Final Project : The Adaptive Finite Element Method

Manmeet Bhabra

December 12, 2018

1 Introduction

Typically numerical solutions to complex problems are very difficult to predict. A user will generally not know in advance where the dynamics is complicated and, if the mesh used does not have a high enough resolution, the overall solution quality suffers. To overcome this, a naive idea is to simply use a very fine mesh from the start. However, with only limited computational power at our disposal, such an approach is impractical in most engineering applications.

To resolve this issue, an additional approach may be used: Adaptive methods. Adaptive methods work by attempting to estimate the numerical error and then automatically refine or coarsen the mesh in regions of the domain. In this way, the solver efficiently allocates computational resources by making the mesh more fine only where it needs to be while making it coarse where only a few elements suffice in resolving the solution. Several approaches exist, in the finite element framework, to perform this adaptation. The first is h-adaptation, in which elements in the mesh are split up (to refine) or combined (to coarsen) Thus, as the number of elements over a given region increase, the solution is able to be more accurately resolved. The second approach is p-adaptation, where the order of the basis functions used to approximate the solution is varied. Finally, recent years have seen increased research into hp-adaptive methods, where both the mesh size and polynomial order may be adjusted [1]. In this work, we investigate the h-adaptive finite element method. As test cases, the Poisson equation is studied using manufactured solutions.

2 Finite Element Discretization

We begin, in this section, outlining the discretization of the physical equations using the finite element method. The cases studied involved the Poisson equation with Dirichlet boundary conditions and can be written as:

$$\nabla^2 u = f(x,y) \quad on\ \Omega, \tag{1}$$

$$u = g_D \quad on\ \partial\Omega. \tag{2}$$

We start by first deriving the weak form. Consider a test function v. After multiplying both sides of the differential equation by the test function and integrating by parts, the variational form is derived and is given by

$$
\begin{aligned}
a(u,v) &= b(v), \quad v \in H^1(\Omega) \\
a(u,v) &= \int_\Omega \nabla u \cdot \nabla v \; d\Omega \\
b(v) &= \int_\Omega -f(x,y) \cdot v \; d\Omega.
\end{aligned} \tag{3}
$$

The space of functions satisfied by the test functions and solution can be seen to be given by

$$v \in \mathcal{X}_0 = \{v, v \in H^1(\Omega), v|_{\partial\Omega} = 0\} \tag{4}$$
$$u \in \mathcal{X}_{BC} = \{u, u \in H^1(\Omega), u|_{\partial\Omega} = g_D\} \tag{5}$$

We now consider searching for the finite element solution. To do this, the finite dimensional subspaces within which the numerical solution is searched for must be specified. These subspaces are given as:

$$v \in \mathcal{X}_0^h \subset \mathcal{X}_0 \tag{6}$$
$$u \in \mathcal{X}_{BC}^h \subset \mathcal{X}_{BC} \tag{7}$$
$$\tag{8}$$

For the Galerkin method, we require the finite dimensional subspaces to be identical, which can be satisfied if $\mathcal{X}_0^h$ is the homogeneous version of $\mathcal{X}_{BC}^h$. Moreover, if all basis functions that are non-zero on the boundary are given as $\psi_{i,BC}$, then the finite dimensional space for the solution may then be reformulated as $\mathcal{X}_{BC}^h = \{\psi_{i,BC} + \phi, \; \phi \in X_0^h, \; \psi_{i,BC}|_{\partial\Omega} \neq 0\}$.

For the remaining portion of the derivation we restrict ourselves to the case of nodal basis functions which have been used for all cases. The approximate numerical solution, u^h, in this case can be represented as

$$u^h = \sum_{i=1}^{N_{BC}} g_{D,i} \cdot \psi_{i,BC} + \sum_{i=1}^{N_{DOF}} u_i^h \cdot \psi_i \tag{9}$$

where N_{BC} corresponds to the number of nodal basis functions that are non-zero on the boundary, $g_{D,i}$ are the specified values of the Dirichlet boundary condition at these nodes, N_{DOF} are the number of basis functions that vanish at the boundary (which is equivalent to the number of degrees of freedom), and ψ_i are the these basis functions that are only non-zero in the interior of the domain. Using this formulation for the numerical solution, the Galerkin approximation is then given as

$$a(u^h, \psi_j) = b(\psi_j) \quad \forall j = 1, ..., N_{DOF}. \tag{10}$$

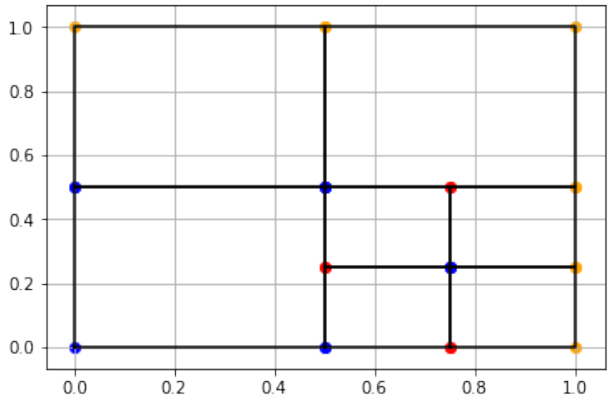

Figure 1: A quadrilateral mesh with two hanging nodes at $(x, y) = (0.75, 0.5)$ *and* $(x, y) = (0.5, 0.25)$.

Substituting the expansion for u^h, we obtain:

$$\begin{aligned} a(\sum_{i=1}^{N_{BC}} g_{D,i} \cdot \psi_{i,BC} + \sum_{i=1}^{N_{DOF}} u_i^h \cdot \psi_i, \psi_j) &= b(\psi_j) \quad \forall j = 1, ..., N_{DOF} \\ \sum_{i=1}^{N_{DOF}} u_i^h \cdot a(\psi_i, \psi_j) &= b(\psi_j) - \sum_{i=1}^{N_{BC}} g_{D,i} \cdot a(\psi_{i,BC}, \psi_j) \quad \forall j = 1, ..., N_{DOF} \end{aligned} \tag{11}$$

Equation (11) forms the system of equations to determine the values of the numerical solution at the degrees of freedom (u_i^h).

2.1 Triangular Elements

To allow for efficient refining of the mesh, a choice had to be made between using quadrilateral or triangular elements. After some initial investigation, it was determined that triangles offered the most versatility on this front. This was due to the presence of what are known as hanging nodes on quadrilateral meshes. These nodes result when a single quadrilateral element has as its neighbors, on one of its edges, multiple quadrilateral elements (this can be seen in Figure (1)). The treatment of hanging nodes is non trivial as the value at each node must be constrained in order to retain continuity of the global solution. Moreover, the nodal basis functions must also be modified to ensure that they too are not discontinuous (so that they remain in $H^1(\Omega)$). Triangle meshes avoid these complexities as they allow for refinement strategies that disallow the generation of hanging nodes and do not need extra treatment of the basis functions to retain their continuity.

2.2 Transformation Mapping

To solve for the numerical solution, all computation was completed by mapping the equations onto a common reference element on a reference domain spanned by coordinates ξ and η as shown in Figure (2).

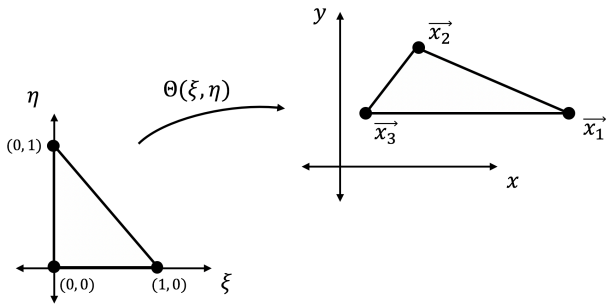

Figure 2: A scehmatic depicting the mapping of the reference triangle onto the physical domain.

To begin, first the nodal basis functions (used to resolve the numerical solution) were defined on the triangle reference element. These functions can be found (for the linear case) to be given as

$$\begin{aligned}\hat{\psi}_1(\xi,\eta) &= 1-\xi-\eta \\ \hat{\psi}_2(\xi,\eta) &= \xi \\ \hat{\psi}_3(\xi,\eta) &= \eta\end{aligned} \tag{12}$$

where we have used the hat notation to highlight that these functions are defined on the reference domain.

Using these basis functions, a mapping from the reference to the physical domain is possible for each element. If an arbitrary element has vertices given as $\vec{x}_1$, $\vec{x}_2$ and $\vec{x}_3$, the transformation from the reference to physical domain is given by

$$\begin{bmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{bmatrix} = \vec{x}(\xi,\eta) = \sum_{i=1}^{3} \hat{\psi}_i(\xi,\eta)\cdot\vec{x}_i. \tag{13}$$

A schematic showing how this transformation would be applied to an arbitrary element on the mesh is shown in Figure (2).

The transformation now allows the ability to reformulate all integrals on the physical domain in equation (3) onto the reference domain. First, all partial derivatives on the physical domain can be expressed as

$$\frac{\partial f}{\partial x} = \frac{1}{J}\cdot\left[\frac{\partial y}{\partial \eta}\frac{\partial f}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial f}{\partial \eta}\right], \tag{14}$$

$$\frac{\partial f}{\partial y} = \frac{1}{J}\cdot\left[-\frac{\partial x}{\partial \eta}\frac{\partial f}{\partial \xi} + \frac{\partial x}{\partial \xi}\frac{\partial f}{\partial \eta}\right], \tag{15}$$

where J is the Jacobian and is given by

$$J = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \eta}. \tag{16}$$

All terms of the form $\frac{\partial x_i}{\partial \xi_j}$ are known as metric terms and can be computed using equation (13). To map the bilinear form to the reference domain we first split the integral into a sum of integrals over each element and then map each integral onto the reference domain (Ω_R). In doing so, we obtain for the bilinear form with basis functions ψ_i and ψ_j as inputs the following:

$$\begin{aligned} a(\psi_i, \psi_j) = \sum_{k=1}^{N_{elements}} \int_{\Omega_R} \left[\left[\frac{\partial y}{\partial \eta}\frac{\partial \hat{\psi}_i}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial \hat{\psi}_i}{\partial \eta} \right] \cdot \left[\frac{\partial y}{\partial \eta}\frac{\partial \hat{\psi}_j}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial \hat{\psi}_j}{\partial \eta} \right] + \right. \\ \left. \left[-\frac{\partial x}{\partial \eta}\frac{\partial \hat{\psi}_i}{\partial \xi} + \frac{\partial x}{\partial \xi}\frac{\partial \hat{\psi}_i}{\partial \eta} \right] \cdot \left[-\frac{\partial x}{\partial \eta}\frac{\partial \hat{\psi}_j}{\partial \xi} + \frac{\partial x}{\partial \xi}\frac{\partial \hat{\psi}_j}{\partial \eta} \right] \right] \cdot \frac{1}{J} d\Omega_R \end{aligned} \tag{17}$$

where $N_{elements}$ corresponds to the number of elements on the mesh and, in each integral in the sum, the metric and Jacobian terms are those obtained using the mapping for that specific triangle. It is important to note that the basis functions ψ_i and ψ_j in the integrand contain a hat as, when mapping the integral, these transform into the nodal basis functions defined on the reference domain. Similarly, the linear form may also be mapped to the reference domain and is given as

$$b(\psi_j) = \sum_{k=1}^{N_{elements}} \int_{\Omega_R} -\hat{\psi}_j \cdot f(x(\xi,\eta), y(\xi,\eta)) \cdot J \; d\Omega_R. \tag{18}$$

where $x(\xi,\eta)$ and $y(\xi,\eta)$ are from the transformation given in equation (13).

The final topic that needs to be addressed concerns now the numerical evaluation of the given integrals. On triangular elements, accurate quadrature rules may be derived. For an exact evaluation of integrals of functions of degree 1, a quadrature rule of order 1 may be used and is given as

$$\int_{\Omega_k} f(x,y)\; dx\; dy = A_k \cdot f(\hat{x}, \hat{y}) \tag{19}$$

where Ω_k is the domain of the triangle, A_k is its area, and $(\hat{x}, \hat{y})$ correspond to the centroid of the given triangle. Moreover, a quadrature rule of order 3, which is exact for integrating polynomials of order 2, is given by

$$\int_{\Omega_k} f(x,y)\; dx\; dy = \frac{1}{3}\; A_k \; \sum_{l=1}^{3} f(x_l, y_l) \tag{20}$$

where the quadrature nodes (x_l, y_l) are the midpoints of the sides of each triangle. For the evaluation

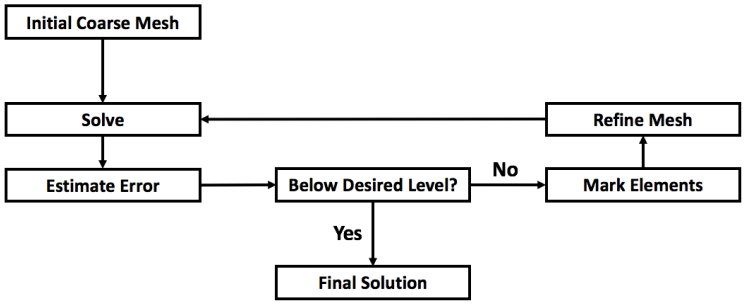

Figure 3: The general mesh adaptation algorithm used.

of integrals, the quadrature rule given by equation (20) was used. For more details regarding integration over triangles, we refer the reader to [2].

3 The Adaptation Algorithm

In this section, we outline the automatic mesh adaptation algorithm. The general procedure to the adaptation process is shown in Figure (3). Here, it can be seen that we first begin with an initial coarse mesh. Once the solution is computed on this mesh, the difficult task of estimating the error local to each element is completed. Following this, if the error is low enough, the adaptation procedure is discontinued. Otherwise, the elements with the highest error are marked and refined and the process is repeated.

3.1 Error Estimates

The estimation of the error is a crucial step in the adaptation algorithm. From implicit and explicit a-posterioi estimators, to adjoint based approaches, numerous techniques exist for this task [3].

In this work, we utilize the method proposed by Babuska and Rheinboldt in [4] to estimate the error of a given numerical solution. This approach entails solving a localized auxiliary problem on subdomains in the mesh. To illustrate this, we consider the problem of estimating the error when solving the Poisson equation as given in equation (1). Consider that the numerically computed solution on the domain is $\hat{u}_0$. Moreover, if the space of nodal basis functions to resolve the Galerkin approximation is given as $\{\psi_i\}$, then the support of a given node, denoted as $supp(\psi_i)$, is the subdomain of all nodes on which the nodal basis function is non-zero. An example of the support of a nodal basis function that is equal to 1 at $(x, y) = (0.5, 0.5)$ is shown in Figure (4) for a triangular mesh. With these defined, the auxiliary problem is given as

$$\begin{aligned} \nabla^2 w &= f \quad on\ \Omega_i = supp(\psi_i) \\ w &= \hat{u}_0 \quad on\ \partial\Omega_i. \end{aligned} \tag{21}$$

In practice, this auxiliary problem cannot be solved exactly and a finite element discretization is once more used. However, in order to obtain a more accurate solution than $\hat{u}_0$, high order basis

functions or a refinement of the domain Ω_i is done. An example of this process is shown in Figure (5) when computing the auxiliary problem over $supp(\psi_i)$ where ψ_i corresponds to the node at $(x, y) = (0.5, 0.5)$ (the same basis function shown in Figure (4)). In this case, a more accurate auxiliary solution is computed by refining the subdomain into more elements as seen in the Figure. With the auxiliary problem solved for w, the error indicator on the subdomain Ω_i is then given as

$$\eta_i^2 = a(\hat{u}_0 - w, \hat{u}_0 - w) = \int_{\Omega_i} \nabla(\hat{u}_0 - w) \cdot \nabla(\hat{u}_0 - w) d\Omega. \tag{22}$$

The error indicator can be seen to use the energy norm to estimate the error on each subdomain Ω_i. Moreover, it is important to note that in general multiple nodal basis functions will be non-zero in each element resulting in multiple values of η_i in that element (due to the fact that $supp(\psi_i)$ may overlap between some basis functions). In these situations, we define the error indicator on a given element as the maximum of all these computed values.

The error indicator on each element finally gives a measure of how much each element contributes to the global error. It therefore makes sense to mark the elements to refine as those with the higher values of η_i in order to reduce the solution error. This is what was done for all numerical cases presented, as all elements with

$$\eta \geq \lambda \cdot \max_{1 \leq j \leq N_e} \eta_j, \quad \lambda \in [0, 1].$$

where N_e refers to the number of elements on the mesh, were marked and refined in the adaptation process. Moreover, values of λ between 0.4 and 0.5 were used for the cases investigated.

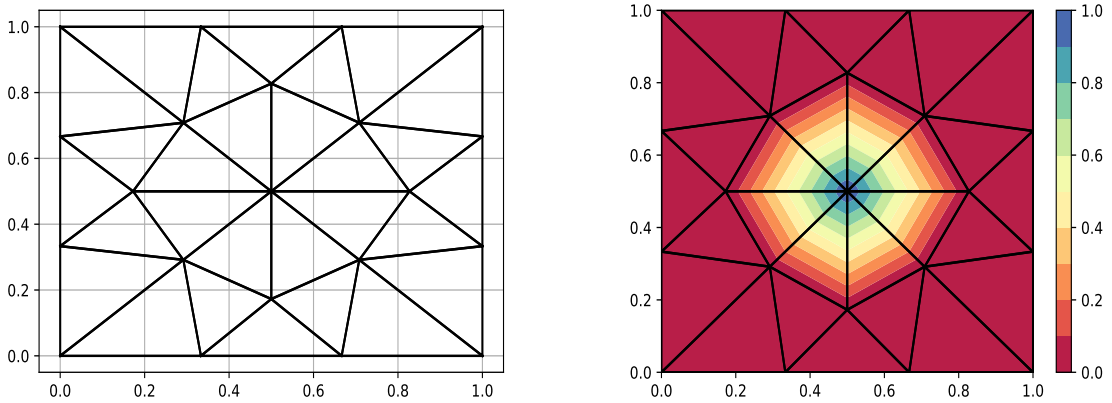

Figure 4: A sample of a triangular mesh and support of the nodal basis function that is non-zero at $(x, y) = (0.5, 0.5)$.

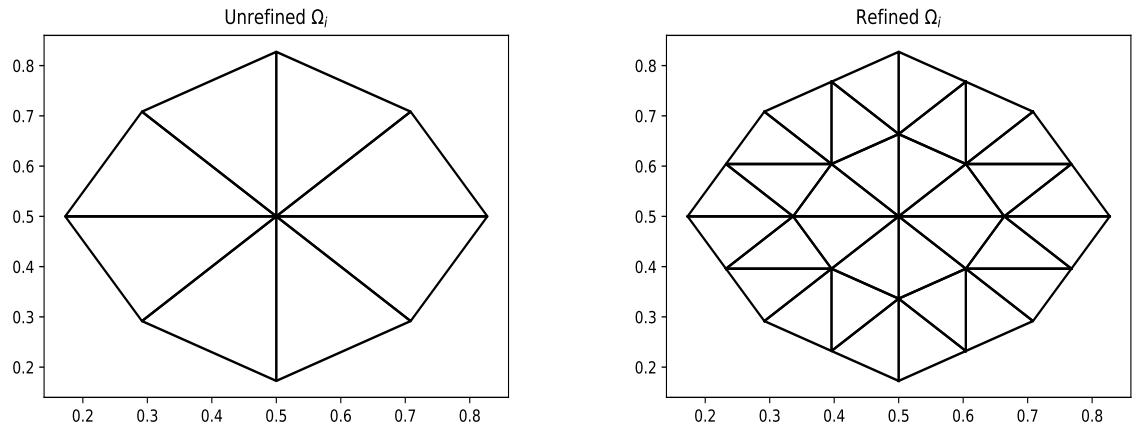

Figure 5: A schematic depicting the unrefined and refined subdomain Ω_i used to numerically solve the auxiliary problem.

3.2 Mesh Refinement

With the details covered on how the error may be estimated, we finally turn to describing how to split elements in order to generate finer meshes.

Central to the refinement process is that the angles of each triangle remain bounded away from 0 and π, for it has been numerically shown that as the maximum angle in a given mesh approaches π the interpolation error grows [5]. As a result, any refinement process must ensure that this is taken into account when determining how to split the elements. In this work, the refinement algorithm proposed by Bank (sometimes referred to as the Red-Green algorithm) has been used [6] [7]. In this approach, elements are either divided into 4 (by joining the midpoints of the edges) or bisected, as seen in Figure (6). Moreover, hanging nodes are removed by refining surrounding triangles in a systematic manner. For details of this algorithm, we refer readers to [5], [6] and [7].

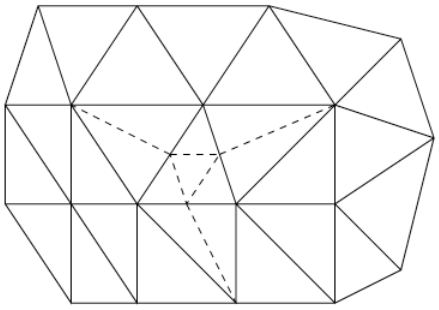

Figure 6: A schematic depicting the refinement of a given element using the Red-Green algorithm.

4 Numerical Results

To test the efficacy of the adaptation algorithm, a method of manufactured solutions was used. An analytical solution $u_{analytical}$ was first defined on a given domain. Then, this analytical solution was used to obtain the exact expression for $f(x, y)$ (the right hand side for the Poisson equation) and for the Dirichlet boundary condition g_D on the boundary of the domain. Moreover, for all cases, an initial coarse unstructured mesh, generated using GMSH, has been used.

4.1 Case 1

As a first case, we consider solving the following Poisson equation with the specified analytical solution:

$$\begin{aligned} \nabla^2 u &= f(x,y) \quad \text{on } \Omega = [0,1] \text{ x } [0,1] \\ u &= g_D \quad \text{on } \partial\Omega \\ u_{analytical} &= e^{5\,(x+y)}\ sin(\pi x)\ sin(\pi y). \end{aligned} \tag{23}$$

The sequence of refined meshes generated during the adaptation algorithm are shown in Figure (7). It can be seen that, as expected, the mesh refines itself at the top corner where the analytical solution has the highest amount of variation. Moreover, Figure (8) depicts the numerical solution on the coarsest mesh, the numerical solution on the final adapted mesh and the analytical solution. Good agreement can be seen between the final numerical and the analytical solution. Moreover, the mesh can be seen to be finest near the boundary at the corner as the solution rapidly decreases from a value with a magnitude of about 1000 to a value of 0 (at the boundary). Figure (9) shows the distribution in the error indicator η for each mesh. Near $(x, y) = (0, 0)$, where the solution variation is small, the error indicator has a low value while at the top right corner of the domain the indicator value is large, resulting in these elements to be marked for refinement. Finally, the convergence in the L_∞ norm of the error as a function of degrees of freedom is shown in Figure (10) where both adaptive and uniform refinement (which was completed using GMSH) was compared. A significant drop in the L_∞ norm can be seen in the first few adaptation steps, and the adaptive scheme can be seen to get quite low error values with a relatively low number of elements. As the number of degrees of freedom increase, negligible difference is noted between the adaptive and uniform refinement, which may primarily be due to the moderately smooth nature of the analytical solution.

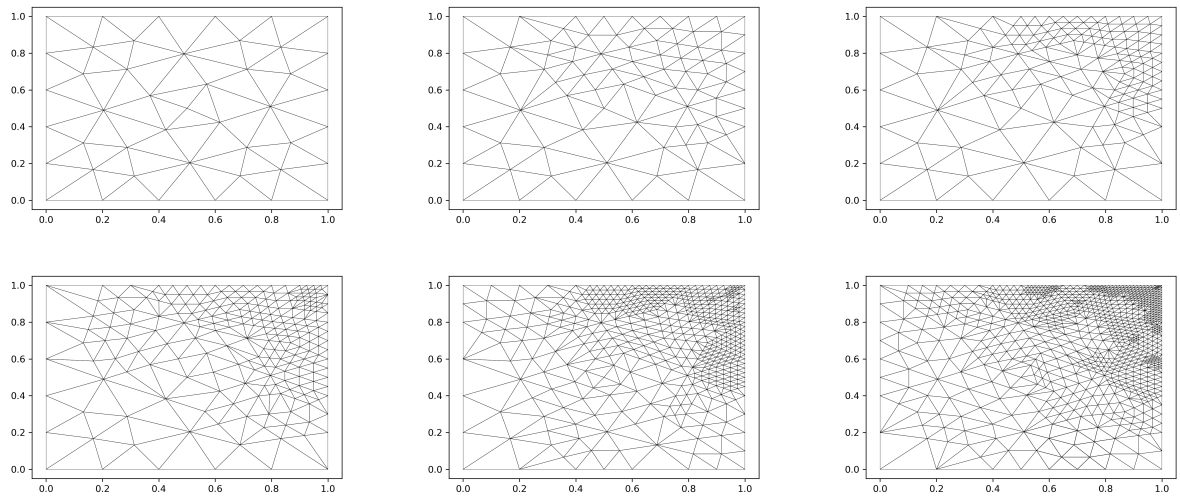

Figure 7: The sequence of refined meshes obtained in the adaptation process. The initial mesh is shown on the top left and the final adapted mesh is at the bottom right.

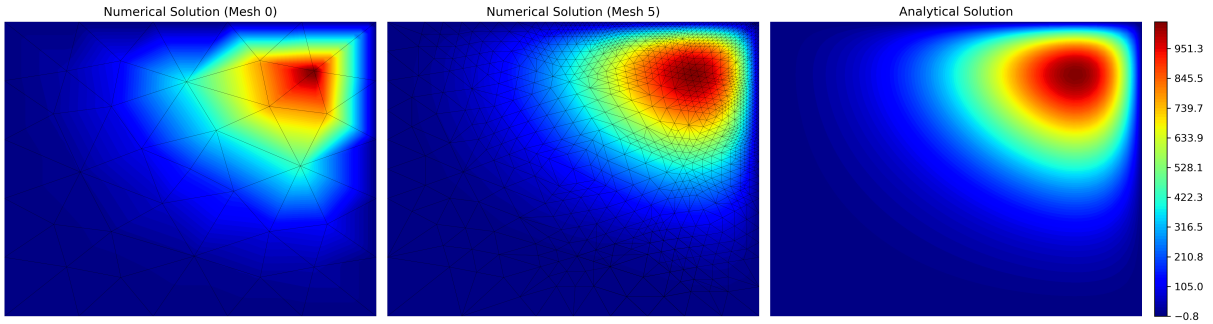

Figure 8: The numerical solution computed on the initial mesh (left), the final adapted mesh (middle) and the analytical solution (right).

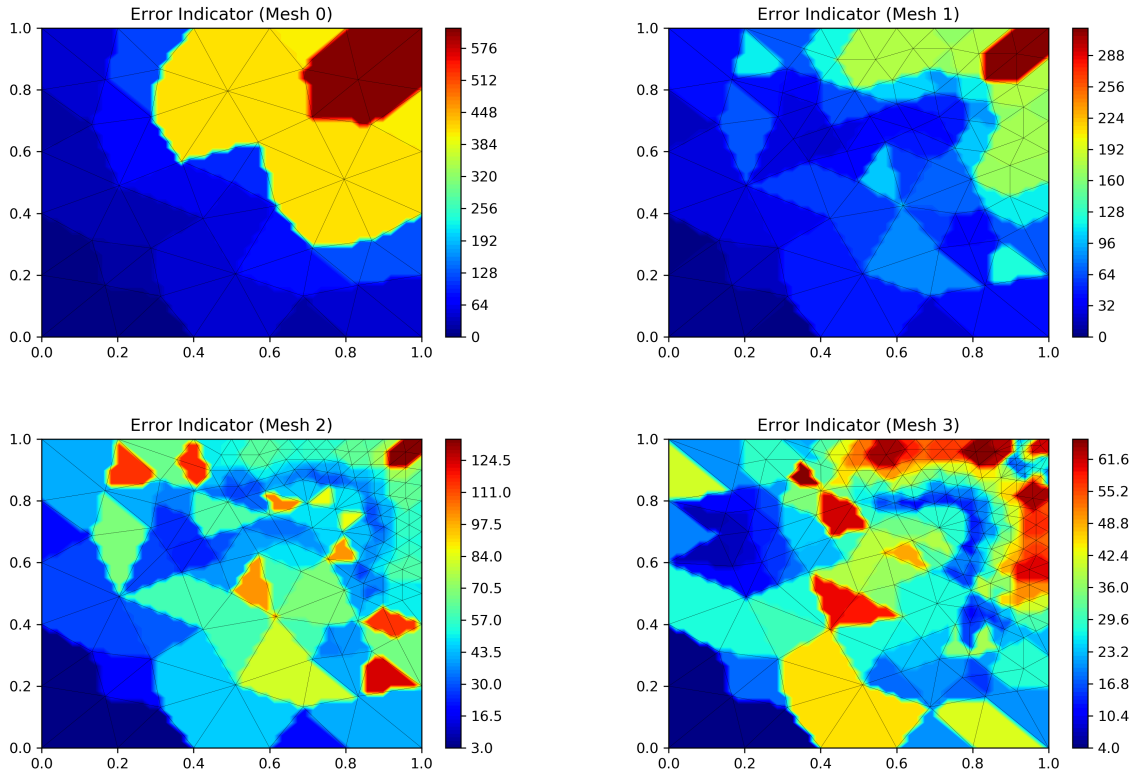

Figure 9: The error indicator (η_i) distribution for the first few meshes in the adaptation process.

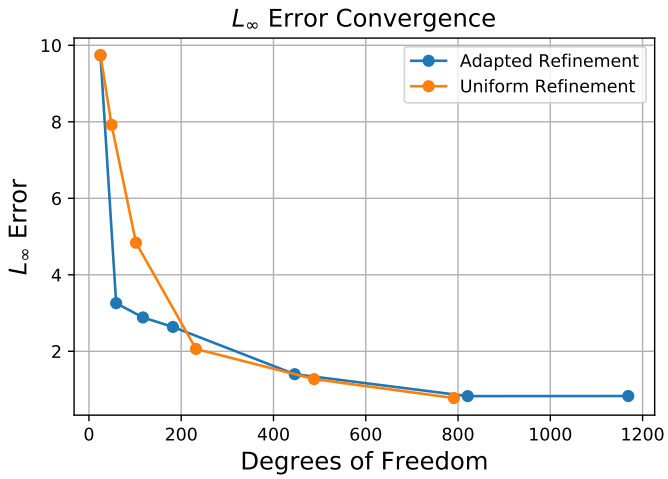

Figure 10: The convergence in the global L_∞ error as degrees of freedoms are added during the mesh adaptation process.

4.2 Case 2

We consider in this case a similar Poisson problem with an alternate analytical solution:

$$\begin{aligned} \nabla^2 u &= f(x,y) \quad && \text{on } \Omega = [0,1] \text{ x } [0,1] \\ u &= g_D && \text{on } \partial\Omega \\ u_{analytical} &= e^{-50\,((x-0.35)^2+(y-0.35)^2)} \end{aligned} \tag{24}$$

As before, the sequence of refined meshes can be seen in Figure (11). The meshes refine more and more about the peak of the exponential, while there is little to no refinement in the region of the domain where the solution is approximately constant with a value of 0. Figure (12) compares the numerical solutions on the coarsest mesh and the final adapted mesh along with the analytical solution, where it can be seen that the final numerical solution follows the analytical one quite closely. Moreover, Figure (13) shows the progression in the distribution in the error indicator η for each mesh where it can be seen that the solver consistently marks elements to refine near $(x, y) = (0.35, 0.35)$ (the error indicator is highest in this region). Lastly, the convergence in the L_∞ norm of the error versus degrees of freedom is shown in Figure (14) where both adaptive and uniform refinement has been compared. This case shows a significant advantage in using adaptation, as consistently lower error values are seen for the adaptive case. In fact, to obtain an error value near 0.001, about 800 degrees of freedom are needed for the uniform refinement case compared to only 150 degrees of freedom for the adaptive case. This non-negligible difference in degrees of freedom gives the adaptive approach much more computationally efficient for this case.

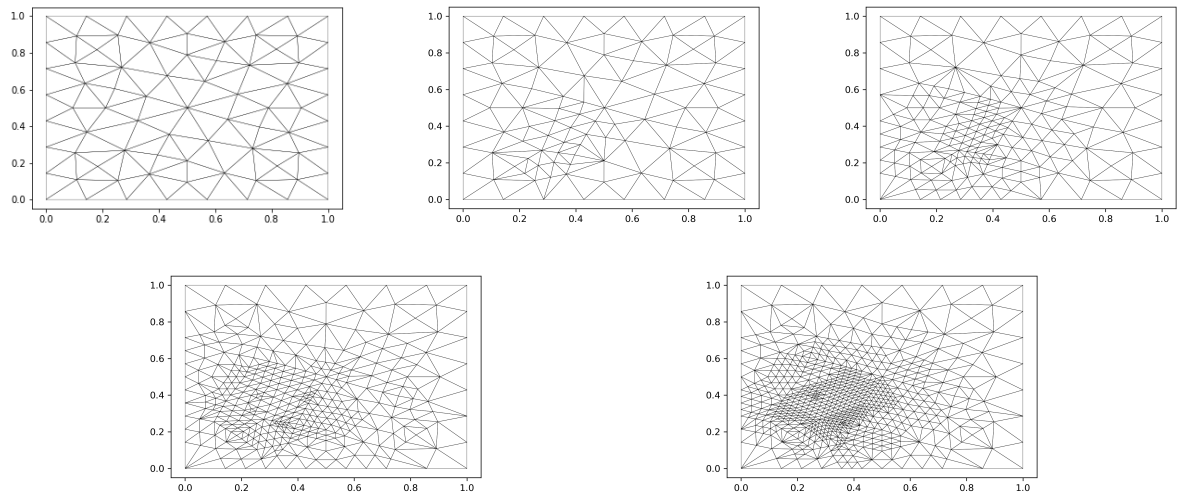

Figure 11: The sequence of refined meshes obtained in the adaptation process. The initial mesh is shown on the top left and the final adapted mesh is shown on the bottom right.

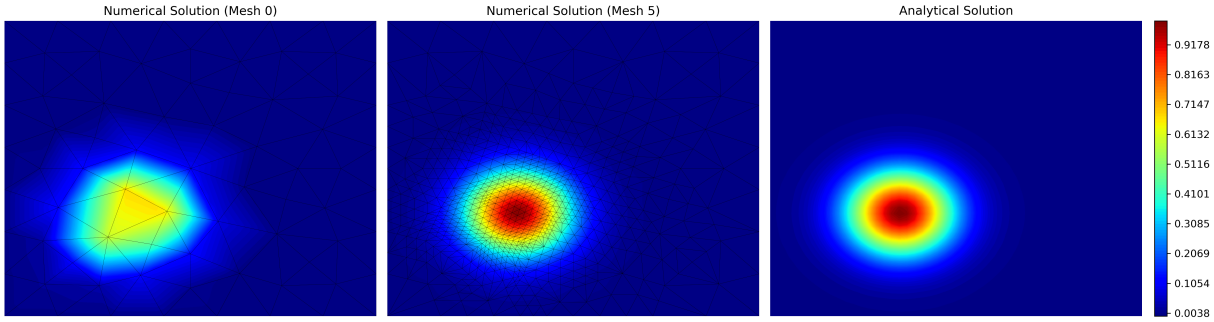

Figure 12: The numerical solution computed on the initial mesh (left), the final adapted mesh (middle) and the analytical solution (right).

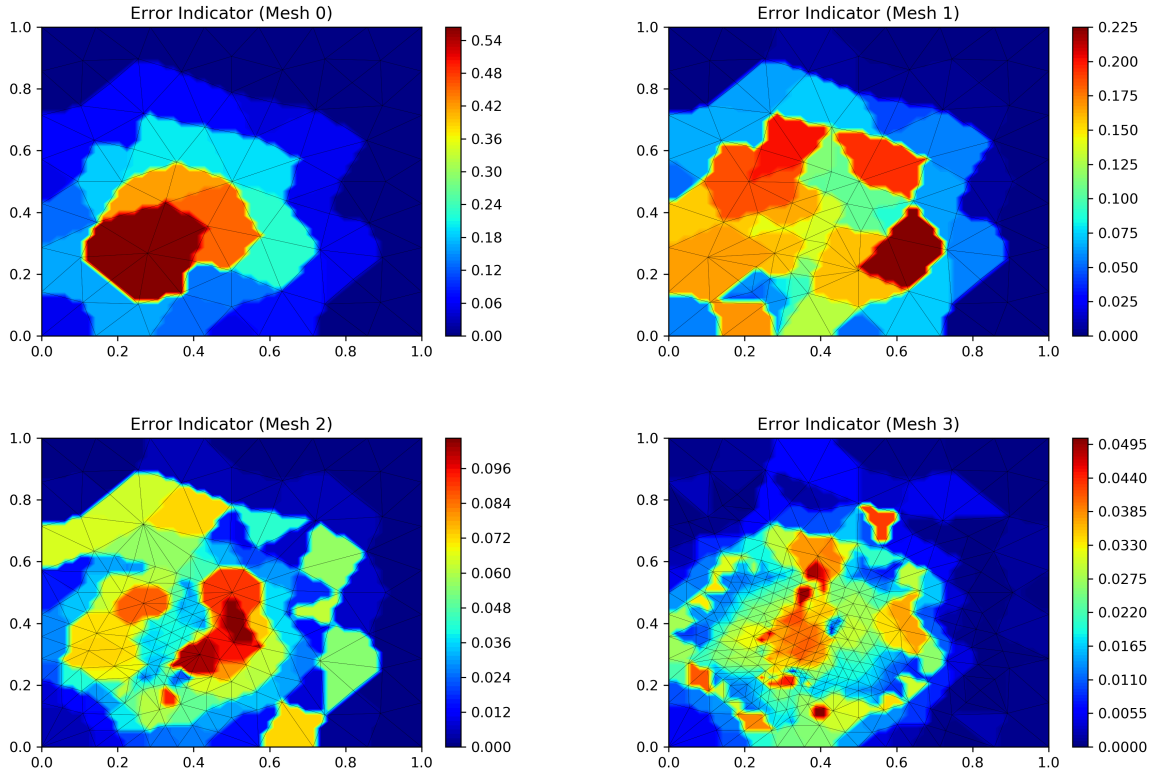

Figure 13: The error indicator (η_i) distribution for the first few meshes in the adaptation process.

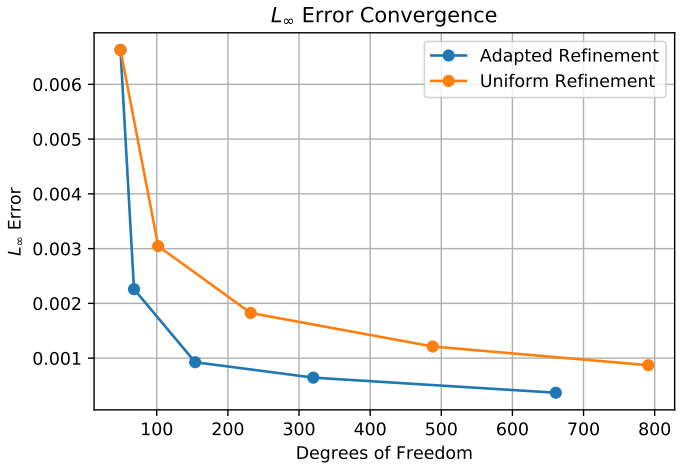

Figure 14: The convergence in the global L_∞ error as degrees of freedoms are added during the mesh adaptation process.

5 Conclusions and Future Work

In this work, the adaptive finite element method has been studied. In particular, h-adaptive methods have been investigated and compared to the case of uniform refinement. Appreciable increases in the efficiency were noted for the adaptive approach, as relatively low errors could be reached with fewer degrees of freedom.

Several additional avenues may be further investigated. One of these is the study of p-adaptivity coupled with the h-adaptive framework that has been implemented. In this way, the influence of changing the polynomial order may also be tested to see if further gains are realized. Moreover, different error estimators may be studied as well. Several approaches exist to quantify the error to know which elements to refine, and it would be quite relevant to investigate the efficacy of each of these different approaches. Finally, it would be interesting to study adaptivity to different PDEs. We have focused in this work exclusively on the Poisson equation (an Elliptic PDE), but more work may be done to investigate how the effects of adaptivity vary for Hyperbolic or Parabolic equations.

References

[1] I. Babuška and B. Guo, "The h, p and hp version of the finite element method; basis theory and applications," *Advances in Engineering Software*, vol. 15, no. 3-4, pp. 159–174, 1992.

[2] B. D. Reddy, "Introductory functional analysis with applications to boudary value problems and finite elements," 1998.

[3] M. Ainsworth and J. T. Oden, *A posteriori error estimation in finite element analysis*, vol. 37. John Wiley & Sons, 2011.

[4] I. Babuvška and W. C. Rheinboldt, "Error estimates for adaptive finite element computations," *SIAM Journal on Numerical Analysis*, vol. 15, no. 4, pp. 736–754, 1978.

[5] W. F. Mitchell, "A comparison of adaptive refinement techniques for elliptic problems," *ACM Transactions on Mathematical Software (TOMS)*, vol. 15, no. 4, pp. 326–347, 1989.

[6] R. E. Bank and A. H. Sherman, "An adaptive, multi-level method for elliptic boundary value problems," *Computing*, vol. 26, no. 2, pp. 91–105, 1981.

[7] R. E. Bank, A. H. Sherman, and A. Weiser, "Some refinement algorithms and data structures for regular local mesh refinement," *Scientific Computing, Applications of Mathematics and Computing to the Physical Sciences*, vol. 1, pp. 3–17, 1983.