1 Description of System

We consider a system with a Lagrangian,

$$\mathcal{L} = \frac{1}{2}\phi\left(\Box + m_\phi^2\right)\phi + \frac{\gamma}{2}\chi\left(\Box + m_\chi^2\right)\chi + V(\phi,\chi). \tag{1}$$

Where $\Box = \partial_t^2 - \nabla^2$ is the D'Alembertian operator, ∇ is a gradient, ϕ and χ are scalar fields, and $V(\phi,\chi)$ is a nonlinear potential that couples the scalar fields. The ghost appears in this Lagrangian if the scalar fields are assigned oppositely signed kinetic energy terms, $\gamma = -1$.

We find the strong equations of motion

$$\Box\phi + m_\phi^2\phi + \partial_\phi V = 0, \tag{2}$$

$$\Box\chi + m_\chi^2\chi + \gamma\partial_\chi V = 0. \tag{3}$$

Since the original system is second order in time, it is ideal for the weak formulation to introduce auxiliary variables such that our system can be reduced to one that is first order in time.

Let $u = \partial_t\phi$ and $v = \partial_t\chi$. From Eqns. 2 and 3, we have

$$\frac{\partial u}{\partial t} - \nabla^2\phi + m_\phi^2\phi + \frac{\partial V}{\partial \phi} = 0, \tag{4}$$

$$\frac{\partial \phi}{\partial t} - u = 0, \tag{5}$$

$$\frac{\partial v}{\partial t} - \nabla^2\chi + m_\chi^2\chi + \gamma\frac{\partial V}{\partial \chi} = 0, \tag{6}$$

$$\frac{\partial \chi}{\partial t} - v = 0. \tag{7}$$

2 Space-time FEM Formulation

An important step in any FEM is determining the element stiffness matrices. These matrices are found by solving the weak form of the equations of motion. Solving the space-time FEM for the 1+1 case results in 4×4 matrices, the 2+1 case finds 8×8 matrices, and the 3+1 case has 16×16 matrices.

The weak equations are 8, 9, 10, and 11 and we display them in their 2+1 form.

$$K_1 = \int_{X,Y,T} \left(u_t\Psi + \phi_x\Psi_x + \phi_y\Psi_y + m_\phi^2\phi\Psi + V_\phi\Psi\right) dx\,dy\,dt \tag{8}$$

$$K_2 = \int_{X,Y,T} \left(\phi_t\Psi - u\Psi\right) dx\,dy\,dt \tag{9}$$

$$G_1 = \int_{X,Y,T} \left(v_t\Psi + \chi_x\Psi_x + \chi_y\Psi_y + m_\chi^2\chi\Psi + \gamma V_\chi\Psi\right) dx\,dy\,dt \tag{10}$$

$$G_2 = \int_{X,Y,T} \left(\chi_t\Psi - v\Psi\right) dx\,dy\,dt \tag{11}$$

Note that an integration by parts was done over the spatial derivative to move a partial derivative to the test function, Ψ, then, we employed our periodic boundary conditions to remove the boundary term. The test function is a piecewise polynomial function and can take on many different forms. For our purposes, a rectangular (tensor-product) basis function is used over a space–time element. Once the element basis function is determined, it will be inserted into the weak equations. These integrals will be solved over each element and four 8×8 element stiffness matrices will be produced: one time matrix, two spatial stiffness matrices (x and y), and one mass matrix.

2.1 Element Basis Function

The rectangular-prism element basis function is a piecewise polynomial with eight terms,

$$\Psi_i(x, y, t) = b_{1,i}x + b_{2,i}y + b_{3,i}t + b_{4,i}xy + b_{5,i}xt + b_{6,i}yt + b_{7,i}xyt + b_{8,i}. \tag{12}$$

To solve for the unknown coefficients in Eq. 12, we consider the eight corners of a rectangular prism in the (x, y, t) space. In this element, h_x, h_y, and h_t represent the step sizes in x, y, and t:

$$\begin{aligned} h_x &= \frac{x_{\text{final}} - x_{\text{initial}}}{n_x}, \\ h_y &= \frac{y_{\text{final}} - y_{\text{initial}}}{n_y}, \\ h_t &= \frac{t_{\text{final}} - t_{\text{initial}}}{n_t - 1}. \end{aligned}$$

where n_x, n_y, and n_t are the number of nodes in each direction in the entire domain. Thus, n_x, n_y, and n_t control the resolution of the simulation; as the number of nodes grows, the distance between nodes shrinks, and the domain becomes more refined. It should be noted that the (n_x), (n_y), and $(n_t - 1)$ terms in the denominators of the above equations is not a mistake. It must be this way since we implement periodic spatial boundaries but impose no such condition temporally.

As in the 1+1 case, we determine nodal basis functions by requiring that each Ψ_i takes the value 1 at node i and 0 at the other nodes of the element. The eight nodes are represented in Table 1.

Table 1: Representation of local element node coordinates j in 2+1.

j	x	y	t
1	0	0	0
2	h_x	0	0
3	h_x	h_y	0
4	0	h_y	0
5	0	0	h_t
6	h_x	0	h_t
7	h_x	h_y	h_t
8	0	h_y	h_t

With $\xi = x/h_x$, $\tau = t/h_t$, and $\zeta = y/h_y$, this results in the trilinear basis functions:

$$\begin{aligned}
\Psi_1(\xi,\tau,\zeta) &= (1-\xi)(1-\tau)(1-\zeta), \\
\Psi_2(\xi,\tau,\zeta) &= \xi(1-\tau)(1-\zeta), \\
\Psi_3(\xi,\tau,\zeta) &= \xi\tau(1-\zeta), \\
\Psi_4(\xi,\tau,\zeta) &= (1-\xi)\tau(1-\zeta), \\
\Psi_5(\xi,\tau,\zeta) &= (1-\xi)(1-\tau)\zeta, \\
\Psi_6(\xi,\tau,\zeta) &= \xi(1-\tau)\zeta, \\
\Psi_7(\xi,\tau,\zeta) &= \xi\tau\zeta, \\
\Psi_8(\xi,\tau,\zeta) &= (1-\xi)\tau\zeta.
\end{aligned}$$

2.2 Element Stiffness Matrices

Now that we have the basis functions, the element stiffness matrices can be determined. We will explain the set up of the element stiffness matrices, display the matrices conceptually, and then discuss practical implementation into the simulation.

In the three dimensional (2+1) case, the weak form of the equations of motion, equations 8, 9, 10, and 11, are triple integrals over one space–time element in the domain,

$$\begin{aligned}
K_1 &= \int_0^{h_t}\int_0^{h_y}\int_0^{h_x} \left(u_t\Psi + \phi_x\Psi_x + \phi_y\Psi_y + m_\phi^2\phi\Psi + V_\phi\Psi\right)\,dx\,dy\,dt,\\
K_2 &= \int_0^{h_t}\int_0^{h_y}\int_0^{h_x} \left(\phi_t\Psi - u\Psi\right)\,dx\,dy\,dt,\\
G_1 &= \int_0^{h_t}\int_0^{h_y}\int_0^{h_x} \left(v_t\Psi + \chi_x\Psi_x + \chi_y\Psi_y + m_\chi^2\chi\Psi + \gamma V_\chi\Psi\right)\,dx\,dy\,dt,\\
G_2 &= \int_0^{h_t}\int_0^{h_y}\int_0^{h_x} \left(\chi_t\Psi - v\Psi\right)\,dx\,dy\,dt.
\end{aligned}$$

For demonstration purposes, we will discuss the element stiffness matrices pertaining to the linear terms in the weak EOM. The linear terms are the ones that do not include contributions from V or its partial derivatives. This results in 8×8 element stiffness matrices of the form

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,8}\\ \vdots & \ddots & \vdots\\ a_{8,1} & \cdots & a_{8,8}\end{bmatrix}. \tag{13}$$

The time, spatial (x and y), and mass element stiffness matrices come from solving

$$\begin{aligned}
\text{Time}_{i,j} &= \int_0^{h_t}\int_0^{h_y}\int_0^{h_x} \left(\frac{\partial\Psi_i}{\partial t}\Psi_j\right)\,dx\,dy\,dt,\\
\text{Space}_{x i,j} &= \int_0^{h_t}\int_0^{h_y}\int_0^{h_x} \left(\frac{\partial\Psi_i}{\partial x}\frac{\partial\Psi_j}{\partial x}\right)\,dx\,dy\,dt,\\
\text{Space}_{y i,j} &= \int_0^{h_t}\int_0^{h_y}\int_0^{h_x} \left(\frac{\partial\Psi_i}{\partial y}\frac{\partial\Psi_j}{\partial y}\right)\,dx\,dy\,dt,\\
\text{Mass}_{i,j} &= \int_0^{h_t}\int_0^{h_y}\int_0^{h_x} \left(\Psi_i\Psi_j\right)\,dx\,dy\,dt.
\end{aligned}$$

Since these terms are linear in the weak EOM, we are able to calculate their element stiffness matrices one time. This can be done by hand, but we choose to utilize symbolic integration (via SymPy) to compute these matrices and then hard-code them into the simulation.

The nonlinear terms of the weak EOM must be dealt with a little differently, but the gist is the same. Instead of being able to find the element stiffness matrices once at the beginning of the simulation, we must use a Gaussian quadrature routine to approximate the integrals of the potential terms over each element. This is necessary as the solution U changes every iterate, and the nonlinear terms must be updated in the stiffness matrices. This is done inside the `FormResidual()` and `FormJacobian()` routines.

3 Nonlinear Solver: PETSc SNES

In order to computationally handle the nonlinearity of the system, we utilize PETSc's SNES library. The library contains methods, like Newton's method with line search, for solving nonlinear equations of the form

$$\mathcal{F}(U) = 0.$$

Where $\mathcal{F}$ is the nonlinear differential operator, and U is our solution vector.

The two main user-created components of the code are the `FormResidual()` and `FormJacobian()` functions. `FormResidual()` takes as input an approximate guess of the solution vector U. Here, the solution is plugged into the weak form of the EOM. Over each element, the integration is done with the pre-calculated element stiffness matrices for the linear terms, and Gaussian quadrature for the nonlinear terms.

This process creates a residual vector, R. If the norm of the residual vector is near zero, then we have solved $\mathcal{F}(U) \approx 0$, and we are done. If the current U is not acceptable, then we aim to update U in a way that minimizes $\mathcal{F}(U)$.

To do this, `FormJacobian()` takes in the current iterate U_k and then implements and returns $J = \frac{\partial \mathcal{F}}{\partial U}|_{U_k}$. Then, a perturbation of U_k that should decrease the residual R_k is found by solving $J\delta U = -R_k$. Now, we find a new guess by replacing U_k with $U_k + \delta U_k$ and repeat the process.

The only parts of this process that we manually implement are the creation of the residual vector, R, and the Jacobian matrix, J, that are made by solving the weak EOM over each element. These routines will necessarily change from user to user depending on their choice of numerical discretization. PETSc's SNES call handles all the other operations (checking the norm of the residual, solving $J\delta U_k = -R_k$, and updating the iterate according to a line search). See table 3 for a step-by-step process.

SNES Nonlinear Solve (Newton with line search)

```
Input: initial guess $U_0$ (seeded from initial conditions)
Repeat for $k = 0, 1, 2, \dots$
    1. $R_k \leftarrow F(U_k)$   (FormResidual())
    2. If $\|R_k\| \leq \mathrm{rtol}$, stop (converged)
    3. $J_k \leftarrow \dfrac{\partial F}{\partial U}(U_k)$   (FormJacobian())
    4. Solve $J_k\,\delta U_k = -R_k$   (KSP/PC linear solve, PETSc)
    5. Choose step $\alpha \in (0, 1]$ (line search, PETSc)
    6. $U_{k+1} \leftarrow U_k + \alpha\,\delta U_k$   (PETSc)
```

4 Method of Manufactured Solutions MMS

We consider the following coupled partial differential equations (PDEs) in 2+1 dimensions:

$$\phi_{tt} - \phi_{xx} - \phi_{yy} + \phi + V_\phi(\phi, \chi) = 0, \tag{14}$$

$$\chi_{tt} - \chi_{xx} - \chi_{yy} + \chi - V_\chi(\phi, \chi) = 0, \tag{15}$$

where the highly nonlinear potential terms are defined as

$$V_\phi(\phi, \chi) = -2\lambda\, \phi\, \frac{A}{B^{1.5}}$$
$$V_\chi(\phi, \chi) = 2\lambda\, \chi\, \frac{C}{B^{1.5}}$$

with

$$A = \phi^2 - \chi^2 + 1$$
$$B = \left(\phi^2 - \chi^2 - 1\right)^2 + 4\phi^2,$$
$$C = \phi^2 - \chi^2 - 1.$$

Here, λ is a parameter controlling the strength of the nonlinear potential. It is set to 1.0 for all testing in this section.

4.1 Problem Setup and Goal

The aim of our numerical implementation is to solve equations (14)–(15) using a space–time finite element method (FEM) within PETSc with a nonlinear solver (SNES). The code uses a formulation in which each space–time node carries four degrees of freedom:

- $\phi(x, y, t)$ and an auxiliary field $u(x, y, t)$, which is related to the time derivative of ϕ, and
- $\chi(x, y, t)$ and an auxiliary field $v(x, y, t)$, which is related to the time derivative of χ.

To verify our implementation, we use the *method of manufactured solutions* (MMS). The idea behind MMS is to choose a smooth (and nontrivial) manufactured solution for $\phi(x, y, t)$ and $\chi(x, y, t)$, compute all the required derivatives, and then add appropriate forcing functions so that the manufactured solution exactly satisfies the modified PDEs. In our case, we modify the equations to

$$\phi_{tt} - \phi_{xx} - \phi_{yy} + \phi + V_\phi(\phi, \chi) = F_\phi(x, y, t), \tag{16}$$

$$\chi_{tt} - \chi_{xx} - \chi_{yy} + \chi - V_\chi(\phi, \chi) = F_\chi(x, y, t). \tag{17}$$

The goal is to compute these forcing functions for ϕ and χ (F_ϕ and F_χ) so that when the manufactured solution is substituted into the left-hand side of (16) and (17), the LHS exactly matches $F_\phi(x, y, t)$ and $F_\chi(x, y, t)$.

4.1.1 Manufactured Solution

We choose the following manufactured solutions:

$$\phi(x, y, t) = 1 + \cos(\pi x) \cos(\pi y) \cos(\pi t), \tag{18}$$

$$\chi(x, y, t) = 1 + \cos(\pi x) \cos(\pi y) \sin(\pi t). \tag{19}$$

These functions are chosen because they are smooth, nontrivial, and spatially periodic on symmetric spatial domains.

4.1.2 Derivatives of the Manufactured Solution

Beginning with $\phi(x, y, t)$ the first and second derivatives of the manufactured solution with respect to time are:

$$\phi_t(x, y, t) = \frac{\partial}{\partial t}\Big[1 + \cos(\pi x) \cos(\pi y) \cos(\pi t)\Big] = -\pi \cos(\pi x) \cos(\pi y) \sin(\pi t),$$

$$\phi_{tt}(x, y, t) = \frac{\partial}{\partial t}\Big[-\pi \cos(\pi x) \cos(\pi y) \sin(\pi t)\Big] = -\pi^2 \cos(\pi x) \cos(\pi y) \cos(\pi t).$$

The spatial derivatives are:

$$\phi_x(x, y, t) = -\pi \sin(\pi x) \cos(\pi y) \cos(\pi t),$$

$$\phi_{xx}(x, y, t) = -\pi^2 \cos(\pi x) \cos(\pi y) \cos(\pi t),$$

$$\phi_y(x, y, t) = -\pi \cos(\pi x) \sin(\pi y) \cos(\pi t),$$

$$\phi_{yy}(x, y, t) = -\pi^2 \cos(\pi x) \cos(\pi y) \cos(\pi t).$$

Thus,

$$\phi_{tt} - \phi_{xx} - \phi_{yy} = -\pi^2(\cdots) - \big[-\pi^2(\cdots)\big] - \big[-\pi^2(\cdots)\big] = +\pi^2 \cos(\pi x) \cos(\pi y) \cos(\pi t).$$

Therefore, the left-hand side of the ϕ-equation (16) becomes

$$\pi^2 \cos(\pi x) \cos(\pi y) \cos(\pi t) + \phi + V_\phi.$$

Computing the time derivatives for $\chi(x, y, t)$:

$$\chi_t(x, y, t) = \pi \cos(\pi x) \cos(\pi y) \cos(\pi t),$$

$$\chi_{tt}(x, y, t) = -\pi^2 \cos(\pi x) \cos(\pi y) \sin(\pi t),$$

and the spatial derivatives:

$$\chi_x(x, y, t) = -\pi \sin(\pi x) \cos(\pi y) \sin(\pi t),$$

$$\chi_{xx}(x, y, t) = -\pi^2 \cos(\pi x) \cos(\pi y) \sin(\pi t),$$

$$\chi_y(x, y, t) = -\pi \cos(\pi x) \sin(\pi y) \sin(\pi t),$$

$$\chi_{yy}(x, y, t) = -\pi^2 \cos(\pi x) \cos(\pi y) \sin(\pi t).$$

Thus,

$$\chi_{tt} - \chi_{xx} - \chi_{yy} = -\pi^2(\cdots) - \left[-\pi^2(\cdots)\right] - \left[-\pi^2(\cdots)\right] = +\pi^2 \cos(\pi x)\cos(\pi y)\sin(\pi t).$$

The left-hand side of the χ-equation (17) then becomes

$$\pi^2 \cos(\pi x)\cos(\pi y)\sin(\pi t) + \chi - V_\chi.$$

4.1.3 Forcing Functions

Thus, we define the forcing functions as:

$$F_\phi(x,y,t) = \pi^2 \cos(\pi x)\cos(\pi y)\cos(\pi t) + \phi(x,y,t) + V_\phi(\phi(x,y,t),\chi(x,y,t)), \tag{20}$$

$$F_\chi(x,y,t) = \pi^2 \cos(\pi x)\cos(\pi y)\sin(\pi t) + \chi(x,y,t) - V_\chi(\phi(x,y,t),\chi(x,y,t)). \tag{21}$$

With these definitions, the modified (forced) problem becomes:

$$\phi_{tt} - \phi_{xx} - \phi_{yy} + \phi + V_\phi(\phi,\chi) = F_\phi(x,y,t), \tag{22}$$

$$\chi_{tt} - \chi_{xx} - \chi_{yy} + \chi - V_\chi(\phi,\chi) = F_\chi(x,y,t). \tag{23}$$

By construction, substituting the manufactured solutions (18) and (19) into the left-hand sides of (22) and (23) yields exactly $F_\phi(x,y,t)$ and $F_\chi(x,y,t)$, respectively.

This recast's our weak EOM:

$$\begin{aligned}
K_1 &= \int \left(\frac{\partial u}{\partial t}\Psi + \frac{\partial \phi}{\partial x}\frac{\partial \Psi}{\partial x} + \frac{\partial \phi}{\partial y}\frac{\partial \Psi}{\partial y} + m_\phi^2 \phi\Psi + \frac{\partial}{\partial \phi}V(\phi,\chi)\Psi - F_\phi\Psi \right)\\
K_2 &= \int \left(\frac{\partial \phi}{\partial t}\Psi - u\Psi \right)\\
G_1 &= \int \left(\frac{\partial v}{\partial t}\Psi + \frac{\partial \chi}{\partial x}\frac{\partial \Psi}{\partial x} + \frac{\partial \chi}{\partial y}\frac{\partial \Psi}{\partial y} + m_\chi^2 \chi\Psi - \frac{\partial}{\partial \chi}V(\phi,\chi)\Psi - F_\chi\Psi \right)\\
G_2 &= \int \left(\frac{\partial \chi}{\partial t}\Psi - v\Psi \right)
\end{aligned}$$

By incorporating the forcing functions into the numerical solver, we can compute numerical solutions and compare them to the exact manufactured solutions. A convergence study is shown in table 2 which indicates approximately 2nd order convergence.

4.1.4 Results

The rates in table 2 are calculated by doing

$$Rate_{\phi/\chi} = \frac{log\,(E_{coarse}/E_{fine})}{log(h_{coarse}/h_{fine})}.$$

The mesh is setup up so that $h = h_x = h_y = h_t$. Thus, $log(h_{coarse}/h_{fine}) = log(2)$.

We see from table 2 that the numerical results compare nicely with the manufactured results. These findings give us good reason to believe that our 2+1 numerical implementation is working properly.

Table 2: Convergence Study Results (2+1 MMS)

Mesh ($nx \times ny \times nt$)	ϕ L_2 Error	Rate$_\phi$	χ L_2 Error	Rate$_\chi$
$10 \times 10 \times 11$	1.0936e-01	-	8.0519e-02	-
$20 \times 20 \times 21$	2.2424e-02	2.286	1.7299e-02	2.218
$40 \times 40 \times 41$	5.3358e-03	2.071	4.2056e-03	2.041