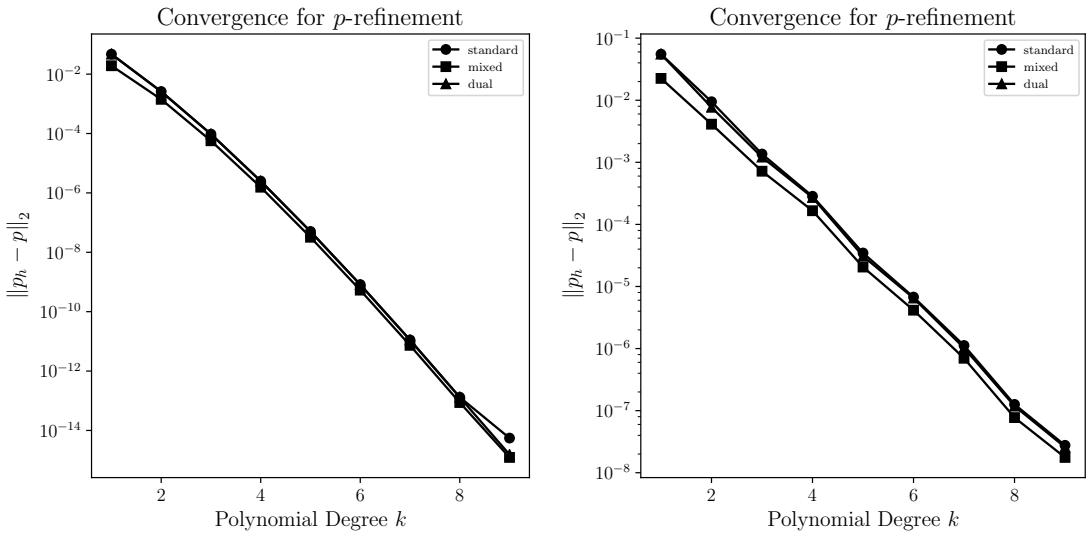
(a) p: quadratic elements on GLL nodes

(b) u: linear elements on GL nodes

Figure 3: P vs. x and u vs. x, Dual-Mixed Formulation Result

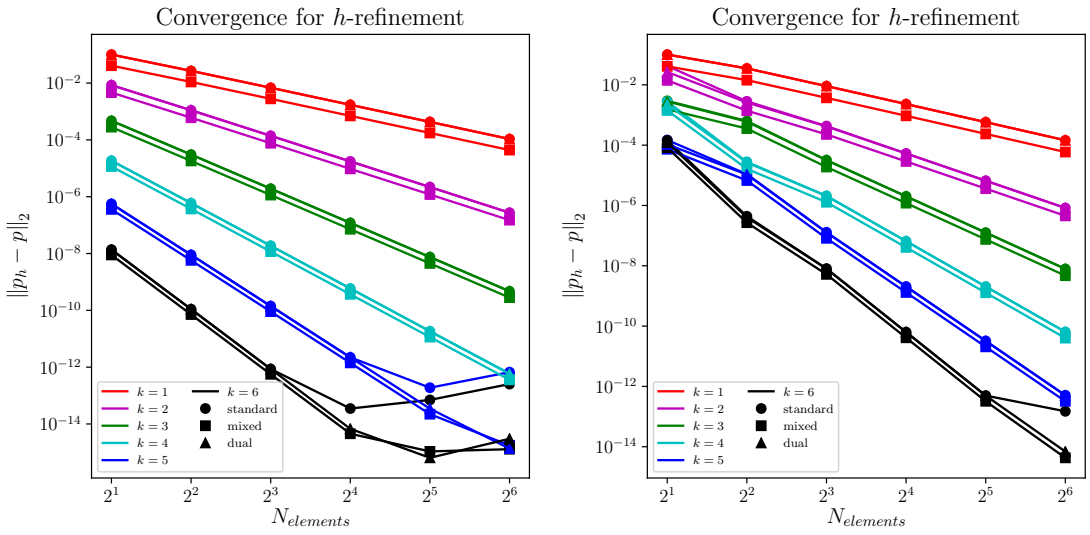
2.7 Convergence

To study convergence, we investigate both p-refinement and h-refinement, by sweeping a range of polynomial degree and a range of number of elements, respectively, and evaluating 2-norm error. For p-refinement, we compare identical polynomial degree for p between the methods, which actually means that the system for the mixed formulation is larger, since u in that formulation is of higher degree. Convergence is studied on both a "straight" mesh, which can also include linear transforms, and a "curved" mesh, which implies non-linear transform. From the plots, we see expected exponential convergence for p-refinement, and optimal $k+1$ convergence rate for h-refinement (we know standard is optimal, and the other formulations follow).



(a) Convergence on a "straight" mesh (b) Convergence on a "curved" mesh

Figure 4: Error vs. polynomial degree of p, for a 3 element mesh



(a) Convergence on a "straight" mesh (b) Convergence on a "curved" mesh

Figure 5: Error vs. number of elements, for varying polynomial degree

3 Two-Dimensional Results

For the 2D mixed FEM we will need to construct approximations for $H(\nabla\cdot)$ and $H(\text{curl})$. In addition to their special regularity needs, functions in these spaces also require complicated change of variable transformation. Both of these needed to be carefully considered to produce a proper implementation.

3.1 Approximating $H(\nabla\cdot)$

As shown in eq. (3), functions in $H(\nabla\cdot)$ are vector-valued and must be continuous in their normal components across element boundaries. Thus the normal pointing basis functions must have DoF associated with the boundaries between neighboring elements. There are several choices of function spaces that could satisfy these conditions. We chose to use Raviart-Thomas elements on quadrilaterals for our two-dimensional code. Specifically, the x-component basis functions are $P_{k+1}^{GLL} \times P_k^{GL}$ while the y-component basis function are $P_k^{GL} \times P_{k+1}^{GLL}$. We will refer to this space as $\mathcal{RT}_{[k]} \coloneqq \left(P_{k+1}^{GLL} \times P_k^{GL}\right) \times \left(P_k^{GL} \times P_{k+1}^{GLL}\right)$. As in the one-dimensional case, this basis choice should produce spectral convergence with polynomial degree k. Additionally, this space has been defined such that for $\vec{w} \in \mathcal{RT}_{[k]}$ then $\nabla \cdot \vec{w} \in P_k \times P_k$.

3.2 Approximating $H(\text{curl})$

In $\mathbb{R}^2$ we do not get to used $H(\nabla\times)$, but instead $H(\text{curl})$. Similar to $H(\nabla\cdot)$, these functions are vector-valued, although they require continuity in the tangential component across element borders. We will use a very similar space to $\mathcal{RT}_{[k]}$ based on the Nédélec elements of the first kind. This space is $\mathcal{N}_{[k]} \coloneqq \left(P_k^{GL} \times P_{k+1}^{GLL}\right) \times \left(P_{k+1}^{GLL} \times P_k^{GL}\right)$.

3.3 2D implementation

The implementation of the 2D solver is much like what was done for 1D. Conveniently, the same basis nodes that are generated for 1D GL and GLL can be used, since the 2D mesh, depending on the formulation, will be a grid of some combination of these nodes in the x and y directions. The real effort in the 2D implementation lies in the meshing, and having a useful system of indexing all the nodes so that other parts of the code, like the main class or the plotting utility, can easily utilize the data.

3.4 Standard formulation

Using $Q_h \coloneqq \left\{q_h : q_h \in H_0^1(\Omega),\, q_h|_K \in \left(P_k^{GLL} \times P_k^{GLL}\right)(K)\right\}$, the discrete statement of eq. (V^{S}) becomes: Look for $p_h \in Q_h$ such that

$$\langle \nabla p_h, \nabla q_h \rangle_\Omega = \langle f, q_h \rangle_\Omega \qquad \forall q_h \in Q_h \tag{V_h^{S}}$$

For implementation we use a reference element $\hat{K}$ and utilize a mapping $\mathcal{F}(\hat{x})$ where $K = \mathcal{F}\left(\hat{K}\right)$, $\mathcal{DF}\left(\hat{x}\right)$ is the Jacobian matrix, and $\mathcal{J}\left(\hat{x}\right)$ is the Jacobian. It then is useful

to integrate on the reference element. The integrals in eq. (V_h^{S}) become [1]

$$\int_K \nabla p_h \cdot \nabla q_h \mathrm{d}x = \int_{\hat{K}} \left[\mathcal{DF}(\hat{x})\right]^{-T} \nabla \hat{p}_h \cdot \left[\mathcal{DF}(\hat{x})\right]^{-T} \nabla \hat{q}_h \mathcal{J}(\hat{x}) \mathrm{d}\hat{x} \tag{14}$$

and

$$\int_K f q_h \mathrm{d}x = \int_{\hat{K}} \hat{f} \hat{q}_h \mathcal{J}(\hat{x}) \mathrm{d}\hat{x} \tag{15}$$

3.5 Mixed formulation

For the 2D Mixed formulation we need the approximation spaces

$$W_h^{(k)} \coloneqq \left\{ \vec{w}_h : \vec{w}_h \in H(\nabla\cdot;\Omega),\, \vec{w}_h|_K \in \mathcal{RT}_{[k]}(K) \right\} \tag{16}$$

and

$$Q_h^{(k)} \coloneqq \left\{ q_h : q_h \in L^2(\Omega),\, q_h|_K \in \left(P_k^{GL} \times P_k^{GL}\right)(K) \right\} \tag{17}$$

The discrete statement of eq. (V^{M}) is then: Look for $u_h \in W_h^{(k+1)}$ and $p_h \in Q_h^{(k)}$ such that

$$\begin{cases} -\langle \vec{u}_h, \vec{v}_h \rangle_\Omega + \langle p_h, \nabla \cdot \vec{v}_h \rangle_\Omega = 0 & \forall \vec{v}_h \in W_h^{(k+1)} \\ \langle \nabla \cdot \vec{u}_h, q_h \rangle_\Omega = \langle f, q_h \rangle_\Omega & \forall q \in Q_h^{(k)} \end{cases} \tag{V_h^{M}}$$

Since the reference element $\hat{K}$ is used our integrals in eq. (V_h^{M}) must again be transformed. [1]

$$\int_K \vec{u}_h \cdot \vec{v}_h \mathrm{d}x = \int_{\hat{K}} \mathcal{DF}(\hat{x}) \hat{\vec{u}}_h \cdot \mathcal{DF}(\hat{x}) \hat{\vec{v}}_h \frac{1}{\mathcal{J}(\hat{x})} \mathrm{d}\hat{x} \tag{18}$$

and

$$\int_K p_h \nabla \cdot \vec{v}_h \mathrm{d}x = \int_{\hat{K}} \hat{p}_h \nabla \cdot \hat{\vec{v}}_h \mathrm{d}\hat{x} \tag{19}$$

are the new types of integrals not already addressed in section 3.4. Notice that eq. (19) contains no information about the change of variables.

3.6 Dual-Mixed formulation

The Dual-Mixed formulation in 2D utilizes the approximation spaces

$$W_h^{(k)} \coloneqq \left\{ \vec{w}_h : \vec{w}_h \in H(\mathrm{curl};\Omega),\, \vec{w}_h|_K \in \mathcal{N}_{[k]}(K) \right\} \tag{20}$$

and

$$Q_h^{(k)} \coloneqq \left\{ q_h : q_h \in H_0^1(\Omega),\, q_h|_K \in \left(P_k^{GLL} \times P_k^{GLL}\right)(K) \right\} \tag{21}$$

The discrete statement of eq. (V^{DM}) is then: Look for $u_h \in W_h^{(k-1)}$ and $p_h \in Q_h^{(k)}$ such that

$$\begin{cases} \langle \vec{u}_h, \vec{v}_h \rangle_\Omega + \langle \nabla p_h, \vec{v}_h \rangle_\Omega = 0 & \forall \vec{v}_h \in W_h^{(k-1)} \\ \langle \vec{u}_h, \nabla q_h \rangle_\Omega = -\langle f, q_h \rangle_\Omega & \forall q \in Q_h^{(k)} \end{cases} \tag{V_h^{DM}}$$

Here we get the new reference element integration rules [1]

$$\int_K \vec{u}_h \cdot \vec{v}_h \mathrm{d}x = \int_{\hat{K}} \left[\mathcal{DF}(\hat{x})\right]^{-T} \hat{\vec{u}}_h \cdot \left[\mathcal{DF}(\hat{x})\right]^{-T} \hat{\vec{v}}_h \mathcal{J}(\hat{x}) \mathrm{d}\hat{x} \tag{22}$$

and

$$\int_K \nabla p_h \cdot \vec{v}_h \mathrm{d}x = \int_{\hat{K}} \nabla \hat{p}_h \cdot \hat{\vec{v}}_h \mathrm{d}\hat{x} \tag{23}$$

Again, we have eq. (23) that has no term from the variable transformation.

3.7 Visualization of results

For brevity, and because the methods produce very similar results, only the plots for the Dual-Mixed formulation are shown here. Furthermore, since the results for a "straight" mesh look quite like the analytical solution, we show the p and u results for a curved mesh. Consider the following forcing function,

$$f = 2\pi^2 \sin(\pi x) \sin(\pi y) \tag{24}$$

with its exact solution

$$p = \sin(\pi x) \sin(\pi y) \tag{25}$$

which is plotted in fig. 6. Compare this to our calculated result using the dual-mixed formulation, shown in fig. 7, for p piecewise-cubic and a 4x4 grid of cells. This would normally appear identical to the exact solution, so results created with a curved mesh are plotted instead to illustrate an example of curvature. Notice from the elevation/color of the plot that the p value computed is correct for the curved mesh, though the grid lines in fig. 7 illustrate the imposed curvature on the system.

Since u is the gradient of p, it should appear as the outward normal of the surface. Indeed, the vector map in fig. 8 shows this result. The asymmetry is a result of the mesh curvature and how the points are sampled, so the arrows appear more dense in some areas and less dense in others. However, their relative magnitudes and directions are correct for their locations. Lastly, since pretty pictures are fun and exciting, we provide similar plots for p and u on a straight mesh but on a larger domain, in fig. 9 and fig. 10!

3.8 2D Convergence

A similar p-refinement and h-refinement study is performed for the 2D solver, with results plotted in fig. 11 and fig. 12. Note again that the mixed method has consistently "better" error for the same polynomial degree of p, though this requires a larger matrix because the corresponding polynomial degree of u is larger. Note similar behavior as described for the 1D convergence study: exponential convergence for p-refinement, and $k+1$ convergence rate for h-refinement. The horizontal axis on the h-refinement plots is number of elements per dimension, so the actual range is from 4 cells to 256 cells. We stop here because of the high cost of increasing the number of cells.

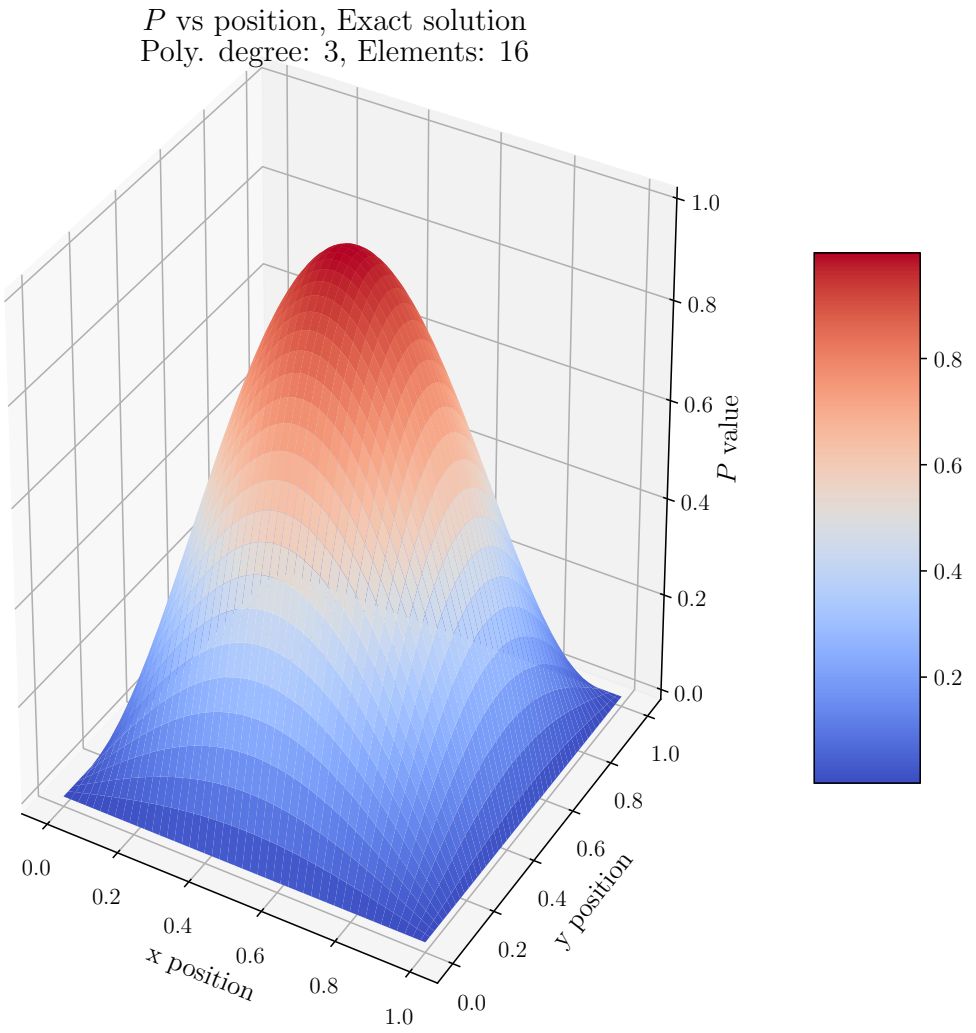


Figure 6: Exact solution $p = \sin(\pi x)\sin(\pi y)$ for $f = 2\pi^2 \sin(\pi x)\sin(\pi y)$

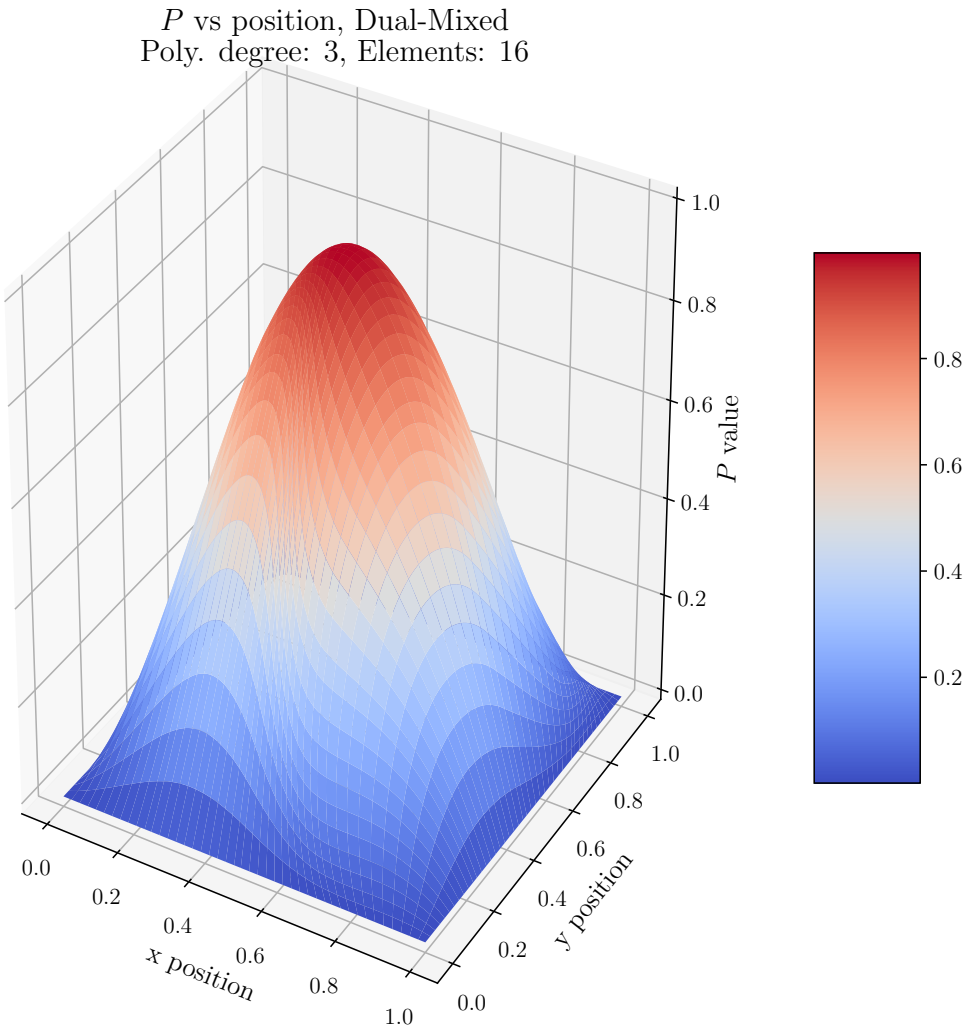


Figure 7: Calculated result for p using dual-mixed formulation with a curved mesh

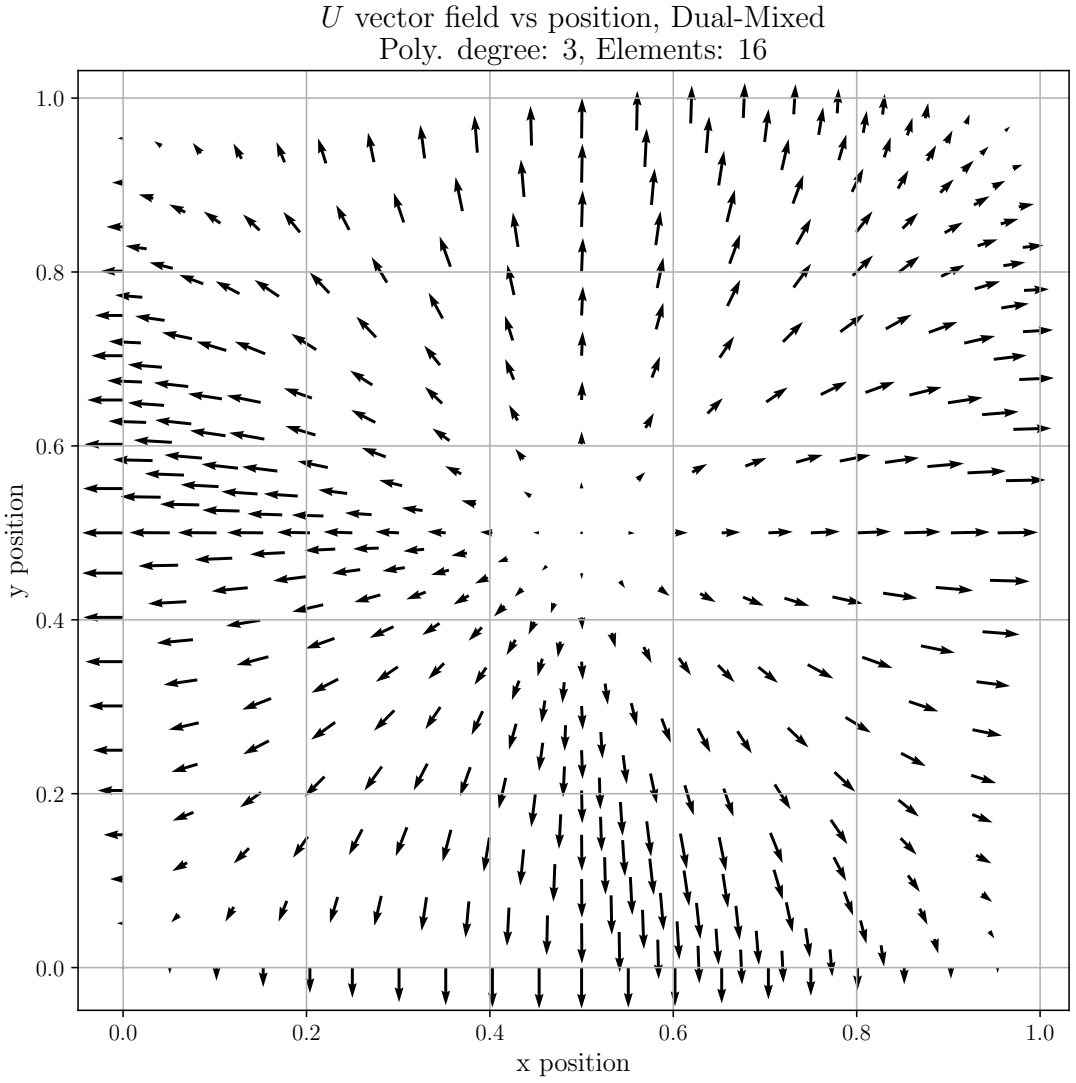


Figure 8: Calculated result for u using dual-mixed formulation with a curved mesh

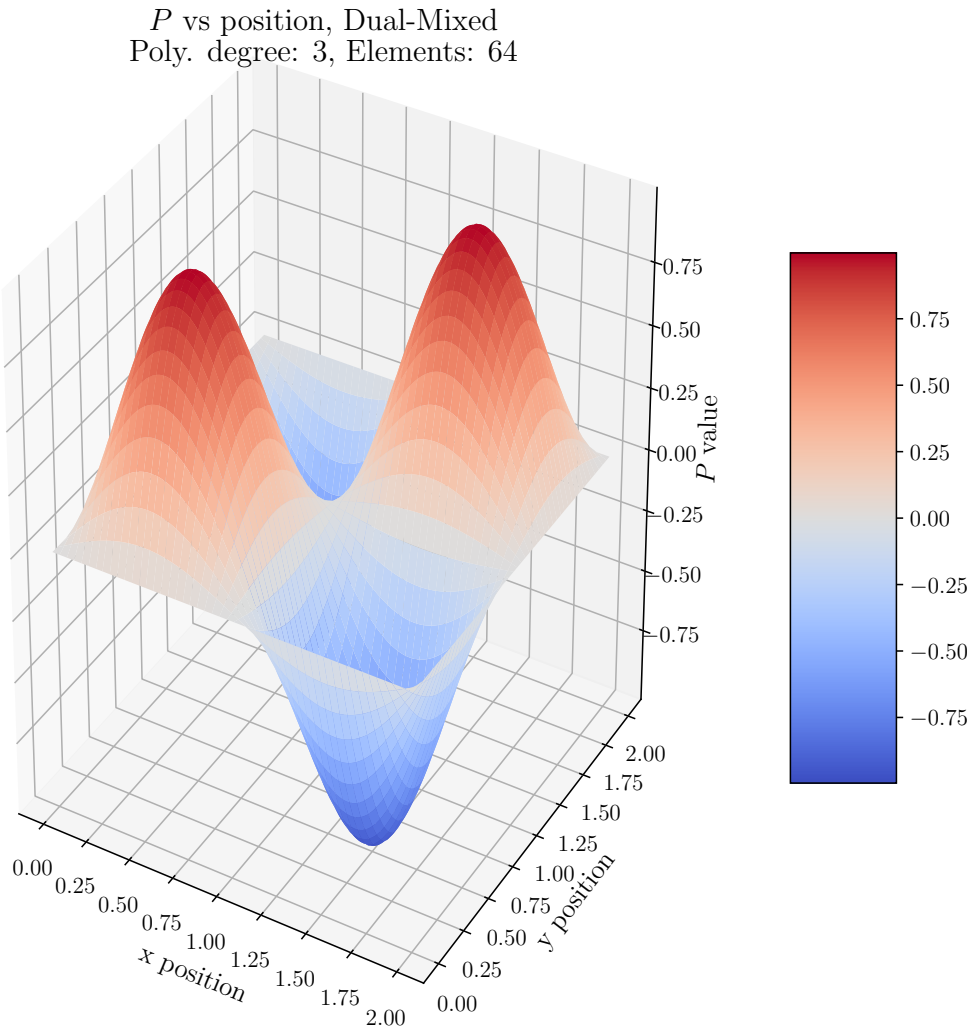


Figure 9: Calculated result for p using dual-mixed formulation, expanded domain

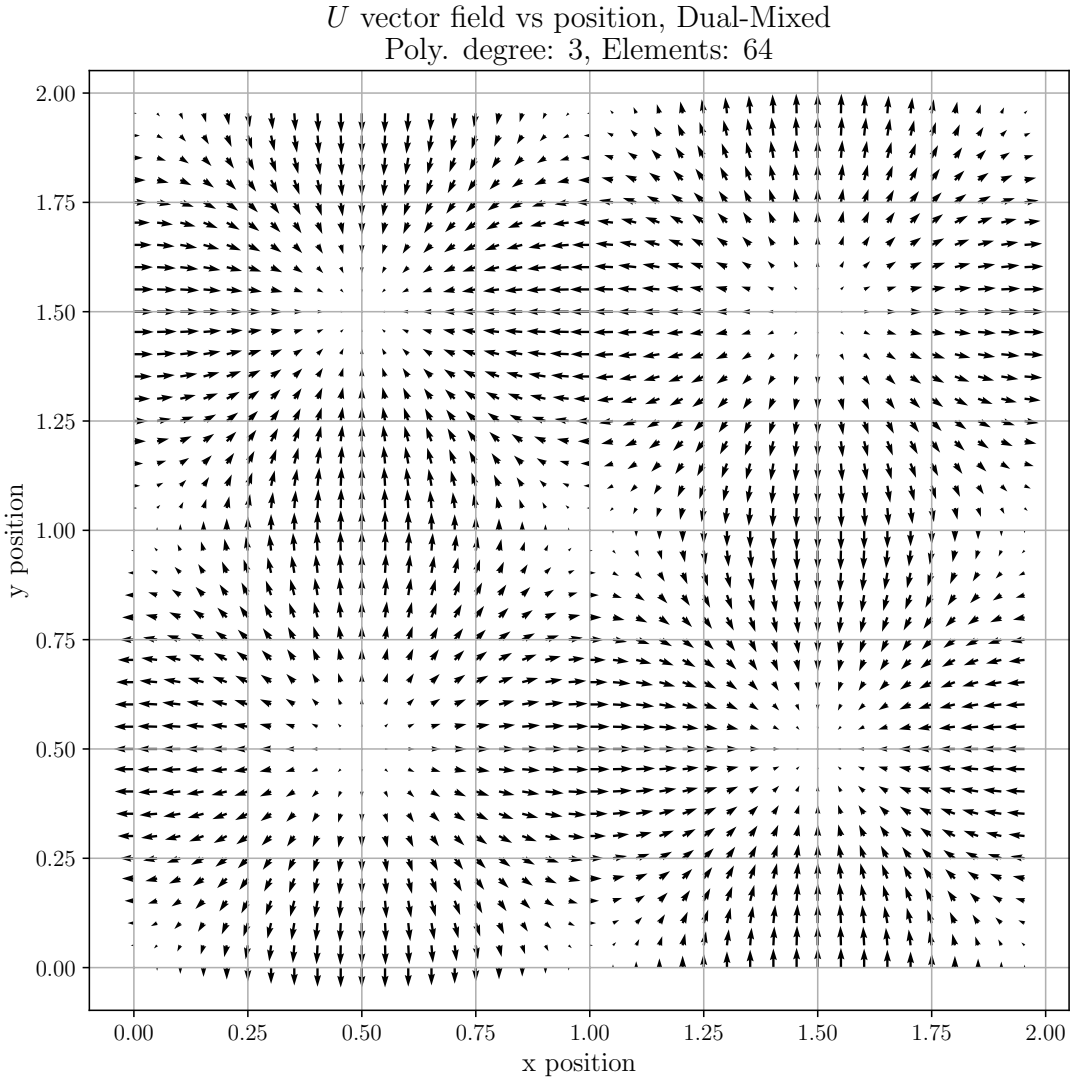
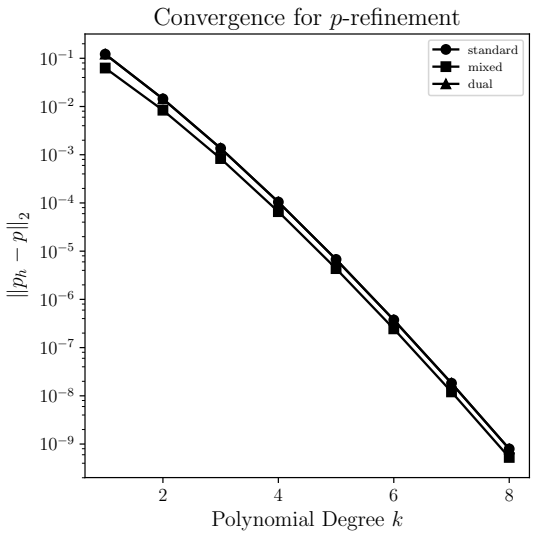
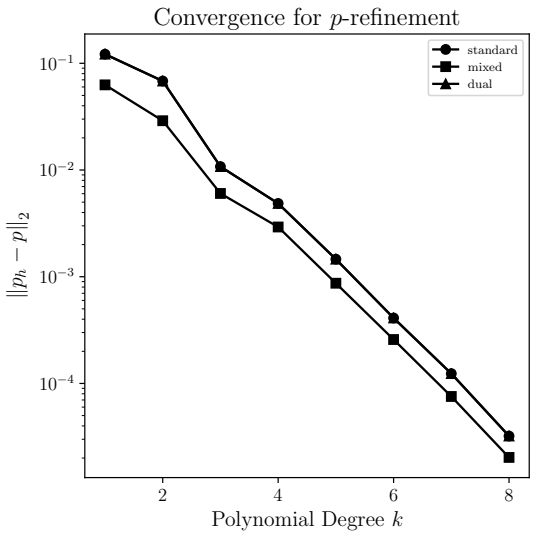


Figure 10: Calculated result for u using dual-mixed formulation, expanded domain

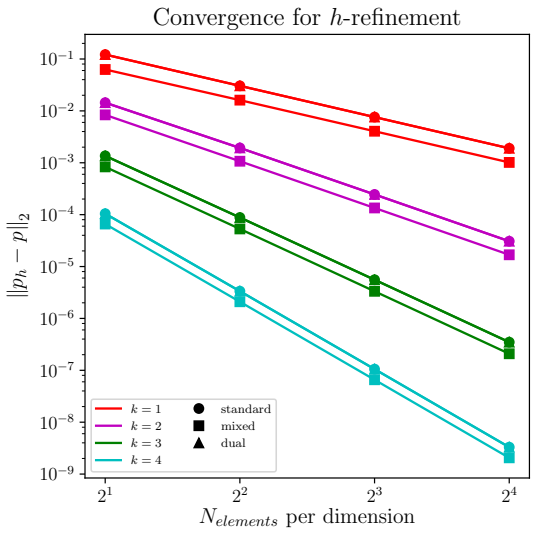


(a) Convergence on a "straight" mesh

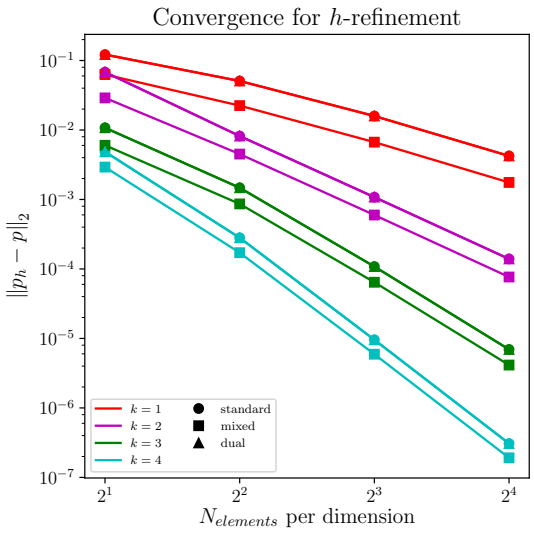


(b) Convergence on a "curved" mesh

Figure 11: Error vs. polynomial degree of p, for a 4 cell mesh



(a) Convergence on a "straight" mesh



(b) Convergence on a "curved" mesh

Figure 12: Error vs. number of elements, for varying polynomial degree

4 Conclusions

We have implemented codes to study the Standard, Mixed, and Dual-Mixed formulations of the Poisson equation in 1D and 2D. This code can be found at `github.com/bcornille/FESpaces`. Through convergence analysis we have demonstrated that our implementation achieves optimal convergence even when our mesh is distorted. We also see that there can be some improvement in accuracy when a mixed FEM is used with the Poisson equation. This can be especially true when the mesh is highly distorted compared to the resolved spatial scale.

While the Poisson equation was a good test system for developing an understanding of mixed FEM, our experience has shown that since it is a relatively simple problem there is not sufficient benefit in terms of accuracy of p to warrant the large increase in system size of a mixed FEM. The main benefit for using mixed FEM for such a problem is the simultaneous generation of the u result, if this variable is of scientific interest. This might especially be important for a tightly coupled system where the u should be part of the matrix, instead of calculated afterwards from the p result. Lastly, the finite element approximation spaces for $H(\nabla\cdot)$ and $H(\nabla\times)$ that we have worked with have greater applicability in other settings, such as Maxwell's equations.

Acknowledgements

Our code was built using two existing libraries. The first is JSON for Modern C++ [2]. Leveraging this library to use the JSON file format for both input and output was a great time-saver. The second library that we utilized was Eigen [3]. Eigen is a linear algebra library that allows for very expressive programming and contains a variety of powerful dense and sparse linear algebra solvers that we used to great effect.

References

(1) Boffi, D.; Brezzi, F.; Fortin, M., *Mixed finite element methods and applications*; Springer Series in Computational Mathematics, Vol. 44; Springer: 2013.

(2) Lohmann, N. JSON for Modern C++., https://github.com/nlohmann/json, 2013.

(3) Guennebaud, G.; Jacob, B., et al. Eigen v3., http://eigen.tuxfamily.org, 2010.