MATH 745: Topics in Numerical Analysis

Spectral Element Method

Ashiful Bhuiyan

Instructor: Dr. Bartosz Protas

Submission Date: December 20, 2025

Contents

1 Introduction and Problem Statement **2**

1.1 The Poisson Equation (Boundary Value Problem) 2

1.2 The Wave Equation (Initial Boundary Value Problem) 3

2 Theory **3**

2.1 Weak Formulation 3

2.2 Discretization 4

2.3 Spectral Element Discretization 5

2.4 Discrete Matrix System 6

2.5 Global Assembly and Temporal Discretization 8

2.6 Implementation Strategy 8

3 Computational Results and Discussion **9**

3.1 Validation and Accuracy 9

3.2 Convergence Analysis 10

3.2.1 h-Refinement (Mesh Refinement) 10

3.2.2 p-Refinement (Spectral Refinement) 10

3.3 Efficiency in Dynamics 11

3.4 Computational Efficiency and Cost Analysis 12

3.5 Stability and Grid Spacing 13

4 Conclusion **14**

1 Introduction and Problem Statement

This project aims at addressing the numerical solution of boundary value problems and initial-boundary value problems in one spatial dimension. The spatial domain of interest is defined as $\Omega = (0, 1)$. The focus on two specific problems: The elliptic Poisson equation and the hyperbolic Wave equation.

Spatial discretization is carried out using a Spectral Element Method (SEM), which employs high-degree piecewise polynomials basis functions with the goal of achieving spectral accuracy. All while retaining the geometric flexibility of finite element methods. For the temporal discretization of the wave equation, we utilize an explicit time-stepping scheme, a standard second-order symplectic scheme (Leapfrog/Center Difference). For future studies implementing a high-order Runge-Kutta method, allowing for a comparative study of stability and accuracy is an interesting topic to explore.

A benchmark is made comparing the methods performance against known analytical solutions for both equations to validate accuracy and convergence. Additionally, we investigate the computational efficiency of the SEM and its convergence behavior relative to theoretical expectations.

Used later in the theory, we define N_{el} as the number of spectral elements used to partition the domain, while p denotes the polynomial degree on each element. The number of spectral elements controls the h-refinement.

Finally, an analysis on the convergence rate under h-refinement (mesh element refinement) and p-refinement (increasing polynomial degree) to investigate the convergence behaviour of the SEM is conducted. Theoretically, SEM is expected to behave like FEM under h-refinement and like global spectral methods under p-refinement [3]. This project seeks to verify whether these distinct convergence regimes appear for the specific Poisson and Wave problems under consideration.Error is also analyzed as a function of computational cost in order to assess the overall efficiency of the method.

1.1 The Poisson Equation (Boundary Value Problem)

We are looking for the scalar field $u(x)$ that satisfies the Poisson equation, subjected to the homogeneous Dirichlet boundary conditions:

$$-\frac{d^2u}{dx^2} = f(x) \quad \text{for} \quad x \in \Omega$$

$$u(0) = 0, \quad u(1) = 0$$

To help with method validation and error analysis, we use the exact solution to create the PDE:

$$u_{\text{exact}}(x) = \sin(\pi x)$$

Substituting this into the equation determines the required source term $f(x)$:

$$f(x) = -\frac{d^2}{dx^2}(\sin(\pi x)) = \pi^2 \sin(\pi x)$$

And so, the precise problem we need to solve numerically is:

$$-u''(x) = \pi^2 \sin(\pi x), \quad u(0) = u(1) = 0$$

1.2 The Wave Equation (Initial Boundary Value Problem)

With the machinery we develop for the Poisson equation, we extend our tools and analysis to the time-dependent scalar wave equation for $u(x,t)$.

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \quad \text{for} \quad (x,t) \in \Omega \times (0,T]$$

where c represents the wave propagation speed, which we set to $c = 1$ to simplify equations. The problem is subject to the homogenous Dirichlet boundary conditions:

$$u(0,t) = 0, \quad u(1,t) = 0 \quad \forall t > 0$$

The system is closed with initial conditions specifying the displacement and velocity at $t = 0$:

$$u(x,0) = u_0(x), \quad \frac{\partial u}{\partial t}(x,0) = v_0(x)$$

Using the exact standing wave solution:

$$u_{\text{exact}}(x,t) = \sin(\pi x)\cos(c\pi t)$$

We derive the corresponding initial conditions:

1. Initial Displacement:
$$u_0(x) = u_{\text{exact}}(x,0) = \sin(\pi x)$$

2. Initial Velocity:
$$v_0(x) = \frac{\partial}{\partial t}\left(\sin(\pi x)\cos(c\pi t)\right)\Big|_{t=0} = -c\pi \sin(\pi x)\sin(0) = 0$$

Now, the task at hand is to numerically advance the system from the initial state (u_0, v_0) while maintaining the boundary constraints. To accomplish this, we first setup the theoretical formulation of the SEM method, and apply the theory to build a model which we use to numerically solve the solutions of the Poisson and Wave equations.

2 Theory

To establish the theoretical model for the SEM method, this project mainly follows the algorithm outlined in Pozrikidis' Intro to Finite and Spectral Element Methods [2].

2.1 Weak Formulation

We begin by deriving the weak formulation for the Poisson equation. The **strong form** is given by:

$$-u''(x) = f(x), \quad x \in \Omega$$

We multiply this equation by an arbitrary test function $v(x)$ from a suitable function space. Specifically the $H_0^1(\Omega)$ space, which is the space of functions with square-integrable first derivatives that go to zero at the boundaries. As long as the test function is smooth "enough" and goes to zero at the boundary, we can proceed.

Integrating over the domain Ω:

$$-\int_0^1 u''(x)v(x)\,dx = \int_0^1 f(x)v(x)\,dx$$

Integrating the left-hand side by parts yields:

$$-\left[u'(x)v(x)\right]_0^1 + \int_0^1 u'(x)v'(x)\,dx = \int_0^1 f(x)v(x)\,dx$$

We have the homogeneous Dirichlet boundary conditions $u(0) = u(1) = 0$, our test functions $v(x)$ must also satisfy $v(0) = v(1) = 0$. Therefore, the boundary term $[u'(x)v(x)]_0^1$ goes to zero. This reduces our strong form into the weak form where the problem becomes a task of finding $u \in H_0^1(\Omega)$ such that for all $v \in H_0^1(\Omega)$:

$$\int_0^1 u'(x)v'(x)\,dx = \int_0^1 f(x)v(x)\,dx$$

For the wave equation, we apply the same method starting with the wave equation:

$$\begin{aligned} u_{tt} &= c^2 u_{xx} \\ -u_{xx} &= -\frac{1}{c^2}u_{tt} \\ f(x,t) &= -\frac{1}{c^2}u_{tt} \end{aligned}$$

Then we get the weak form after integration by parts:

$$\int_0^1 u_x v_x dx = -\frac{1}{c^2}\int_0^1 u_{tt} v dx \tag{1}$$

This is the same weak spatial operator used for the wave equation, where the term $f(x)$ is replaced by the inertial term $-\frac{1}{c^2}\ddot{u}$.

2.2 Discretization

Since we will be computing the solutions computationally, and we have only a finite number of d.o.f, we pick a subspace $V_h \subset V = H_0^1$, spanned by our basis functions $\{\psi_i(x)\}_{i=1}^N$. Then we approximate our solutions

$$u_h = \sum_{j=1}^N U_j \psi_j(x)$$

where $\psi_j(x)$ are known functions of our choice and U_j are scalars we solve for.

We apply the Galerkin method by choosing test functions from the same space V_h. This involves plugging our u_h into the weak form:

$$\int_0^1 u_h'(x)\,\psi_i'(x)\,dx = \int_0^1 f(x)\,\psi_i(x)\,dx.$$

Now substitute $u_h = \sum_j U_j\psi_j$:

LHS:

$$\int_0^1 \left(\sum_{j=1}^N U_j\psi_j'(x)\right)\psi_i'(x)\,dx = \sum_{j=1}^N U_j \int_0^1 \psi_j'(x)\psi_i'(x)\,dx.$$

Define the matrix:

$$A_{ij} = \int_0^1 \psi_j'(x)\psi_i'(x)\,dx, \qquad b_i = \int_0^1 f(x)\psi_i(x)\,dx.$$

Which gets us the algebraic system:

$$\sum_{j=1}^N A_{ij}U_j = b_i \quad \Rightarrow \quad A\mathbf{U} = \mathbf{b}.$$

This is the Galerkin method for transforming our PDE into a linear system.

2.3 Spectral Element Discretization

Our objective now is to decompose the domain $\Omega = (0,1)$ into N_{el} non-overlapping elements $\Omega_e = [x_e, x_{e+1}]$. Within each element, we map the local coordinate $x \in [x_e, x_{e+1}]$ to a reference coordinate $\xi \in [-1,1]$ using the affine mapping:

$$x(\xi) = \frac{1-\xi}{2}x_e + \frac{1+\xi}{2}x_{e+1}$$

This linear affine map allows our Jacobian to be constant on each element, ensures our derivatives scale simply and the quadrature and differentiation are straightforward.

When we do the reference element mapping in this method, we only have to define the basis functions once on $[-1,1]$. This makes the quadrature rules standard in the unit $[-1,1]$ domain and the element matrices can be computed uniformly. Next we introduce the Jacobian to account for the scaling, which we define as:

$$J^e = \frac{dx}{d\xi} = \frac{x_{e+1} - x_e}{2} = \frac{h_e}{2}$$

The Jacobian allows us to do the following two fundamental transformations:

$$\int_{x_e}^{x_{e+1}} g(x)dx = J^e \int_{-1}^1 g(x(\xi))d\xi$$

and

$$\frac{d}{dx} = \frac{1}{J^e}\frac{d}{d\xi}$$

We approximate the solution $u(x)$ on each element as a polynomial expansion of degree p. We choose the Lagrange polynomials supported on the Gauss-Lobatto-Legendre (GLL) quadrature nodes as our basis functions.

To be more precise, $\psi_j(\xi)$ are the Lagrange interpolation polynomials defined as:

$$\psi_j(\xi_i) = \delta_{ij}$$

where the $\{\xi_i\}$ are the GLL nodes. If $\psi_i(\xi)$ represents the i-th Lagrange polynomial, the numerical solution u_h on an element is:

$$u_h(\xi) = \sum_{j=0}^{N} u_j \psi_j(\xi)$$

where $u_j = u_h(\xi_j)$ are the unknown nodal values at the GLL points.

2.4 Discrete Matrix System

Substituting the basis expansion into the weak form and testing with basis functions $v = \psi_i$, we obtain the local elemental matrices. The Stiffness Matrix (K^e) entries correspond to the integral of the derivatives:

$$K_{ij}^e = \int_{\Omega_e} \frac{d\psi_i}{dx}\frac{d\psi_j}{dx} dx = \int_{-1}^{1} \left(\frac{d\psi_i}{d\xi}\frac{d\xi}{dx}\right)\left(\frac{d\psi_j}{d\xi}\frac{d\xi}{dx}\right) J^e d\xi$$

Substituting the Jacobian $J^e = \frac{dx}{d\xi} = \frac{x_{e+1}-x_e}{2} = \frac{h_e}{2}$:

$$\begin{aligned} K_{ij}^e &= \int_{-1}^{1} \left(\frac{d\psi_i}{d\xi}\frac{1}{J^e}\right)\left(\frac{d\psi_j}{d\xi}\right) d\xi \\ K_{ij}^e &= \frac{2}{h_e}\int_{-1}^{1} \psi_i'(\xi)\psi_j'(\xi) d\xi \end{aligned}$$

The Mass Matrix (M^e) entries correspond to the L^2 inner product:

$$\begin{aligned} M_{ij}^e &= \int_{\Omega_e} \psi_i \psi_j dx \\ &= \int_{\Omega_e} \psi_i \psi_j d\xi \cdot \frac{dx}{d\xi} \\ &= \frac{h_e}{2}\int_{-1}^{1} \psi_i(\xi)\psi_j(\xi) d\xi \end{aligned}$$

A key advantage of SEM using GLL nodes is the use of GLL quadrature for integration. Since the Lagrange basis is defined on the quadrature points, $\psi_i(\xi_k) = \delta_{ik}$. This results in a diagonal mass matrix, which significantly reduces computational cost for explicit time-stepping.

Our logic for this goes as follows:

1. Approximate the mass matrix integral using GLL quadratures

$$\int_{-1}^{1} f(\xi)d\xi \approx \sum_{k=0}^{N} w_k f(\xi_k)$$

2. Apply this to the mass matrix

$$M_{ij}^e \approx \frac{h_e}{2} \sum_{k=0}^{N} w_k \psi_i(\xi_k)\psi_j(\xi_k)$$

3. If we then use the interpolation property $\psi_i(\xi_k)\psi_j(\xi_k) = \delta_{ik}\delta_{jk}$ we get:

$$M_{ij}^e \approx \frac{h_e}{2} w_i \delta_{ij}$$

So we get a mass matrix that is diagonal under quadrature, but not exactly diagonal analytically.

Our explicit time-stepping will require solving:

$$\mathbf{M}\ddot{\mathbf{U}} = (\text{discrete forces}).$$

where the double dot represents second time derivative (Newton notation).

If M is diagonal we find that inversion is trivial and the cost is only $O(N)$. Otherwise inversion would cost $O(N^3)$ or with iterative solves.

With $\mathbf{F}$ as the nodal values and the integral evaluated by GLL quadrature, we can construct $\mathbf{M}$

Comparing the Pure Galerkin form (integral RHS)

$$\mathbf{K}\mathbf{U} = \mathbf{F}, \qquad F_i = \int f\psi_i \, dx$$

to the SEM nodal/quadrature form

$$\mathbf{K}\mathbf{U} = \mathbf{M}\mathbf{F}, \qquad F_i = f(x_i)$$

So we have it that when the load integral is evaluated using GLL quadrature, the discrete right-hand side can be written in the form $\mathbf{MF}$, where $\mathbf{F}$ contains the nodal values of the source term. This representation is equivalent to the pure Galerkin form $\mathbf{KU} = \mathbf{F}$ but is computationally advantageous due to the diagonal structure of the mass matrix.

So finally, the load vector satisfies $F_i \approx \sum_k w_k f(\xi_k)\psi_i(\xi_k) = w_i f(\xi_i)$ hence

$$\mathbf{F}_{\text{integral}} \approx \mathbf{M}\mathbf{F}_{\text{nodal}}.$$

2.5 Global Assembly and Temporal Discretization

The local element matrices are assembled into global mass and stiffness matrices, denoted by $\mathbf{M}$ and $\mathbf{K}$. Homogeneous Dirichlet boundary conditions are enforced by eliminating the boundary degrees of freedom from the global system.

Since we know $U_0 = u(0) = 0, U_{N_g} = u(1) = 0$, where N_g denotes the last global node, we remove these fixed entries from our state vectors and solve only for the interior nodal values. This is especially crucial for the wave equation as if the boundary d.o.f are kept in the system, they will be evolved with time, violating the imposed BC.

For the Poisson equation, the resulting discrete system is

$$\mathbf{KU} = \mathbf{MF},$$

where $\mathbf{U}$ is the vector of nodal unknowns and $\mathbf{F}$ represents the source term evaluated at the GLL nodes. This form naturally arises when the load integral is evaluated using GLL quadrature and the source term is represented by its nodal values.

For the wave equation, spatial discretization gives us the following system

$$\mathbf{M}\frac{d^2\mathbf{U}}{dt^2} + c^2\mathbf{KU} = 0.$$

Time integration is performed using a second-order central difference (leapfrog) scheme. Discretizing time with step size Δt, the update rule is

$$\mathbf{M}\frac{\mathbf{U}^{n+1} - 2\mathbf{U}^n + \mathbf{U}^{n-1}}{\Delta t^2} = -c^2\mathbf{K}\mathbf{U}^n.$$

Solving for $\mathbf{U}^{n+1}$ gives

$$\mathbf{U}^{n+1} = 2\mathbf{U}^n - \mathbf{U}^{n-1} - \Delta t^2 c^2 \mathbf{M}^{-1}\mathbf{K}\mathbf{U}^n.$$

Because the mass matrix $\mathbf{M}$ is diagonal when GLL quadrature is used, the inversion $\mathbf{M}^{-1}$ is trivial, making this explicit time-stepping scheme computationally highly efficient.

2.6 Implementation Strategy

The theoretical formulation derived above is translated into a modular Python implementation using an Object-Oriented Programming (OOP) approach. A single class, `SEM1D`, encapsulates the core spectral element operations, ensuring consistency between the mesh generation, basis construction, and matrix assembly steps.

Grid Generation

Upon initialization, the solver computes the GLL nodes and weights ξ_i, w_i on the reference element $[-1, 1]$ using the roots of the derivative of the Legendre polynomial $P'_N(\xi)$. These are then mapped to the physical domain Ω_e via the affine transformation $x(\xi)$ described in Section 2.3.

Matrix Assembly

The global stiffness (K) and mass (M) matrices are constructed by iterating over each spectral element. We compute the local elemental matrices K^e and M^e using the reference derivative matrix $\hat{D}$ and the diagonal GLL weights. These local contributions are then accumulated into the global system via a connectivity map that links local node indices to unique global degrees of freedom, ensuring C^0 continuity across element boundaries.

Solver Design

For the Poisson equation, boundary conditions are enforced by partitioning the system and solving for the free degrees of freedom using a direct linear solver. For the Wave equation, the efficiency of the method is maximized by exploiting the diagonal structure of the mass matrix M. This allows the explicit Leapfrog time-stepping scheme to proceed without expensive matrix inversions, as the acceleration term can be computed via simple element-wise operations ($a = M^{-1}F_{net}$).

Error Metric and Convergence Criteria

To quantify the accuracy of the Spectral Element approximation, we evaluate the difference between the numerical solution u_h and the exact analytical solution u_{exact}. In this study, we utilize the discrete infinity norm (L_∞), which measures the maximum pointwise error across all nodal points in the domain:

$$||E||_\infty = \max_{1 \leq i \leq N_{dof}} |u_h(x_i) - u_{\text{exact}}(x_i)|$$

This metric provides a strict assessment of accuracy by capturing the worst-case deviation in the domain. We analyze the behavior of this error under two refinement strategies: h-refinement, where the mesh size h is reduced while keeping the polynomial degree p fixed, and p-refinement, where p is increased on a fixed mesh. The convergence rate is determined by the slope of the error curve on a logarithmic scale, allowing us to verify the theoretical algebraic ($O(h^{p+1})$) and exponential ($O(C^{-N})$) convergence regimes respectively.

3 Computational Results and Discussion

To verify the accuracy and efficiency of the SEM method, we implemented the algorithm in Python and tested it against the known analytical solutions for both the Poisson and Wave equations. The implementation utilized the model we built in section 2 with Gauss-Lobatto-Legendre (GLL) quadrature for integration and Lagrange polynomials for basis reconstruction.

3.1 Validation and Accuracy

We first validated the solver using the method of manufactured solutions on the domain $\Omega = [0, 1]$.

- ▷ Poisson Equation: We solved $-u''(x) = \pi^2 \sin(\pi x)$ with $u(0) = u(1) = 0$.

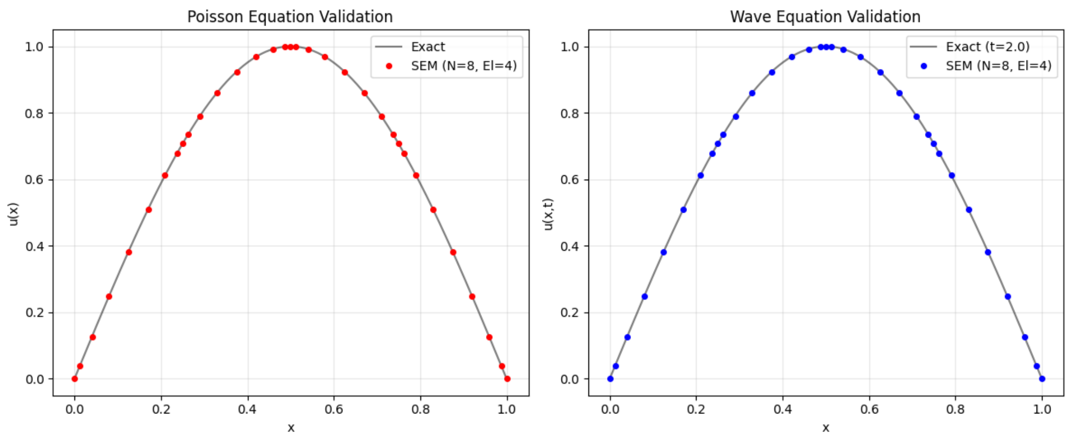

Figure 1: Validation plot showing SEM solution vs Exact solution for Poisson and Wave equations

- ▷ Wave Equation: We solved $u_{tt} = u_{xx}$ with initial condition $u(x, 0) = \sin(\pi x)$ and $v(x, 0) = 0$, which corresponds to the standing wave solution $u(x, t) = \sin(\pi x)\cos(\pi t)$.

Using a polynomial order of $p = 8$ and just $N_{el} = 4$ elements, the method achieved near machine-precision accuracy. The L_∞ error norms obtained were:

- ▷ Poisson Error: $\approx 5.53 \times 10^{-14}$
- ▷ Wave Equation Error ($t = 2.0$): $\approx 8.31 \times 10^{-12}$

These results confirm the correct implementation of the stiffness and diagonal mass matrices, as well as the explicit Leapfrog time-stepping scheme.

3.2 Convergence Analysis

A key feature of the Spectral Element Method is its dual convergence properties. We investigated both h-refinement and p-refinement regimes.

3.2.1 h-Refinement (Mesh Refinement)

In this study, we fixed the polynomial order at $p = 3$ and varied the number of elements (N_{el}) from 2 to 32.

As shown in Figure 2, the error decreases algebraically as the mesh is refined. On a log-log plot, this convergence appears linear with a slope dictated by the polynomial order. This regime mimics traditional Finite Element Method (FEM) behavior, where acuracy is improved by reducing the grid size h. We see that the error follows $O(h^{p+1})$ behavior. As expected for $p = 3$, we observe a convergence rate of approximately $O(h^4)$

3.2.2 p-Refinement (Spectral Refinement)

Here, we fixed the mesh size at $N_{el} = 2$ elements and increased the polynomial degree p from 2 to 12.

Figure 3 demonstrates spectral convergence. The error drops exponentially (appearing as a steep straight line on a semi-log plot) until it hits the machine precision floor ($\approx 10^{-14}$)

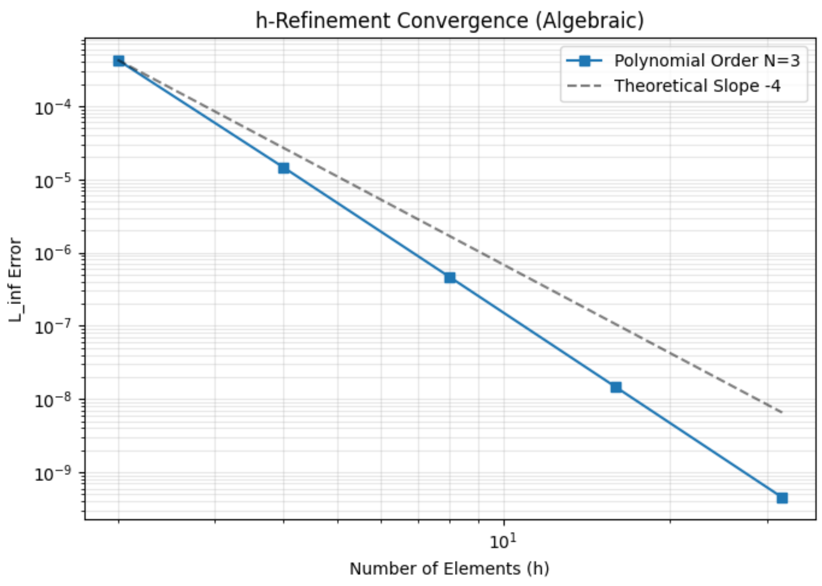

Figure 2: h-refinement convergence plot (Semi-Log scale)

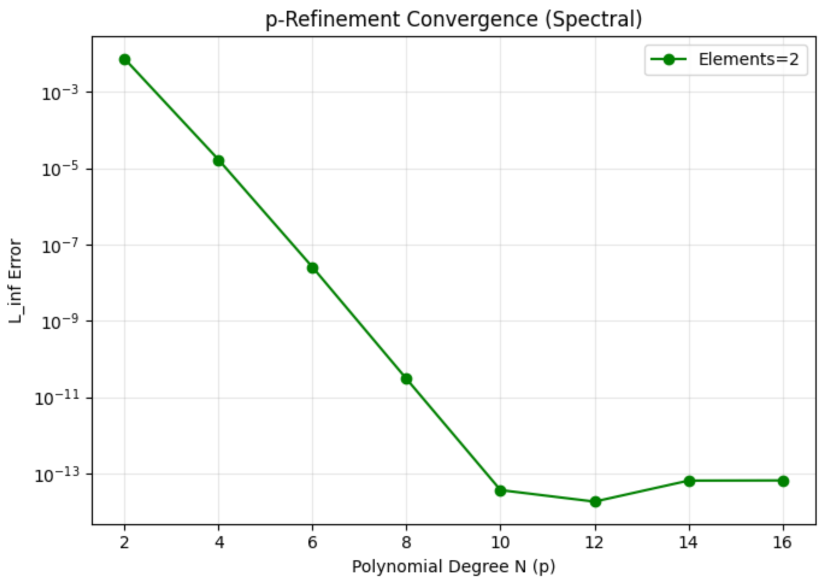

Figure 3: p-refinement convergence plot (Semi-Log scale)

at $p = 8$.This confirms the efficiency of SEM methods for smooth problems. We achieved 10^{-14} error with only moderate degrees of freedom, whereas low-order methods would require significantly finer meshes to reach comparable accuracy.

3.3 Efficiency in Dynamics

For the wave equation, the use of GLL quadrature resulted in a diagonal mass matrix. This had a profound impact on computational efficiency. The explicit Leapfrog scheme requires solving $M\mathbf{a} = \mathbf{F}$ at every time step. Because M is diagonal, its inverse is trivial to compute, reducing the linear solve to simple element-wise multiplication. This allowed us to run thousands of time steps ($dt \approx 1.25 \times 10^{-3}$) rapidly while maintaining stability and high accuracy.

3.4 Computational Efficiency and Cost Analysis

To rigorously assess the performance of the SEM methods, we conducted a direct comparison of computational cost versus accuracy. While theoretical convergence rates (Sections 3.2.1 and 3.2.2) describe the rate of error reduction per degree of freedom, they do not account for the actual runtime costs associated with assembling matrices and solving the system.

We measured the wall-clock execution time required to solve the Poisson problem under two distinct refinement strategies:

▷ h-Refinement (Mesh Refinement): The polynomial order was fixed at a low degree ($p = 4$), and the number of elements was doubled successively from 2 to 64

▷ p-Refinement (Spectral Refinement): The mesh was fixed at a coarse resolution ($N_{el} = 4$), and the polynomial degree was increased from $p = 2$ to $p = 14$.

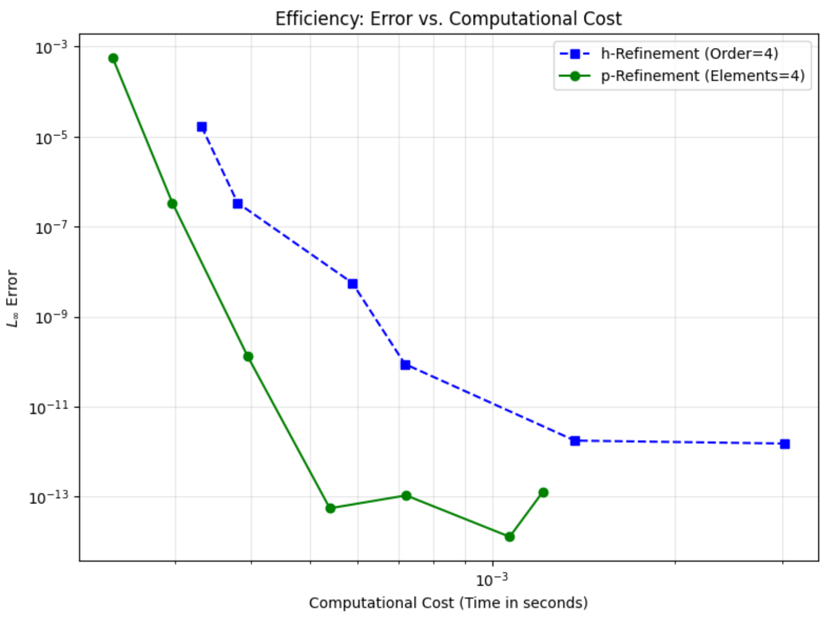

Figure 4: Comparing the efficiency between mesh and spectral refinement.

As illustrated in figure 4, we can see that the two strategies exhibit fundamentally different efficiency profiles.

Firstly and clearly, the spectral refinement is superior. The p-refinement curve (green) demonstrates a significantly steeper slope than the h-refinement curve (blue). This confirms that for smooth problems, increasing the polynomial order is far more efficient than refining the mesh. For example, achieving an error of $\approx 10^{-10}$ required significantly less computational time using p-refinement compared to h-refinement, which would require a prohibitive number of elements to reach the same precision with $p = 4$.

Next we can see a difference between the algebraic vs. exponential cost. The h-refinement strategy shows an algebraic relationship where the cost increases linearly with the number of elements, but the error only decreases at a fixed algebraic rate ($O(h^{N+1})$). Compared to the p-refinement which exploits the spectral convergence property, where the error decays exponentially while the matrix size (and thus computational cost) grows only moderately.

To see which method hits machine precision first, it is important to see the plots in terms of degrees of freedom. This allows us to assess the memory efficiency and storage requirements of this method. In the 1D Spectral Element Method with homogeneous Dirichlet boundary conditions, the total degrees of freedom correspond to the number of interior nodes: $N_{dof} = (N_{el} \times N) - 1$. We compared two strategies once again.

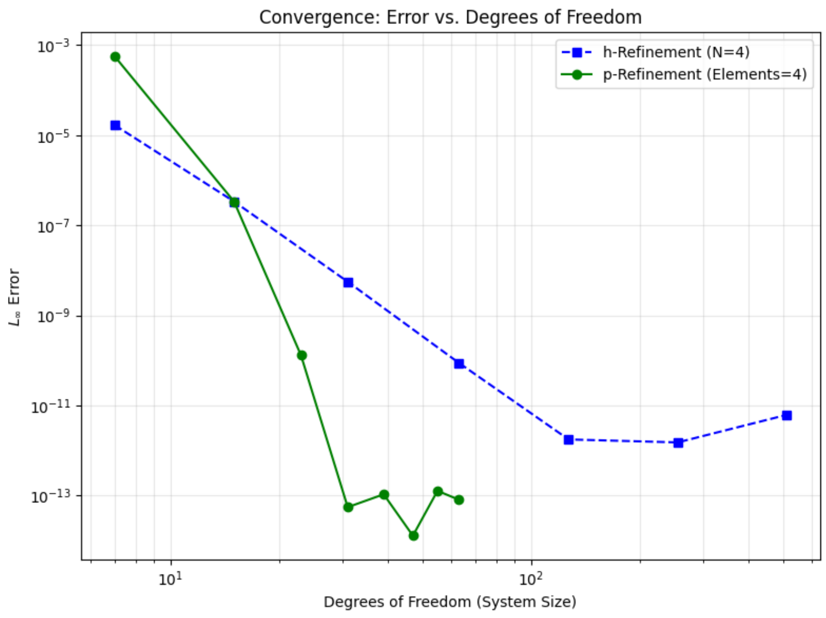

Figure 5: Comparing the efficiency between mesh and spectral refinement while looking at d.o.f.

The p-refinement strategy (green line) exhibits a ”waterfall” convergence profile. It achieves machine precision ($\approx 10^{-14}$) with fewer than 60 DOFs. In contrast, the h-refinement strategy (blue line) follows a linear algebraic slope. To achieve even a moderate error of 10^{-8}, h-refinement required significantly more degrees of freedom (¿ 200) compared to p-refinement.

This comparison highlights that for problems with smooth solutions, p-refinement is vastly more efficient in terms of information density. It captures the solution structure with a fraction of the parameters required by low-order h-refinement, leading to smaller system matrices and reduced memory consumption.

3.5 Stability and Grid Spacing

A known limitation of the Spectral Element Method with explicit time-stepping is the strict restriction on the time step size Δt. Unlike uniform grids where Δx is constant, the Gauss-Lobatto-Legendre (GLL) nodes used in SEM are clustered near the element boundaries. As we increase the polynomial degree p, these boundary nodes get closer together at a rate of $O(p^{-2})$

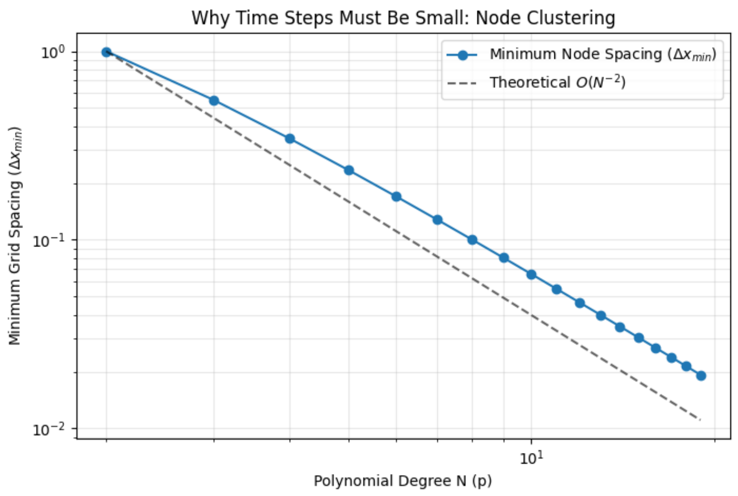

Figure 6: Comparing the efficiency between mesh and spectral refinement while looking at d.o.f.

We verified this geometric property in Figure 6, which plots the minimum distance between nodes as p increases. Because the grid spacing shrinks quadratically, our time step must also shrink quadratically ($\Delta t \propto p^{-2}$) to maintain stability. This explains why high-order simulations require very small time steps compared to low-order methods, even if the total number of unknowns is the same.

4 Conclusion

In this project, we developed and implemented a one-dimensional Spectral Element Method (SEM methods) for both elliptic and hyperbolic partial differential equations, specifically the Poisson and Wave equations on a bounded domain with homogeneous Dirichlet boundary conditions. Starting from a rigorous weak formulation, we constructed the SEM discretization using high-order Lagrange polynomials defined on Gauss-Lobatto-Legendre (GLL) nodes, enabling both accurate spatial approximation and efficient numerical integration.

For the Poisson problem, the SEM demonstrated excellent accuracy, achieving near machine-precision error with only a small number of elements and moderate polynomial degree. The convergence studies clearly revealed the dual nature of SEM method: algebraic convergence under h-refinement, consistent with classical finite element behavior, and exponential (spectral) convergence under p-refinement for smooth solutions. These results align with theoretical expectations and confirm the correct implementation of the stiffness operator and numerical quadrature.

For the Wave equation, the use of GLL quadrature resulted in a diagonal mass matrix, which proved crucial for computational efficiency. This structure allowed for an explicit time integration scheme via the Leapfrog method, avoiding costly linear solves at each time step. However, we also confirmed that the clustering of GLL nodes near element boundaries imposes a strict stability constraint, requiring the time step to scale as $O(N^{-2})$. Despite this restriction, the numerical solution accurately reproduced the analytical standing wave solution over long integration times, while maintaining stability and low dispersion error. The efficiency gains highlight one of the central strengths of SEM for time-dependent problems when combined with appropriate quadrature and explicit time-stepping.

Overall, this work demonstrates that the Spectral Element Method provides a powerful and flexible framework that combines the geometric adaptability of finite elements with the rapid convergence properties of spectral methods. Even in one spatial dimension, the advantages of SEM method in terms of accuracy per degree of freedom and computational efficiency are evident. These results provide a solid foundation for future extensions to higher-dimensional problems, more complex geometries, and nonlinear or variable-coefficient equations. Most likely, I will employ this to do some physics.

References

[1] R. L. Burden and J. D. Faires, *Numerical Analysis*, 9th ed. (Brooks/Cole, Boston, 2011).

[2] C. Pozrikidis, *Introduction to Finite and Spectral Element Methods Using MATLAB* (Chapman and Hall/CRC, Boca Raton, 2005).

[3] P. E. J. Vos, S. J. Sherwin, and R. M. Kirby, "From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low- and high-order discretisations," *Journal of Computational Physics*, vol. 229, no. 14, pp. 5161–5181, 2010. `https://doi.org/10.1016/j.jcp.2010.03.031`