Analytic Tools for the Finite Element Method on Parabolic Equations

Geneva Porter, SDSU Spring 2020
Numerical Partial Differential Equations
Dr. Uduak George, Applied Mathematics

7 May, 2020

The finite element method (FEM) is a technique used to solve many different types of partial differential equations on a variety of meshes. Almost always implemented numerically, the FEM has several practical applications in applied mathematics. This report will detail one particular application, solving a parabolic reaction-diffusion equation on a surface geometry, and discuss ways in which the theoretical application is valid. Then, we will compare the analytic solution on the surface of a sphere with a solution using the FEM. The proceeding error analysis will examine to solution L2 and H1 norms, as well as the difference when mesh density changes.

1 The Surface Finite Element Method

Solving the Schnakenberg equations using the finite element method uses the Laplace-Beltrami operator for a 2D domain in a 3D space. The discretized space is represented numerically by a linear system of matrices, and for parabolic problems, the square matrices are conveniently formatted as symmetric and positive definite. There are several analyses on this process already [**?**] [**?**], so only a brief overview will be given here.

To begin, recall (**??**), the system of differential equations on the 2D domain Γ:

$$\frac{\partial u}{\partial t} - \Delta_\Gamma u = \gamma\ f(u,v) \qquad \frac{\partial v}{\partial t} - \delta\Delta_\Gamma v = \gamma\ g(u,v) \qquad \text{with} \qquad u = v = 0 \ \text{ on } \ \partial\Gamma \tag{1}$$

We treat this as a closed system; therefore there in no flux through the surface. The first step in applying the surface finite element method is to multiply the equations by a test function. The goal of the test function is to give a value to one neighborhood in the discretized domain and return zero for all other neighborhoods. The test function φ must satisfy:

$$\varphi\frac{\partial u}{\partial t} - \varphi\Delta_\Gamma u = \varphi\gamma f(u,v) \qquad \text{and} \qquad \varphi\frac{\partial v}{\partial t} - \varphi\delta\Delta_\Gamma v = \varphi\gamma g(u,v) \tag{2}$$

We continue by using the *weak formulation* method by integrating:

$$\int_\Gamma \varphi\frac{\partial u}{\partial t} - \int_\Gamma \varphi\Delta_\Gamma u = \int_\Gamma \gamma\varphi f(u,v) \qquad \int_\Gamma \varphi\frac{\partial v}{\partial t} - \int_\Gamma \varphi\delta\Delta_\Gamma v = \int_\Gamma \gamma\varphi g(u,v) \tag{3}$$

Note that we can use integration by parts here, and substitute a more easily computed term for Δ_Γ in this context. Green's Theorem states:

$$\int_{\delta\Gamma} a\cdot\nabla_\Gamma b = \int_\Gamma a\cdot\Delta_\Gamma b + \int_\Gamma \nabla_\Gamma a\cdot\nabla_\Gamma b \tag{4}$$

Since the flux on the boundary $\partial\Omega$ is zero, we can rewrite (3) as:

$$\int_\Gamma \varphi\frac{\partial u}{\partial t} + \int_\Gamma \nabla_\Gamma\varphi\cdot\nabla_\Gamma u = \gamma\int_\Gamma \varphi f(u,v) \;\; \text{and} \;\; \int_\Gamma \varphi\frac{\partial v}{\partial t} + \delta\int_\Gamma \nabla_\Gamma\varphi\cdot\nabla_\Gamma v = \gamma\int_\Gamma \varphi g(u,v) \tag{5}$$

Now recall the discussion of the Laplace-Beltrami operator in Section **??**. When considering that our domain is not continuous but a mesh of countable elements, we can represent Γ as the discretized space $\mathbb{T}$ with elements K. In this case, $\mathbb{T}$ is a representation of Γ that is partitioned into non-overlapping quadrilaterals. Using this notation, we approximate the system as:

$$\begin{gathered}\sum_{K\in\mathbb{T}}\int_K \varphi\frac{\partial u}{\partial t} + \sum_{K\in\mathbb{T}}\int_K \nabla_K\varphi\cdot\nabla_K u = \gamma\sum_{K\in\mathbb{T}}\int_K \varphi f(u,v) \\ \sum_{K\in\mathbb{T}}\int_K \varphi\frac{\partial v}{\partial t} + \delta\sum_{K\in\mathbb{T}}\int_K \nabla_K\varphi\cdot\nabla_K v = \gamma\sum_{K\in\mathbb{T}}\int_K \varphi g(u,v)\end{gathered} \tag{6}$$

Because our domain is a surface, we must use the tangential gradients when computing the spatial integral (as opposed to the standard gradients for a bulk volume). Recall that the matrices here are symmetric and positive definite. In this context, the tangential gradient is defined for each variable as:

$$\nabla_K u = D\mathbf{x}_K G_K^{-1}\nabla u \qquad \nabla_K v = D\mathbf{x}_K G_K^{-1}\nabla v \qquad \nabla_K \varphi = D\mathbf{x}_K G_K^{-1}\nabla \varphi = \nabla\varphi^T G_K^{-1} D\mathbf{x}_K^T$$

Furthermore, we can express the spatial elements in terms of a reference element $\hat{K} = [0,1]^2$, which is beneficial for efficiency in numerical implementation. The reference element is derived from the discretization of the domain mesh. Since our surface mesh is made of quadrilaterals, the reference element will be a unit square. The discretized space can be expressed in terms of the reference element and simplified using the relation $G_k = D\mathbf{x}_K^T D\mathbf{x}_K$.

$$\begin{aligned}\int_K \nabla_K\varphi \cdot \nabla_K u &= \int_{\hat{K}} \left(\nabla\varphi^T G_K^{-1} D\mathbf{x}_K^T\right) D\mathbf{x}_K G_K^{-1}\nabla u = \int_{\hat{K}} \nabla\varphi^T G_K^{-1}\nabla u \\ \int_K \nabla_K\varphi \cdot \nabla_K v &= \int_{\hat{K}} \left(\nabla\varphi^T G_K^{-1} D\mathbf{x}_K^T\right) D\mathbf{x}_K G_K^{-1}\nabla v = \int_{\hat{K}} \nabla\varphi^T G_K^{-1}\nabla v\end{aligned} \tag{7}$$

It is prudent to note that this process is referred to as *triangulation*, named from the standard practice in computational mesh generation, which creates surfaces with triangular elements. For this model's validation, we use quadrilaterals within a spherical manifold refined 5 times, yielding a discretized surface with 6×4^5 quadrilaterals.

The triangulation process lends itself to numerical representation in vector form. Vertices on the mesh are assigned to positions in a vector, and can be solved as a linear system. With the discretization of space, the variables u and v, along with the test function φ, must in turn be discretized. Below is our discretized approximation for u and v, with new variables explained below:

$$u \approx u_K = \sum_j \vartheta_j U_j \qquad v \approx v_K = \sum_j \vartheta_j V_j \tag{8}$$

We now introduce the basis function ϑ_j. The basis function is similar to the linear algebra standard: a piecewise polynomial whose purpose is to assign a value to a vertex with index j, interpolate values for the vertices that share an edge with vertex j, and return zero for all other vertices. We create a different basis function for each node on the mesh. In "classical" FEM formulations, low-degree polynomials are used for each basis function. Predictably, higher degree polynomials result in both increased accuracy and increased computing time. For this thesis, a second degree polynomial will be used for each basis function. U_j and V_j are unknown coefficients, which are treated as variables.

The *deal.ii* library produces a triangulated model with a few simple commands. The use of tangential gradients for surface calculations is more tedious, requiring several nested loops through each vertex on the mesh. References and comments in the code implementing these triangulation procedures can be found in Section **??**, and the complete code is in Appendix **??**.

Our spatially discretized system follows (with the tangential gradient notation omitted for clarity):

$$\begin{aligned}
\sum_{K\in\mathbb{T}}\sum_{j}\int_K \varphi_i\cdot\vartheta_j\left[\frac{\partial U_j}{\partial t}\right]+\sum_{K\in\mathbb{T}}\sum_{j}\int_K \nabla_K\varphi_i\cdot\nabla_K\vartheta_j\,[U_j] &= \gamma\sum_{K\in\mathbb{T}}\sum_{j}\int_K \varphi_i f_K(U_j,V_j)\\
\sum_{K\in\mathbb{T}}\sum_{j}\int_K \varphi_i\cdot\vartheta_j\left[\frac{\partial V_j}{\partial t}\right]+\delta\sum_{K\in\mathbb{T}}\sum_{j}\int_K \nabla_K\varphi_i\cdot\nabla_K\vartheta_j\,[V_j] &= \gamma\sum_{K\in\mathbb{T}}\sum_{j}\int_K \varphi_i g_K(U_j,V_j)
\end{aligned} \tag{9}$$

With these approximations, the test functions φ_i can have a solution for the system at each vertex. For this thesis, it will be sufficient to state that a unique solution to the above system exists. A detailed explanation can be found in [**?**], which states that the proof is a direct application of the Lax-Milgram Theorem. For ease of communication, we will simplify the notation here as follows:

$$\sum_{K\in\mathbb{T}}\sum_{j}\int_K x\cdot y=\Big(\,x,y\,\Big)$$

Now our system is more easily examined when written as

$$\begin{aligned}
\Big(\,\varphi_i,\vartheta_j\,\Big)\frac{\partial U_j}{\partial t}+\Big(\,\nabla_K\varphi_i,\nabla_k\varphi_j\,\Big)U_j &= \gamma\Big(\,\varphi_i, f_K(U_j,V_j)\,\Big)\\
\Big(\,\varphi_i,\vartheta_j\,\Big)\frac{\partial V_j}{\partial t}+\delta\Big(\,\nabla_K\varphi_i,\nabla_k\varphi_j\,\Big)V_j &= \gamma\Big(\,\varphi_i, g_K(U_j,V_j)\,\Big)
\end{aligned} \tag{10}$$

Here, it is useful to explain the expansion of the functions f_K and g_K, the discretized versions of the reaction equations seen in (**??**). Because the solutions are in vector format, we must treat the nonlinear term piecewise; that is, multiply each vector term according to its position in the vector.

Since the basis functions need only satisfy the linear system, we want it to be as simple as possible. To accomplish this, it is not necessary to create nonlinear terms for the basis function itself. We need only use it in the first degree when attached to the nonlinear term. For example, using ϑ_j^3 as the basis function term for $U_j^2 V_j$ is not strictly necessary; using ϑ_j alone will suffice. In addition, α and β are transformed into $\mathbf{a}$ and $\mathbf{b}$, which are simply mono-valued vectors corresponding to the size of the basis function vectors. We can now expand the f_K and g_K terms as follows:

$$\begin{aligned}\Big(\varphi_i, f_K\Big) &= \Big(\varphi_i, \alpha - u_K + u_K^2 v_K\Big) = \Big(\varphi_i, \mathbf{a}\Big) - \Big(\varphi_i, \varphi_j\Big) \cdot U_j + \Big(\varphi_i, \varphi_j\Big) \cdot U_j^2 V_j \\ \Big(\varphi_i, g_K\Big) &= \Big(\varphi_i, \beta - u_K^2 v_K\Big) = \Big(\varphi_i, \mathbf{b}\Big) - \Big(\varphi_i, \varphi_j\Big) \cdot U_j^2 V_j\end{aligned} \tag{11}$$

Note that the notation $U_j^2 V_j$ represents the nonlinear term vectors multiplied piecewise. Rewriting Equation (10) using the substitutions above yields:

$$\begin{aligned}&\Big(\varphi_i, \vartheta_j\Big)\frac{\partial U_j}{\partial t} + \Big(\nabla_K \varphi_i, \nabla_k \varphi_j\Big) U_j = \gamma\Big[\Big(\varphi_i, \mathbf{a}\Big) - \Big(\varphi_i, \varphi_j\Big) \cdot U_j + \Big(\varphi_i, \varphi_j\Big) \cdot U_j^2 V_j\Big] \\ &\Big(\varphi_i, \vartheta_j\Big)\frac{\partial V_j}{\partial t} + \delta\Big(\nabla_K \varphi_i, \nabla_k \varphi_j\Big) V_j = \gamma\Big[\Big(\varphi_i, \mathbf{b}\Big) - \Big(\varphi_i, \varphi_j\Big) \cdot U_j^2 V_j\Big]\end{aligned} \tag{12}$$

We can now use matrix notation for each summation, using the following terms:

$$\mathbf{M} = (\varphi_i, \varphi_j) \qquad \mathbf{L} = (\nabla\varphi_i, \nabla\varphi_j) \qquad \mathbf{A} = (\varphi_i, \mathbf{a}) \qquad \mathbf{B} = (\varphi_i, \mathbf{b})$$

Common nomenclature dictates that $\mathbf{M}$ is the mass matrix, $\mathbf{L}$ is the Laplace matrix, and $\mathbf{A}$ and $\mathbf{V}$ are the forcing term vectors. The simple form is now:

$$\begin{aligned}\mathbf{M} \cdot \frac{d}{dt}[U_j] + \mathbf{L} \cdot U_j &= \gamma(\mathbf{A} - \mathbf{M} \cdot U_j + \mathbf{M} \cdot U_j^2 V_j) \\ \mathbf{M} \cdot \frac{d}{dt}[V_j] + \delta\mathbf{L} \cdot V_j &= \gamma(\mathbf{B} - \mathbf{M} \cdot U_j^2 V_j)\end{aligned} \tag{13}$$

The spatial discretization is now complete. We can use these results to plug into the temporal discretization, which will be discussed in the next section. We will use the following generalized substitutions:

$$\begin{gathered} u \to \mathbf{M} \cdot U_j \qquad v \to \mathbf{M} \cdot V_j \\ \Delta u \to -\mathbf{L} \cdot U_j \qquad \Delta v \to -\mathbf{L} \cdot V_j \\ \alpha \to \mathbf{A} \qquad \beta \to \mathbf{B} \\ u^2 v \to \mathbf{M} \cdot U_j^2 V_j \end{gathered} \tag{14}$$

2 Implicit-Explicit Time Stepping Scheme

The following time discretization will use both implicit and explicit strategies. There is significant evidence that using a combination of implicit and explicit methods improves the stability and decreases the error in temporal discretization schemes [xxx references]. Once we assemble the linear system, we will substitute the spatial discretization terms discussed in Section 1. To begin, we recall our nondimensionalized system of equations:

$$\frac{\partial u}{\partial t} - \Delta u = \gamma\ (\alpha - u + u^2 v)\ , \qquad \frac{\partial v}{\partial t} - \delta\Delta v = \gamma\ (\beta - u^2 v) \tag{15}$$

First, let's examine the first equation in terms of u. We apply a first order implicit Euler method to the time derivative, with the exception of the non-linear term. This is one of many forms known as the Implicit-Explicit, or IMEX, discretization scheme. Note that k is the time step length and N is the time step index number.

$$\frac{u_N - u_{N-1}}{k} - \Delta u_N = \gamma\ (\alpha - u_N + u_{N-1}^2 v_{N-1}) \tag{16}$$

Separating the unknown values with u_n we get

$$u_N + k\left(\gamma u_N - \Delta u_N\right) = k\gamma\left(\alpha + u_{N-1}^2 v_{N-1}\right) + u_{N-1} \tag{17}$$

From here, we can solve the equation for v using a slightly more implicit scheme. This is possible because when solving this system numerically, we can solve one equation before the other for each time step. Therefore, the value u_N will be known when solving the equation for v. This yields:

$$\frac{v_N - v_{N-1}}{k} - \delta\Delta v_N = \gamma(\beta - u_N^2 v_{N-1}) \tag{18}$$

Again, separating unknown terms yields

$$v_N - k\left(\delta\Delta v_N\right) = k\gamma\left(\beta - u_N^2 v_{N-1}\right) + v_{N-1} \tag{19}$$

In operator form, the system simplifies to

$$\begin{aligned} \left[1 + k\left(\gamma - \Delta\cdot\right)\right] u_n &= k\gamma\left(\alpha + u_{N-1}^2 v_{N-1}\right) + u_{N-1} \\ \left[1 - k\delta\Delta\cdot\right] v_n &= k\gamma\left(\beta - u_{N-1}^2 v_{N-1}\right) + v_{N-1} \end{aligned} \tag{20}$$

Using this combination of implicit and explicit approaches allows a moderately high level of accuracy without a significantly complex scheme. We can apply this temporally discretized system to the spatially discretized result of the finite element method linear system by substituting the results to form a linear system. Combining Equation (20) and the substitutions in (14), we get:

$$\begin{aligned} \left[(1+k\gamma)\mathbf{M}+k\mathbf{L}\right] U_N &= k\gamma\left(\mathbf{A}+\mathbf{M}U_{N-1}^2 V_{N-1}\right)+\mathbf{M}U_{N-1} \\ \left[\mathbf{M}+k\delta\mathbf{L}\right] V_N &= k\gamma\left(\mathbf{B}-\mathbf{M}U_N^2 V_{N-1}\right)+\mathbf{M}V_{N-1} \end{aligned} \tag{21}$$

We now have a linear system of the form $\mathbf{A}x=\mathbf{b}$. The *deal.ii* library has a built-in linear solver, and we utilize the solver's conjugate gradient method for each timestep. For simplicity in further discussion, we will represent the system matrix (left-hand side) and the right hand side matrix as follows:

$$\mathbf{A}=\begin{pmatrix}(1+k\gamma)\mathbf{M}+k\mathbf{L}\\ \mathbf{M}+k\delta\mathbf{L}\end{pmatrix} \qquad \mathbf{b}=\begin{pmatrix}k\gamma\left(\mathbf{A}+\mathbf{M}U_{N-1}^2 V_{N-1}\right)+\mathbf{M}U_{N-1}\\ k\gamma\left(\mathbf{B}-\mathbf{M}U_N^2 V_{N-1}\right)+\mathbf{M}V_{N-1}\end{pmatrix} \tag{22}$$

3 The Spherical Domain

Before examining the measures of error for consistency, convergence, and stability, it is useful to establish some details about the domain in question. We will examine an arbitrary pattern on the surface of the sphere for various mesh densities and time step intervals. Each mesh was created using *deal.ii*'s mesh library.

The mesh itself has a base shape of a cube, and can be refined an arbitrary number of times within a spherical manifold. Upon each refinement, the number of cells in the mesh quadruple, and the maximum diameter among all the cells, h, reduces by about half. We will use h_3 to denote the maximum diameter after 3 refinements from a cube, h_4 for four refinements, and h_5 for 5 refinements. Figure 1 below shows the three mesh densities we will be working with for the error analysis, while Table 1 details the numerical information on each mesh, including number of cells and degrees of freedom.

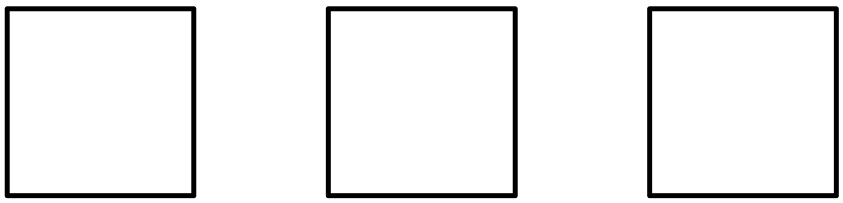

Figure 1. Sphere meshes with density refinements (left to right) for h_3, h_4, and h_5

Table 1. Spherical Mesh Density Values

	h_3	h_4	h_5
Number of Cells	384	1,536	6,144
Degrees of Freedom		6,146	24,578
Maximum Diameter		0.139239	0.0697337

Some visualizations to follow only show data up to $t = 1$. It is important to establish that data collected at $t = 1$ is sufficient to show steady-state dynamics. To do this, we will observe a standard difference norm, as follows:

$$||c|| = \sqrt{\sum_i c_i^2} \tag{23}$$

4 Well-Posedness

We can define a well-posed problem as having three characteristics:

1. A solution for the problem exists
2. The solution is unique
3. The solution changes continuously as the boundary values and initial conditions change.

To verify that the Schnakenberg system meets these requirements, recall the given equations with their boundary conditions and initial values using the variables t, ϕ, and θ:

$$\begin{gathered}
\dot{u} - \Delta_\Gamma u = \gamma f(u,v) \qquad f(u,v) = \alpha - u + u^2 v \\
\dot{v} - \delta\Delta_\Gamma v = \gamma g(u,v) \qquad g(u,v) = \beta - u^2 v \quad \text{with} \\
u(t,\pi,\theta) = u(t,-\pi,\theta) \qquad u(t,\phi,\pi) = u(t,\phi,-\pi) \\
v(t,\pi,\theta) = v(t,-\pi,\theta) \qquad v(t,\phi,\pi) = v(t,\phi,-\pi) \\
\text{and} \qquad u_0 = \alpha + \beta \qquad v_0 = \frac{\beta}{(\alpha+\beta)^2}
\end{gathered} \tag{24}$$

This system is a parabolic reaction-diffusion system, which is guaranteed to be well-posed so long as there are at least 2 boundary conditions and the spatial derivative is of the second degree. We can see that there are indeed two boundary conditions and that the spatial derivatives $\Delta_\Gamma u$ and $\Delta_\Gamma v$ are of the second degree. We know the system is parabolic because it follows the form:

$$\begin{gathered}
A_1 u_{xx} + 2B_1 u_{xt} + C_1 u_{tt} + D_1 u_x + E_1 u_t + F_1 = 0 \\
A_2 v_{xx} + 2B_2 v_{xt} + C_2 v_{tt} + D_2 v_x + E_2 v_t + F_2 = 0
\end{gathered} \tag{25}$$

Here, the spatial derivatives $\Delta_\Gamma u$ and $\Delta_\Gamma u$ are represented by u_{xx} and v_{xx}, respectively. The time derivatives $\dot{u}$ and $\dot{v}$ are likewise represented by u_t and v_t. For both equations, the coefficients corresponding to B and C are equal to zero, so the criteria for parabolic classification $B^2 - AC = 0$ is met. Since the problem is well-posed, we can continue to make other important inferences later on.

5 Consistency

For FDM, consistency occurs when mesh cells and time step size decrease, the truncation error approaches zero. In other words, the discrete system should be a "good" approximation of the partial differential equation system. Normally we would determine consistency by showing:

$$P\phi - P_{k,h}\phi \to 0 \quad \text{as} \quad k,\ h \to 0 \tag{26}$$

However, for nonlinear PDEs, we need a different definition. Applying the FEM to the idea of consistency changes some strategies as well. One way to measure consistency is by examining the local truncation error. Recall the linear system we set up using the FEM-IMEX scheme:

$$\mathbf{A}x = \mathbf{b} \tag{27}$$

To find the truncation error, we apply a simple check after solving the system:

$$\text{Truncation error} = ||\mathbf{b} - \mathbf{A}x|| \tag{28}$$

The truncation error is already an integral part of the algorithm for solving the linear system (in this case, the conjugate gradient method). We can apply a constraint that forces the solver to continue enumerating until the solution gives a truncation error below the desired amount. This feature also allows us to easily control the accuracy to order p, so long as we constrain the truncation error as being less than both $\mathcal{O}(k^p)$ and $\mathcal{O}(h^p)$.

We define order p by the following inequality:

$$||\mathbf{b} - \mathbf{A}x|| \lessapprox h^{p_1} + k^{p_2} \tag{29}$$

We can then say that the numerical scheme is consistent if it is accurate of order $p = (p_1, p_2) > 0$. For this analysis, we constrained the conjugate gradient algorithm to iterate until the truncation error was less than 10^{-20}. Therefore, we can achieve an arbitrarily high consistency order so long as we are willing to sacrifice the computing time needed to achieve it. Since we keep the truncation error at 10^{-20} for all computational runs, almost all permutations of h and k in the proceeding analysis is consistent with an order of accuracy of at least (2,2), *with the exception* of the combination h_2 and $k = 0.01$, which did not converge after 10,000 iterations. Table 2 shows how the number of conjugate gradient iterations decrease as h and k decrease, indicating that the system is more readily consistent as the time and space intervals are refined. Note that the combination the combination h_2 and $k = 0.01$ is left out of the error analysis in future sections.

Table 2. Conjugate Gradient Iterations Needed for Consistency

	$k = 10^{-2}$	$k = 10^{-3}$	$k = 10^{-4}$	$k = 10^{-5}$
h_2	1	1	1	1
h_3	1	1	1	1
h_4	1	1	1	1
h_5	1	1	1	1

6 Convergence

While consistency examines the discretization of a system, convergence implies that the *solution* to the discrete system is a “good” approximation to the *solution* of the partial differential equation system. Now, we cannot measure convergence in the traditional way, because it requires knowledge of the exact solution. The exact solution to the Schnakenberg system on the surface of a spherical domain in unknown. However, we can create a facsimile of the exact solution by producing an output using the smallest time step and cell size that is computationally feasible.

If we call this exact solution facsimile W_E while our approximate solution is W_h, then we can say that the solution converges given that the following criteria is satisfied:

$$\Big|\Big| \, ||W_h|| - ||W_E|| \, \Big|\Big| \lessapprox h^{q_1} + k^{q_2} \quad \text{with} \quad ||W|| = ||w_n - w_{n+1}|| \tag{30}$$

where $n = 1, 2, 3, ...\dim(w)$. So long as $q = (q_1, q_2) > 0$, the solution converges at rate q. The norms are the same as the one defined in (23). We can also show this graphically with the following visualizations:

1. Plotting the difference norm vs. time for various time step solutions at the same cell size
2. Plotting the difference norm vs. time for various cell sizes at the same time step
3. Plotting the difference norm at t=1 vs. timestep size for various cell sizes

7 Stability

Stability in a finite difference system demands that the solution is persistent, that is, small perturbations or errors (such as round-off errors) in the data disappear over time. In addition, it implies that a change of the initial and boundary data leads to a comparable change in the numerical solution. This concept is identical for the finite element method. In fact, the formal definition tells us that our solution *must* be stable, since it is consistent and converges [**?**]:

Theorem 1 (Lax-Richtmyer)**.** *Let W_h be the solution of a numerical method consistent with a well-posed time-dependent problem; in particular, assume that it is accurate of order $p > 0$. Then, if the numerical method is stable, its solution converges with pth-order convergence rate,*

$$\Big|\Big|\, ||W_h|| - ||W_E||\, \Big|\Big| \lessapprox \sum_{m=0}^{n} k \cdot ||\boldsymbol{b} - \boldsymbol{A}x|| \lessapprox h^{p_1} + k^{p_2}, \quad t^n \in [0, T].$$

8 Conclusion