Multigrid for Fourier Finite Element Methods on Axisymmetric Domains

Trevor Crupi[1], Yonah Moise[2], and Hannah Odom[3]

[1]Purdue University, Department of Mathematics
[2]Yeshiva University, Department of Mathematics
[3]University of Texas at Austin, Department of Mathematics

August 2020

Summer @ ICERM 2020

Primary Advisor:	Secondar Advisor:	Teaching Assistants:
Dr. Minah Oh	Dr. Yanlai Chen	Justin Dong
		Alex Mihai

Contents

Abstract

We show numerically that in boundary value problems posed in these weighted $H(curl)$ spaces over axisymmetric domains, the multigrid V-cycle with modern smoothers converge uniformly with respect to meshsizes. Numerical results also indicate that for larger Fourier modes, the convergent rate decreases slightly compared to lower integer values for Fourier modes.

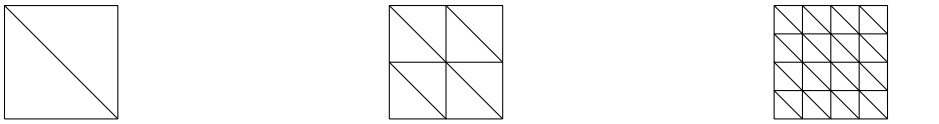
1 Introduction

FEMs are some of the most widely used techniques for approximating solutions to both ordinary and partial differential equations. FEMs have a wide range of applications in physics and engineering ranging from thermodynamics, structural analysis, and fluid dynamics. In order to obtain close approximations, FEMs must solve very large matrix systems in which classical methods can be very inefficient. Thus, iterative methods are often used to solve these systems more efficiently. In this report, we study how the multigrid V-cycle algorithm can be utilized to efficiently solve Fourier Finite Element Methods (Fourier-FEMs) posed in weighted $H(curl)$ spaces on axisymmetric domains, and how this method compares to other iterative methods. Fourier-FEMs over axisymmetric domains have been studied in [4], [3], however it has yet to be shown mathematically or numerically that multigrid V-cycle is uniformly convergent with respect to meshsize for problems posed in weighted Sobolev spaces that use the curl operator arising from a Fourier series decomposition on axisymmetric domains.

2 Background

2.1 Mesh Geometry

FEMs discretize their n-dimensional domains by subdividing the domain into n-dimensional polygons. There are many different options for subdivision by polygons, but for our purposes we will be focusing on domains that are two-dimensional and subdivided by triangles.



(a) Mesh Level 1 (b) Mesh Level 2 (c) Mesh Level 3

Figure 2.1

Definition 2.1 *A mesh τ over a domain $\Omega \subset \mathbb{R}^2$ is a set consisting of all the triangles which subdivide Ω.*

For this report, our mesh will be built by initially placing large triangles across the whole domain and adding more triangles by bisecting the edges of each triangle at the midpoints and drawing diagonal lines. The act of bisecting the domain at each triangle is a *refinement*, with a less-refined grid being referred to as *coarser* and a more well-refined grid being *finer*. Figure 2.1 shows an example of refining a mesh over the unit square 3 times. By a *mesh level* we mean the number of refinements we have made to our mesh, and we use a subscript τ_h to denote the mesh level. Note that the way we have defined our mesh levels indicates the following chain of nested meshes:

$$\tau_1 \subset \tau_2 \subset \cdots \subset \tau_N. \tag{1}$$

This chain has important consequences for the finite-dimensional spaces we will construct. As $N \to \infty$, the mesh becomes a better approximation of the set of points in Ω. This sequence of nest meshes will be fully utilized when using multigrid as well.

2.2 Function Spaces and Weak Forms

We now define the main function spaces of interest: $L^2(\Omega)$, the space of square-integrable functions, and the Sobolev space $H^1(\Omega)$, the space of square-integrable functions whose first derivatives are also square-integrable. We define these more precisely. Let $\Omega \subset \mathbb{R}^n$, then:

$$L^2(\Omega) = \left\{ f : \Omega \to \mathbb{R} : \int_\Omega |f|^2 dV < \infty \right\}$$
$$H^1(\Omega) = \left\{ f \in L^2(\Omega) : \frac{\partial f}{\partial x_i} \in L^2(\Omega), 1 \le i \le n \right\}.$$

We define the partial derivatives $\frac{\partial f}{\partial x_i}$ to be the *weak derivative*. We state this as a formal definition [2].

Definition 2.2 *Let $f, g \in L^2(\Omega)$, and $\psi \in C_c^\infty(\mathbb{R})$, where $C_c^\infty(\mathbb{R})$ is the space of smooth functions with compact support. We say that g is the weak derivative of f if*

$$\int_\Omega f \frac{\partial \psi}{\partial x_i} dx = -\int_\Omega g\psi dx. \tag{2}$$

Given this new definition, we will refer to the popular definition of the derivative given in calculus as the *strong derivative*. Note that in the case of a function with a well-defined strong derivative the weak derivative is equivalent to the strong derivative. We

can speak of the associated *weak form* of a PDE, which replaces the strong derivatives in the differential form with weak derivatives in an integral form. As an example, we will consider the *reaction-diffusion equation*. The well-known differential form is:

$$\begin{cases} -\Delta u + u = f & \text{on } \Omega \\ \nabla u \cdot n = g & \text{on } \partial\Omega \end{cases}. \tag{3}$$

Using definition 2.2, we obtain the weak formulation of the reaction-diffusion equation by multiplying by a test function $v \in H^1(\Omega)$ and integrating by parts:

$$\int_\Omega \nabla u \cdot \nabla v + uv = \int_\Omega fv \quad \forall v \in H^1(\Omega). \tag{4}$$

Because of the dot product of the gradients that shows up on the left-hand side, the possible solutions u to this equation are only valid if the derivatives of u and v have well-defined integrals, i.e. $u, v \in H^1(\Omega)$. Our goal is to construct a finite-dimensional subspace of $H^1(\Omega)$ that can be used to approximate solutions to the weak form of (4).

2.3 Finite Element Spaces

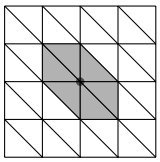
We define the polynomial space V_h to be:

$$V_h = \left\{ v_h : v_h\big|_T = ax + by + c, \forall T \in \tau_h, \text{ and } v_h \text{ is globally continuous } \right\}. \tag{5}$$

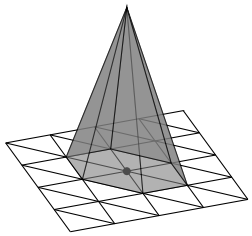
V_h is the continuous piecewise linear space, i.e., the set of all functions that, when restricted to an arbitrary triangle in our mesh, can be written as the polynomial $v_h = ax + by + c$. We may also write v_h in vector form as $\left[a, b, c\right]^T$. Since any $v_h \in V_h$ is in $L^2(\Omega)$ and has well-defined first partial derivatives over any arbitrary triangle T in our mesh, it follows that $V_h \subset H^1(\Omega)$. The basis of V_h is the set of functions $\{\phi_i\}_{i=1}^N$, where N is the number of nodes, or triangle vertices, in our mesh. These basis functions have the following constraint:

$$\phi_i(v_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{6}$$

where v_j is the j^{th} vertex of our mesh. Geometrically, the basis (6) can be thought of as a "tent" function, which connects piecewise planes over a vertex v_j to a height of 1 (see Figure).



(a)



(b)

Figure 2.2

Next, we define W_h, or the *lowest order Nedelec space*, to be:

$$W_h = \left\{ w_h : w_h\big|_T = \begin{bmatrix} B - Ay \\ C + Ax \end{bmatrix} \;\; A, B, C \in \mathbb{R},\;\; A_T \in \tau_h, \right.$$
$$\left. \int_e w_h \cdot t \, ds \text{ is continuous for each edge } e \text{ in } \tau \right\} \quad (7)$$

where t is a unit tangent vector. Note that V_h has a basis corresponding to each node, while W_h has a basis corresponding to each edge. Furthermore, W_h differs from V_h in that the functions in W_h are vector-valued, and that we require the line integral over an arbitrary edge of a triangle to be continuous. We define our basis functions to be the set $\{\psi_i\}_{i=1}^n$, where n is the number of triangle edges. We add the following constraint to the basis functions:

$$\int_{e_j} \psi_i \cdot t \, ds = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}. \quad (8)$$

This constraint means that the line integral must be one over the corresponding edge in the direction of the unit tangent vector. We note that V_h has a basis function corresponding to each *node*, whereas W_h has a basis function corresponding to each *edge*. W_h is widely used to approximate the solution to problems involving the curl operator such as the Maxwell equations.

2.4 Finite Element Methods

In section 2.3, we introduced finite-dimensional spaces that act as good approximations to functions that live in $H^1(\Omega)$. FEMs utilizes these spaces to compute approximations to the weak form of partial differential equations. In this section, we apply FEMs to the *Reaction Diffusion equation* to demonstrate how to obtain a finite element approximation of the exact solution. The finite element space that we use for this problem is the continuous piecewise linear space V_h.

Given a function $f \in L^2(\Omega)$, we wish to solve:

$$-\Delta u + u = f \quad (9)$$

with Neumann boundary conditions:

$$\nabla u \cdot \vec{n} = 0 \text{ on } \partial\Omega. \tag{10}$$

Using Definition 2.2, we rewrite equation (9) in its weak form:

$$\int_\Omega \nabla u \cdot \nabla v + uv = \int_\Omega fv \quad \forall v \in H^1(\Omega). \tag{11}$$

Next, we approximate the weak form by taking $u_h, v_h \in V_h$. We then rewrite equation (11) in terms of our approximated functions u_h and v_h to obtain:

$$\int_\Omega \nabla u_h \cdot \nabla v_h + u_h v_h = \int_\Omega f v_h \quad \forall v_h \in V_h. \tag{12}$$

Since $u_h \in V_h$, we may rewrite u_h as a sum of basis functions:

$$u_h = \sum_{i=1}^{N} c_i \phi_i. \tag{13}$$

Since v_h can be *any* element in V_h, we choose $v_h = \phi_j$, an arbitrary basis function. We can now substitute these new expressions for u_h and v_h:

$$\int_\Omega (\sum_{j=1}^{n} c_j \cdot \nabla\phi_j \cdot \nabla\phi_i + \sum_{j=1}^{n} c_j \cdot \phi_j \cdot \phi_i) = \int_\Omega f \cdot \phi_i \tag{14}$$

for all $i = 1, ..., N$.

Since the basis functions are known explicitly, the only unknown in our equation becomes the constants c_j which completely define our approximate solution u_h. Thus, we may solve this approximation by transforming the equation into a matrix system $\mathbf{A}\vec{c} = \vec{b}$, which we obtain by setting $A_{i,j}$ and b_i as:

$$A_{i,j} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j + \phi_i\phi_j \tag{15}$$

$$b_i = \int_\Omega f \cdot \phi_i. \tag{16}$$

In matrix form:

$$\begin{bmatrix} \int_\Omega(\nabla\phi_1 \cdot \nabla\phi_1 + \phi_1 \cdot \phi_1) & ... & \int_\Omega(\nabla\phi_N \cdot \nabla\phi_1 + \phi_N \cdot \phi_1) \\ \int_\Omega(\nabla\phi_1 \cdot \nabla\phi_2 + \phi_1 \cdot \phi_2) & ... & \int_\Omega(\nabla\phi_N \cdot \nabla\phi_2 + \phi_N \cdot \phi_2) \\ . & ... & . \\ . & ... & . \\ . & ... & . \\ \int_\Omega(\nabla\phi_1 \cdot \nabla\phi_N + \phi_1 \cdot \phi_N) & ... & \int_\Omega(\nabla\phi_N \cdot \nabla\phi_N + \phi_N \cdot \phi_N) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ . \\ . \\ . \\ c_N \end{bmatrix} = \begin{bmatrix} \int_\Omega f \cdot \phi_1 \\ \int_\Omega f \cdot \phi_2 \\ . \\ . \\ . \\ \int_\Omega f \cdot \phi_N \end{bmatrix} \tag{17}$$

Now, once we fill in the A-matrix and the b-vector, it is easy to solve for the c-vector filled with constants. The solution to this matrix system gives us the constants $c_1, \dots c_N$

which completely define our solution u_h. By taking finer mesh sizes, we are able to compute a closer approximation to u.

FEMs allows for very close approximations to solutions of PDEs. However, for large mesh levels the resulting matrix system may become large. In section 2.8, we will explore iterative methods that allow us to more efficiently compute the solution to these large matrix systems.

2.5 Axisymmetric Domains

An axisymmetric domain is any domain $\breve{\Omega} \subset \mathbb{R}^3$ such that $\breve{\Omega}$ is symmetric about an axis. When dealing with axisymmetric domains, it is often more convenient to work in cylindrical coordinates (r, θ, z) as opposed to rectangular coordinates (x, y, z). In cylindrical coordinates, if $\breve{\Omega}$ is axisymmetric, then θ is fixed over $\breve{\Omega}$. Thus, we may perform a dimension reduction by looking at $\breve{\Omega}$ restricted to the (r, z) plane. Given an arbitrary vector-valued function u defined on an axisymmetric domain $\breve{\Omega}$, we will be looking at the Fourier decomposition $u = u^s + u^a$ given by:

$$\begin{aligned} u^s &= \begin{bmatrix} u_r^0 \\ 0 \\ u_z^0 \end{bmatrix} + \sum_{k=1}^{\infty} \begin{bmatrix} u_r^k \cos(k\theta) \\ u_\theta^k \sin(k\theta) \\ u_z^k \cos(k\theta) \end{bmatrix} \\ u^a &= \begin{bmatrix} 0 \\ u_\theta^0 \\ 0 \end{bmatrix} + \sum_{k=1}^{\infty} \begin{bmatrix} u_r^{-k} \sin(k\theta) \\ u_\theta^{-k} \cos(k\theta) \\ u_z^{-k} \sin(k\theta) \end{bmatrix}. \end{aligned} \tag{18}$$

A specific value of $k \in \mathbb{Z}$ is referred to as a k^{th} *order Fourier mode*. In cylindrical coordinates, the curl operator is given by:

$$curl\ \vec{u} = \begin{bmatrix} \frac{1}{r}\frac{\partial u_z}{\partial \theta} - \frac{\partial u_\theta}{\partial z} \\ \frac{\partial u_r}{\partial z} - \frac{\partial u_z}{\partial r} \\ \frac{1}{r}(\frac{\partial u_\theta}{\partial r} - \frac{\partial u_r}{\partial \theta}) \end{bmatrix}. \tag{19}$$

When we apply (19) to the Fourier decomposition in (18), and due to the $L^2(-\pi, \pi)$ orthogonality of $\cos k\theta, \sin k\theta, 1$, each Fourier mode decouples in a weak formulation, and the following curl operator is obtained for the k^{th} Fourier mode.

$$curl_{rz}^k \vec{u} = \begin{bmatrix} -\frac{k}{r}u_z - \partial_z u_\theta \\ \partial_z u_r - \partial_r u_z \\ \partial_r u_\theta + \frac{1}{r}u_\theta + \frac{k}{r}u_r \end{bmatrix}. \tag{20}$$

Where the superscript k indicates that this operator depends on a fixed Fourier mode.

2.6 Weighted Function Spaces

When performing a dimension reduction using cylindrical coordinates, due to the Jacobian arising from change of variables, the problem is posed in weighted function spaces once the dimension reduction is done. Recall the definition of the function space $L^2(\Omega)$ from section 2.2. When we work in cylindrical coordinates, the Jacobian of the coordinate transformation from (x, y, z) to (r, θ, z) gives an extra r in the integral, called the *weight*. We define the weighted space $L^2_r(\Omega)$ to be:

$$L^2_r(\Omega) = \big\{u : \int_\Omega u^2 r dr dz < \infty\big\}. \tag{21}$$

Here, we drop the θ since the definition of the Fourier coefficients do not depend on the θ variable, meaning θ has no effect on the outcome of the integral [5]. Using the $curl^k_{rz}$ operator from (20), we can define a new weighted Sobolev space for our axisymmetric problem:

$$H_r(curl^k_{rz}; \Omega) = \big\{u \in L^2_r(\Omega)^3 : curl^k_{rz} u \in L^2_r(\Omega)^3\big\}. \tag{22}$$

This extra weight has an effect on the size of our spaces. For instance, take $r = 0$ to be the axis of rotation and let this axis be in Ω. Then $\frac{1}{\sqrt{r}} \in L^2_r(\Omega)$, but $\frac{1}{\sqrt{r}} \notin L^2(\Omega)$. Thus, we need to be careful in our analysis on axisymmetric domains as the weighted spaces alter the problem significantly.

2.7 Fourier Finite Element Methods

In [4], [3], the following finite element space $Z_{h,k}$ was introduced in order to utilize Fourier-FEMs on axisymmetric domains. These papers proved the convergence of the Fourier-FEMs in $Z_{h,k}$. $Z_{h,k}$ is defined in the following way:

$$Z_{h,k} = \big\{Z_h : Z_h\big|_T = \begin{bmatrix} -\frac{1}{k}\beta_1 + \beta_4 r - \frac{1}{k}\beta_3 z - \beta_6 rz \\ \beta_1 + \beta_2 r + \beta_3 z \\ \beta_5 r + \beta_6 r^2 \end{bmatrix}, \beta_i \in \mathbb{R}, \forall\ 1 \le i \le 6, \forall\ T \in \tau_h\big\}. \tag{23}$$

The space $Z_{h,k}$ has a basis given by $\{\eta_i\}_{i=1}^{n+N}$, where N is the number of nodes and n the number of edges. η_i is a vector-valued function defined in cylindrical coordinates, i.e. $\eta_i = (\eta_i^r, \eta_i^\theta, \eta_i^z)^T$. Here, we borrow ideas from the finite element spaces V_h and W_h and we define constraints over the triangle vertices and triangle edges in our mesh:

$$\eta_i^\theta(v_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \qquad \int_{e_j} \begin{bmatrix} \frac{k\eta_i^r + \eta_i^\theta}{x} \\ \frac{k\eta_i^z}{x} \end{bmatrix} \cdot \vec{t} ds = \begin{cases} 1 & \text{if } i = n + j \\ 0 & \text{if } i \neq n + j \end{cases}.$$

The exact expression for the basis function is given by the vector:

$$\eta = \begin{bmatrix} -\frac{1}{k}\beta_1 + \beta_4 r - \frac{1}{k}\beta_3 z - \beta_6 rz \\ \beta_1 + \beta_2 r + \beta_3 z \\ \beta_5 r + \beta_6 r^2 \end{bmatrix}.$$

Using the constraints we have set over the edges and vertices, we can solve for $\beta_4, \beta_5, \beta_6$.

We let β_1, β_2, and $\beta_3 = 0$. By the way we have defined our constraints, we may rewrite our basis function η as a combination of basis functions from V_h and W_h by rewriting our β constants in terms of a, b, c and A, B, C. This gives:

$$\phi_i = \begin{bmatrix} -\frac{c_i}{k} - \frac{a_i}{k}r - \frac{b_i}{k}z \\ a_i r + b_i z + c_i \\ 0 \end{bmatrix} \qquad \psi_i = \begin{bmatrix} \frac{B_i}{k}r - \frac{A_i}{k}rz \\ 0 \\ \frac{C_i}{k}r + \frac{A_i}{k}r^2 \end{bmatrix}.$$

Splitting up η into the corresponding ϕ and ψ basis functions is important for computation, the computation of these basis functions are further discussed in the Appendix. We use this space $Z_{h,k}$ to approximate solutions to PDEs involving curl operators on axisymmetric domains. For instance, take the following PDE given in differential form over a dimension reduced axisymmetric domain Ω and $F \in L^2_r(\Omega)$:

$$curl^k_{rz} curl^k_{rz} u + u = F. \tag{24}$$

We wish to find a $u \in H_r(curl^k_{rz};\ \Omega)$ satisfying the weak formulation of (24):

$$\int_\Omega (curl^k_{rz} u \cdot curl^k_{rz} v)_r + (u \cdot v)_r = \int_\Omega (F \cdot v)_r \quad \forall v \in H_r(curl^k, \Omega). \tag{25}$$

We use the following boundary conditions:

$$\begin{cases} (curl^k_{rz} \vec{u})_{rz} \cdot \vec{t} = 0 & \text{on } \Gamma_1 \\ \\ (curl^k_{rz} \vec{u})_\theta = 0 & \text{on } \Gamma_1 \end{cases} \tag{26}$$

where Γ_1 is the subset of the boundary of Ω that is not on the axis of rotation, i.e., the rotation of Γ_1 around the axis of rotation will return the boundary of $\breve{\Omega}$. We approximate the weak formulation by letting $u_h, v_h \in Z_{h,k}$ and setting:

$$\int_\Omega (curl^k_{rz} u_h \cdot curl^k_{rz} v_h)_r + (u_h \cdot v_h)_r = \int_\Omega (F \cdot v_h)_r \quad \forall v_h \in Z_{h,k}. \tag{27}$$

To solve this approximation, we transform the equation into matrix-vector form, where

$$A_{i,j} = \int_\Omega (curl^k_{rz} X_i \cdot curl^k_{rz} X_j + X_i \cdot X_j) r dr dz$$
$$b_i = \int_\Omega (F \cdot X_i) r dr dz.$$

Just as in section 2.4, this gives us the matrix equation $\mathbf{A}\vec{c} = \vec{b}$. For smaller mesh levels, this can be solved using Gaussian elimination; however, for very large mesh levels Gaussian elimination can become inefficient. In the next section, we will discuss some iterative methods that will allow us to solve this matrix equation more efficiently.

2.8 Iterative Methods

As our meshes get finer the dimension of our system gets larger, making it inefficient to use Gaussian elimination to solve. It is more efficient to apply iterative methods to solve our equation. Iterative methods apply an algorithm to an initial input, and subsequently cycle the output as a new input. Each cycle of the iterative method slightly refines the input with this algorithm. This continues until until some stopping criterion is fulfilled. To solve our matrix system, we looked in depth at the *Multiplicative Subspace Correction method* and the *multigrid V-cycle method*.

2.9 Multiplicative Subspace Correction Method

First we applied the Multiplicative Subspace Correction (MSC) Method. The MSC-Method relies on the subspace decomposition

$$Z_h = \sum_{l=1}^{N} Z_\ell$$

that is, our finite element mesh Z of mesh-level h, is the summation of Z_ℓ from 1 to N, where N is the number of nodes in the subspace. Each Z_ℓ is defined as the span of the basis-functions associated with a single node and its connected edges (i.e. the span of the basis-functions associated with one vertex-patch). This is similar to the subspace decomposition used in [1].

Using the basis-functions in each Z_ℓ, the MSC-Method slowly refines the initial condition into an accurate-solution. The MSC-Method utilizes our subspace decomposition by identifying the rows and columns that correspond to the basis-functions of a Z_ℓ, take their intersections, and put them into a special A_j matrix. This dimension of this matrix is smaller than the dimension of our entire subspace, and therefore we will need to use a projection-matrix Q_j to perform operations on matrices and vectors in different dimensions.

The MSC-Method slowly refines some vector U_i with each iteration $U_{(i+1)} = MSC(U_i, f)$, with f set to our right-hand side b. One iteration of this algorithm follows three steps:

1. $U_i^0 = U_i$
2. For $j = 1, 2, ..., N$, where N is the number of nodes,
$$U_i^j = U_i^{j-1} + Q_j^T A_j^{-1} Q_j (f - A U_i^{j-1})$$
3. $U_{i+1} = U_i^N$

This U_{i+1} is now plugged back into the MSC function. If we are solving Ax=0 with the MSC method, this continues until this stopping criterion is fulfilled:

$$\frac{||X_n||_{Hr(curl^k,\omega)}}{||X_0||_{Hr(curl^k,\omega)}} < TOL$$

where TOL is a sufficiently small number as usual.

2.10 Multigrid

In this section, we describe the multigrid V-cycle that will be used to solve our problem (27).

For mesh-level $s = 1$, we define multigrid V-cycle with the inputs u (the initial condition) and f (right-hand side b), as

$$MG_1(u, f) = A_1^{-1} \cdot f$$

In our first mesh we solve the system using Gaussian elimination. This is still efficient, since the dimensions in the first mesh are small. Also, any A_s in multigrid V-cycle is defined as the global-A matrix belonging to mesh-level s.
For a mesh-level $s > 1$, multigrid V-cycle consists of three steps: pre-smoothing, coarse-grid corrections, and post-smoothing. They are formally defined as follows:

1. $V^{(1)}$: One iteration of $MSC(u, f)$.
2. $V^{(2)} = V^{(1)} + Q_{s-1}^T MG_{s-1}(0, Q_{s-1}(f - A_s V^{(1)}))$.
3. $V^{(3)}$: One iteration of $MSC^T(V^{(2)}, f)$.

In steps 1 and 3, multigrid V-cycle incorporates one iteration of the MSC-method and the MSC-method-Transpose to eliminate high-frequency errors. MSC-method-Transpose differs from MSC-method by iterating through the nodes of the mesh backwards, from N to 1. Step 2, multigrid V-cycle's recursive step, projects down to the nested mesh $s-1$ and back up to mesh-level s with the prolongation matrices P_k^T and P_k, which denote Q_{s-1} and Q_{s-1}^T, respectively.

3 Numerical Results

We investigated the convergence of multigrid V-cycle on Fourier-FEM problems that are posed in the space $H_r(curl_{rz}^k; \Omega)$, where Ω is the unit square $(0, 1) \times (0, 1)$ in the rz-plane. We studied equation (24) and computed the resulting matrix system $\mathbf{A}\vec{c} = \vec{b}$ using the Fourier-FEMs outlined in section 2.7. We used the known exact solution to equation (24):

$$u(r, z) = \begin{bmatrix} z - \frac{1}{k}(\frac{1}{3}r^3 - \frac{1}{2}r^2) \\ -kz + \frac{1}{3}r^3 - \frac{1}{2}r^2 \\ r \end{bmatrix}, \; F(r, z) = \begin{bmatrix} k(r-1) + u_r(r, z) \\ -2r + 1 + u_\theta(r, z) \\ u_z(r, z) \end{bmatrix}, \tag{28}$$

to compute the approximations up to mesh level 5 using Gaussian elimination and confirmed the convergence by verifying the weighted L_r^2-norm of the solution given by $||u_h - u||_{L_r^2(\Omega)}$ converged to zero as the mesh level grew. The process we used to compute the basis functions by using local values over triangles is outlined in the Appendix. The following chart shows our results approximating u up to mesh level 5:

Mesh Level	Weighted L_r^2 Error	Convergence Rate
1	0.008	0.25
2	0.007	1.70
3	0.002	1.90
4	0.0006	1.99
5	0.0001	2.00

As the mesh level grows, the weighted L_r^2 error gets closer to zero, and the convergence rate gets closer to 2, indicating that using the Fourier-FEMs and Gaussian elimination yields an approximation of equation (24).

Once we confirmed the convergence using Gaussian elimination, we then studied how iterative methods might improve the convergence rate for higher mesh levels by comparing the error convergence rate and the number of iterations to convergence of the MSC-method (outlined in section 2.9) and the multigrid V-cycle method (outlined in section 2.10). We recorded these values up to mesh level 6 for the MSC-method and up to mesh level 8 for multigrid V-cycle. Once we verified and compared these methods, we then compared the difference in error convergence rate and number of iterations to convergence of multigrid V-cycle for different Fourier modes.

3.1 Multiplicative Subspace Correction Method

We first implemented the MSC-method using Matlab to solve the matrix system arising from the Fourier-FEMs described in section 2.7. We recorded the number of iterations to convergence and the convergence rate up to mesh level 6.

Mesh Level	Number of Iterations	Convergence Rate
2	65	0.59
3	230	0.87
4	864	0.96
5	3274	0.99
6	12466	1

We see that the convergence rate approaches 1 as the mesh level gets finer. This means that the number of iterations to convergence is increasing linearly as the mesh level grows. This could quickly become inefficient if the mesh level gets very fine.

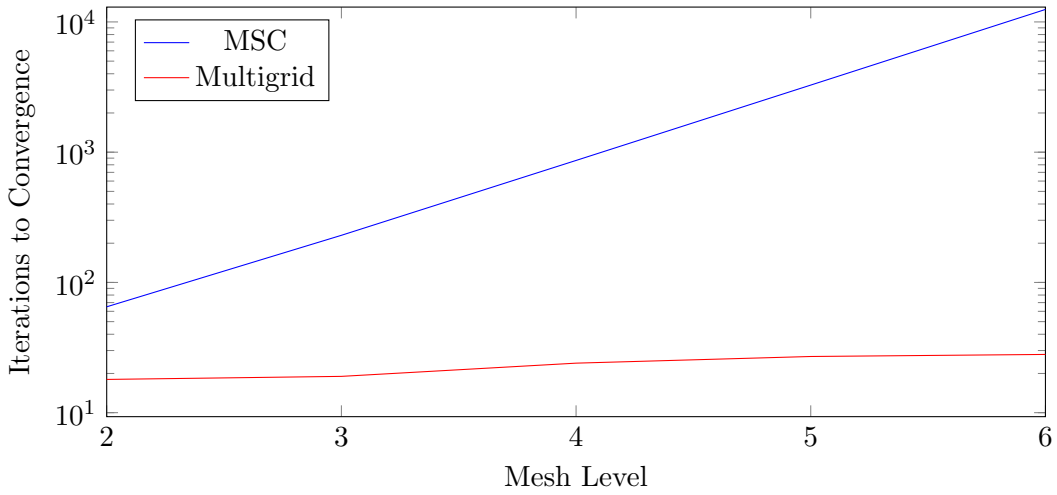
3.2 Multigrid V-Cycle

After implementing the MSC-method we applied the multigrid V-cycle method to the FEM problem for $k = 1$. We again recorded the number of iterations to convergence and the convergence rate up to mesh level 8.

Mesh Level	Number of Iterations	Convergence Rate
2	18	0.14
3	19	0.17
4	24	0.26
5	27	0.30
6	28	0.32
7	27	0.32
8	27	0.32

As the mesh level gets finer, multigrid V-cycle remains constant in both convergence rate and iterations to convergence. This is incredibly efficient even for higher mesh levels, and allows us to obtain close approximations of the solution in substantially less time than both Gaussian elimination and the MSC-method.

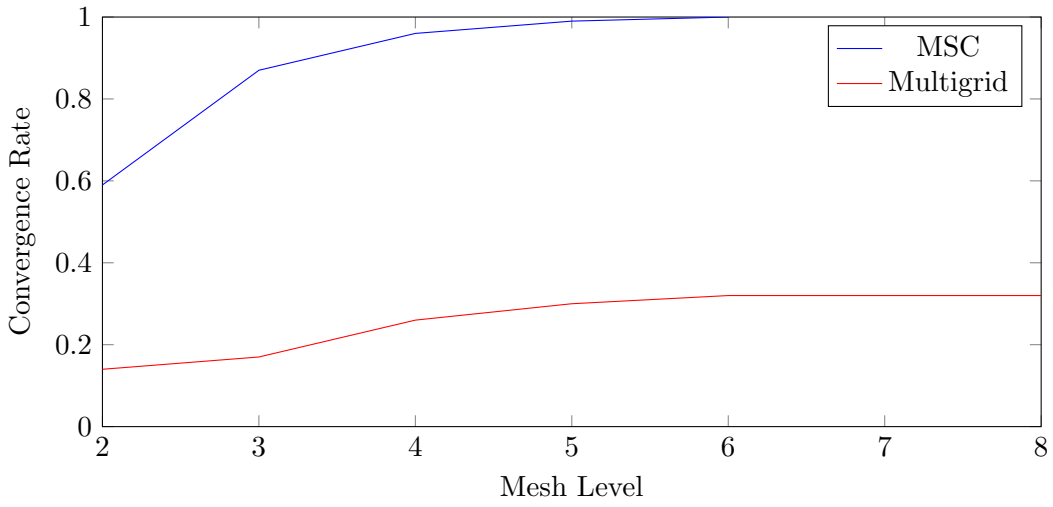
We then compared the results of MSC-method and multigrid V-cycle directly. We used a log plot to compare the number of iterations to convergence and convergence rate between the two methods.



We see again that as the mesh level gets higher, the number of iterations for Multigrid remains constant while the number of iterations for MSC grows linearly with the mesh

level.

We then compared the convergence rate for both methods up to mesh level 8.



We see that as as the mesh level grows, the convergence rate for the MSC-method gets closer to 1, implying a linear convergence rate, whereas the multigrid V-cycle method stays constant as the mesh level grows.

These results show the efficiency of the multigrid V-cycle over the MSC-method for our Fourier-FEM problem, and numerically show the convergence of multigrid V-cycle for curl problems with solutions in $H_r(curl^k_{rz};\ \Omega)$.

We then compared the results for multigrid V-cycle for various Fourier modes. We tested it for $k = 1, 2, 3, 4, 5, 100, 250, 500, 1000$ up to mesh level 8.

Mesh Level	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
2	0.14	0.11	0.08	0.05	0.06
3	0.17	0.14	0.12	0.11	0.09
4	0.26	0.23	0.19	0.16	0.15
5	0.30	0.30	0.29	0.28	0.27
6	0.32	0.32	0.31	0.31	0.31
7	0.32	0.32	0.32	0.32	0.32
8	0.32	0.32	0.32	0.32	0.32

Mesh Level	$k = 100$	$k = 250$	$k = 500$	$k = 1000$
2	0.20	0.20	0.20	0.20
3	0.19	0.19	0.19	0.19
4	0.16	0.16	0.17	0.17
5	0.15	0.16	0.16	0.16
6	0.14	0.15	0.15	0.15
7	0.12	0.14	0.14	0.15
8	0.20	0.12	0.14	0.14

We see that for the smaller Fourier modes ($k = 1, 2, 3, 4, 5$) the convergence rate stays constant for mesh level 8, with small discrepancies between the earlier mesh levels. As the Fourier modes get larger, the convergence rate decreases slightly compared to smaller Fourier modes, and for all mesh levels stays between 0.10 and 0.20. There are minor discrepancies between the larger Fourier modes, but they mostly remain constant throughout the mesh levels.

4 Conclusions

In this report, we showed numerically that performing Fourier-FEMs over axisymmetric domains in $H_r(curl^k_{rz})$ results in accurate approximations to solutions of equation (24). This equation is very closely related to the axisymmetric time harmonic Maxwell equations, making this analysis useful for applications in electromagnetic problems over axisymmetric domains.

We also demonstrated numerically that the multigrid V-cycle algorithm converges uniformly with respect to meshsize for boundary value problems posed in $H_r(curl^k_{rz}; \Omega)$, where Ω is the unit-square in cylindrical coordinates. We compared the results of multigrid V-cycle to the MSC-algorithm, and saw that while the MSC-method had a linearly increasing number of iterations to convergence, multigrid V-cycle had a constant convergence rate and a constant number of iterations to convergence. Since our analysis was using Fourier-FEMs, we tested the results up to 8 mesh levels for different Fourier modes. We found that larger Fourier modes had slightly smaller convergence rates, and that these rates remained constant for all mesh levels. This result is the first time multigrid has been shown to work numerically on problems posed in $H_r(curl^k_{rz})$ and demonstrates an incredibly efficient method of approximating solutions to equation (24).

We intend to extend our analysis to the axisymmetric time harmonic Maxwell equations and its applications. In this report, we have shown the convergence only numerically; however, we hope to extend our numerical results and prove mathematically that the multigrid V-cycle method is convergent for problems posed in $H_r(curl^k_{rz})$.

5 Link to Code Repository

The Matlab code utilized in this report to compute approximations using the MSC-method and the multigrid V-cycle method can be found at https://github.com/trevorcrupi/fem-multigrid.

6 Appendix

By calculating the basis-functions per single triangles on the mesh, we were able to create small local A matrices. We subsequently mapped these together, using Matlab's sparse matrix function, to form a larger global matrix A. We also used the basis-function information to create our right-hand side vector b.

The global ϕ basis-functions must be calculated according to their piecewise components. To do so, we must calculate each ϕ function for a local space, and utilize these local ϕ functions to build an local A matrix. The local space is one triangle on this mesh. Thus we will have three ϕ functions, each of which corresponds to one of the three nodes of the triangle. This will yield a 3×3 local matrix, A - built with the same method used to build equation (17). To build these local ϕ basis-functions, we must calculate the ϕ coefficients of this local space. These ϕ basis-function coefficients may be calculated on a single triangle in our mesh, utilizing the following method. Let $v_i = (x_i, y_i),\ 1 \leq i \leq 3$ denote each vertex:

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{29}$$

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{30}$$

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{31}$$

Solving equation (29) gives us the ϕ coefficients - a, b, and c - corresponding to the first node of the triangle. Equation (30)'s solution gives us the coefficients corresponding to the second ϕ of the triangle. And equation (31) yields the coefficients corresponding to the third node.

While we calculate our local ϕ coefficients as shown above in equations (29) (30) (31), we calculate our local ψ coefficients with the following method:

$$\begin{bmatrix} V_{e_1}^{2x} - V_{e_1}^{1x} & V_{e_1}^{2y} - V_{e_1}^{1y} & 1 \\ V_{e_2}^{2x} - V_{e_2}^{1x} & V_{e_2}^{2y} - V_{e_2}^{1y} & 1 \\ V_{e_3}^{2x} - V_{e_3}^{1x} & V_{e_3}^{2y} - V_{e_3}^{1y} & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{32}$$

$$\begin{bmatrix} V_{e_1}^{2x} - V_{e_1}^{1x} & V_{e_1}^{2y} - V_{e_1}^{1y} & 1 \\ V_{e_2}^{2x} - V_{e_2}^{1x} & V_{e_2}^{2y} - V_{e_2}^{1y} & 1 \\ V_{e_3}^{2x} - V_{e_3}^{1x} & V_{e_3}^{2y} - V_{e_3}^{1y} & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{33}$$

$$\begin{bmatrix} V_{e_1}^{2x} - V_{e_1}^{1x} & V_{e_1}^{2y} - V_{e_1}^{1y} & 1 \\ V_{e_2}^{2x} - V_{e_2}^{1x} & V_{e_2}^{2y} - V_{e_2}^{1y} & 1 \\ V_{e_3}^{2x} - V_{e_3}^{1x} & V_{e_3}^{2y} - V_{e_3}^{1y} & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{34}$$

Any V is a node belonging to an edge (denoted by the subscript), at the start or end of the node (denoted by the superscript). By subtracting the x and y components of the starting-node from the ending-node, we get the x and y vector-components of said edge.

Solving equation (32) gives us the ψ coefficients - A, B, and C - corresponding to the first edge of the triangle. Equation (33)'s solution gives us the coefficients corresponding to the second ψ of the triangle. And equation (34) yields the coefficients corresponding to the third edge.

References

[1] Douglas N Arnold, Richard S Falk, and Ragnar Winther. Multigrid in h (div) and h (curl). *Numerische Mathematik*, 85(2):197–217, 2000.

[2] David Borthwick. *Introduction to partial differential equations*. Springer, 2017.

[3] Patrick Lacoste. Solution of maxwell equation in axisymmetric geometry by fourier series decompostion and by use of h (rot) conforming finite element. *Numerische Mathematik*, 84(4):577–609, 2000.

[4] Minah Oh. de rham complexes arising from fourier finite element methods in axisymmetric domains. *Computers & Mathematics with Applications*, 70(8):2063–2073, 2015.

[5] Minah Oh. The hodge laplacian on axisymmetric domains and its discretization. *IMA Journal of Numerical Analysis*, 2020. URL: `https://doi.org/10.1093/imanum/draa048`.

[6] Francisco-Javier Sayas. A gentle introduction to the finite element method. *Lecture notes, University of Delaware*, 2008. URL: `https://team-pancho.github.io/documents/anIntro2FEM_2015.pdf`.