Numerical Solution of Lotka-voltera predator-prey model using Finite Element Method

Project by Subarna Biswas

06/12/2023

Abstract

We consider spatial-temporal models of population distribution in heterogeneous domains. Mathematical models of such problems are described by a system of nonlinear diffusion-reaction equations. We consider the problem in a two-dimensional domain with circle inclusions. To solve the problem numerically, we construct an unstructured grid that resolves inclusions on the grid level and constructs a semi-discrete system using a finite element method. For time approximation, we apply a semi-implicit scheme where the reaction term of the equation is taken from the previous time layer. We present numerical results for some test problems to investigate the influence of the parameters on time to reach equilibrium and the final equilibrium state. [Wan+23]

1 INTRODUCTION

The finite element (FE) method is a numerical technique for computing approximate solutions to complex mathematical problems described by differential equations. It solve differential equations by discretizing the domain into a finite mesh. Numerically speaking, a set of differential equations are converted into a set of algebraic equations to be solved for unknown at the nodes of the mesh. The first development of FE method can be traced back to the work by Hrennikoff in 1941 and Courant in 1943 .
Aplications: This method can be advantageous for treatment of nonlinearities and complex geometry. By the use of the FE method, any complex domain can be discretized by triangular elements in 2D and by tetrahedra elements in 3D. The FE method is suited for structural analysis, heat transfer, electrical and magnetical analysis, fluid and acoustic analysis, multi-physics, etc.
In this project we will apply the finite element method on second order non-linear differential equation to analysis the predator-prey population dynamics.The Lotka–Volterra model is frequently used to describe the dynamics of ecological systems in which two species interact, one a predator and one its prey. The equations have been known as the basis of many types of models that involve the interactions of different populations, some concepts such as diffusion and functional response have been added to the Lotka–Volterra equations to gain a better understanding of the dynamics of population interactions.We consider population changes due to predation as well as diffusion processes.
The project is organized as follows. The first section describes the mathematical model for two-species population in the heterogeneous domain. Next, we present the construction of the discrete system based on the finite element method and semi-implicit time approximation in the third section. Numerical results and investigation of the influence of the parameters on the solution and final equilibrium state are presented in the fourth section. Finally, we present the conclusion. [Wan+23]

2 Problem formulation.

We consider the Lotka-Volterra Predator-Prey model as our coupled non-linear system in one dimensional heterogenous domain. The mathematical model is described by the following system of nonlinear diffusion-reaction equations for u(x,t) and v(x,t).

$$\begin{aligned} u_t - d_1 u_{xx} &= \alpha u - \beta uv \\ v_t - d_2 v_{xx} &= \delta uv - \gamma v \end{aligned} \tag{1}$$

The variable u is the population density of prey (for example, the number of rabbits per square kilometre). The variable v is the population density of some predator (for example, the number of foxes per square kilometre).

Where, α = prey growth rate; β = rate at which prey decrease; δ = predator growth rate; γ = rate at which predator decrease.
We consider following initial conditions :

$$u = 0,\ t = 0,\ x\epsilon\,\Omega[0,1],$$

Similar for second equation,

$$v = 0,\ t = 0,\ x\epsilon\,\Omega[0,1],$$

With boundary conditions:

$$u(0,t) = g_1, t > 0, x = 0,\ \text{and}\ u_x(1,t) = 0, t > 0, x = 1.$$

Similarly,

$$v(0,t) = g_2, t > 0, x = 0,\ \text{and}\ v_x(1,t) = 0, t > 0, x = 1.$$

3 Approximation by space and time

In order to apply the FE method to solve this problem, we carry out the following process. We seek a weak solution for u and v in -

$$Q = \{q\epsilon H^1(\Omega) : q = 0 \text{ on }\ x = 0\ and\ x = 1\}$$

Multiplying the Equ. (1) with $q\epsilon\ Q$ integrating over Ω and using Green's formula, we obtain, for t on $[0,T]$.

$$\int_\Omega u_t q dx - d_1 \int_\Omega (u)_x q dx = \int_\Omega f1(u,v) q dx$$

$$\int_\Omega v_t q dx - d_2 \int_\Omega (v)_x q dx = \int_\Omega f2(u,v) q dx$$

Where,

$$f_1(u,v) = \alpha u - \beta uv$$

$$f_2(u,v) = \delta uv - \gamma v$$

Let, $u_h\epsilon Q_h \subset Q$ be the space of all continuous piecewise linear functions on a triangle mesh of Ω.

Taking, $u_h = \sum_{j=1}^{N} u_j\ \Phi_j(x)$, $v_h = \sum_{j=1}^{N} v_j\ \Phi_j(x)$ and choosing $q = \Phi_i(x)$, $u_t = \dfrac{u_j^{n+1} - u_j^n}{\tau}$

where, $x\epsilon\Omega$ and $n = 1, 2, ..N$.

Then for explicit scheme our set of equation becomes,

$$\sum_{j=1}^{N} \int_\Omega \frac{u_j^{n+1} - u_j^n}{\tau} \Phi_j(x)\Phi_i(x) dx + \sum_{j=1}^{N} \int_\Omega d_1 u_j^n \frac{\partial \Phi_j(x)}{\partial x} \frac{\partial \Phi_i(x)}{\partial x} dx = \sum_{j=1}^{N} \int_\Omega f_1(u^n, v^n)\Phi_i(x) dx$$

$$\sum_{j=1}^{N} \int_\Omega \frac{v_j^{n+1} - v_j^n}{\tau} \Phi_j(x)\Phi_i(x) dx + \sum_{j=1}^{N} \int_\Omega d_2 v_j^n \frac{\partial \Phi_j(x)}{\partial x} \frac{\partial \Phi_i(x)}{\partial x} dx = \sum_{j=1}^{N} \int_\Omega f_2(u^n, v^n)\Phi_i(x) dx$$

And for Implicit Scheme,

$$\sum_{j=1}^{N} \int_\Omega \frac{u_j^{n+1} - u_j^n}{\tau} \Phi_j(x)\Phi_i(x) dx + \sum_{j=1}^{N} \int_\Omega d_1 u_j^{n+1} \frac{\partial \Phi_j(x)}{\partial x} \frac{\partial \Phi_i(x)}{\partial x} dx = \sum_{j=1}^{N} \int_\Omega f_1(u^n, v^n)\Phi_i(x) dx$$

$$\sum_{j=1}^{N} \int_\Omega \frac{v_j^{n+1} - v_j^n}{\tau} \Phi_j(x)\Phi_i(x) dx + \sum_{j=1}^{N} \int_\Omega d_2 v_j^{n+1} \frac{\partial \Phi_j(x)}{\partial x} \frac{\partial \Phi_i(x)}{\partial x} dx = \sum_{j=1}^{N} \int_\Omega f_2(u^n, v^n)\Phi_i(x) dx$$

Now for each element,

$$e_k = [x_{k-1}, x_k] \text{ we have } \Omega = \bigcup_k e_k$$

$$\text{Let, } \quad M^k = \sum_k m_{ij}^k = \int_{e_k} \Phi_j(x)\Phi_i(x)dx$$

$$A^k = \sum_k a_{ij}^k = \int_{e_k} \frac{\partial \Phi_j(x)}{\partial x}\frac{\partial \Phi_i(x)}{\partial x}dx$$

$$b^{k,n} = \sum_k b_i^{k,n} = \int_{e_k} f(u^n, v^n)\Phi_i(x)dx$$

Therefore, final approximation we obtain for our set of equations-
For Explicit scheme,

$$\begin{aligned} \frac{1}{\tau}Mu^{n+1} - \frac{1}{\tau}Mu^n + d_1 Au^n &= b_1^n \\ \frac{1}{\tau}Mv^{n+1} - \frac{1}{\tau}Mv^n + d_2 Av^n &= b_2^n \end{aligned} \tag{2}$$

For implicit Scheme,

$$\begin{aligned} \frac{1}{\tau}Mu^{n+1} - \frac{1}{\tau}Mu^n + d_1 Au^{n+1} &= b_1^n \\ \frac{1}{\tau}Mv^{n+1} - \frac{1}{\tau}Mv^n + d_2 Av^{n+1} &= b_2^n \end{aligned} \tag{3}$$

4 Numerical results

We can visualize the comparative movement of predator and prey due to predation as well diffusion. I have shown three cases based on h-sensitivity and τ - sensititivity. At first we considered simple ODE model then complicated Diffusion reaction model. For time approximation, we apply a implicit scheme where the reaction term of the equation is taken from the previous time layer. The population equilibrium of this model has the property that the prey equilibrium density depends on the predator's parameters, and the predator equilibrium density on the prey's parameters.We observe that the dynamics of predator and prey populations have a tendency to oscillate. When one species starts to diffuse ,the other one continues to accumulate.

Case-1:

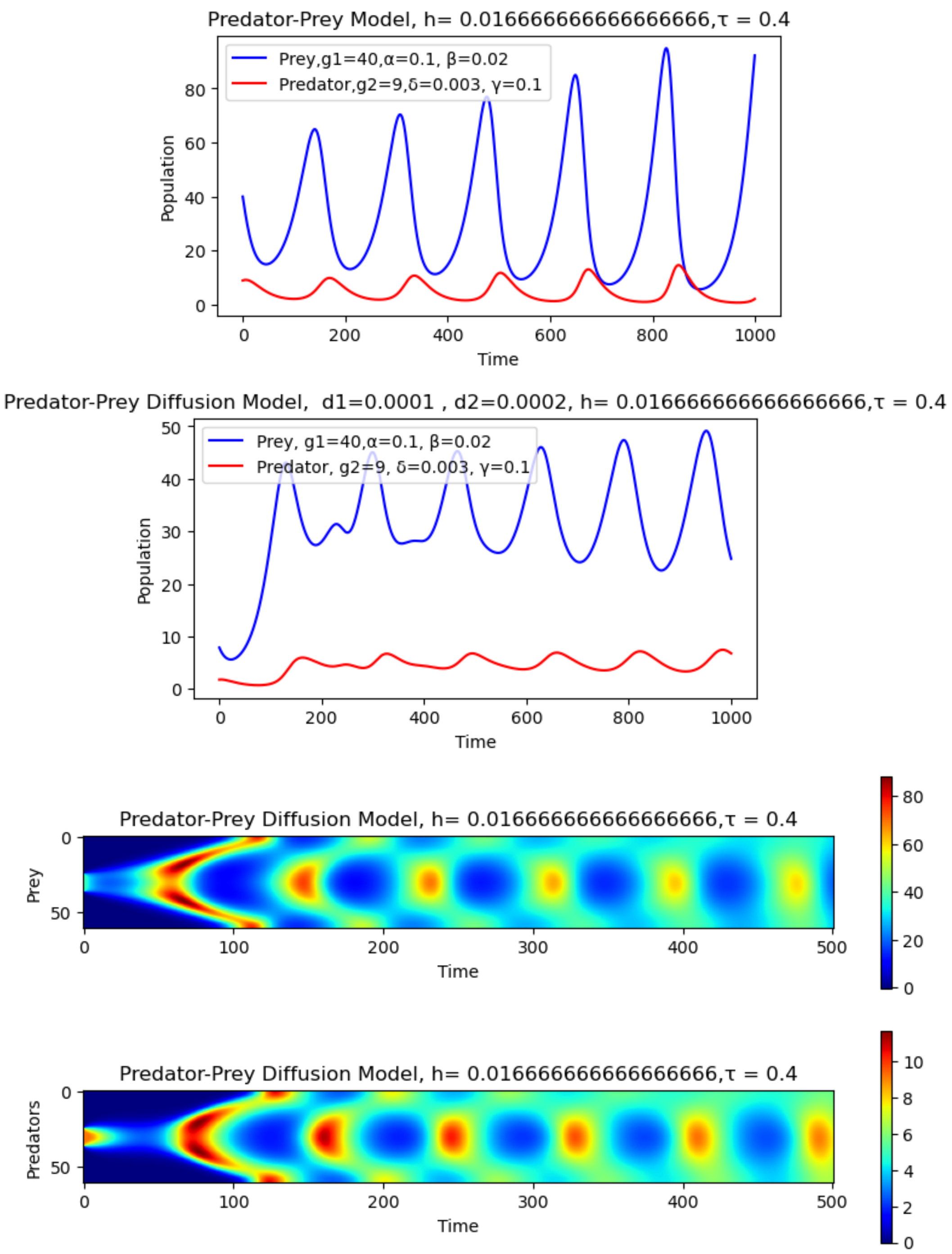

Figure 1: Two species model, effect of diffusion.

Case-2:

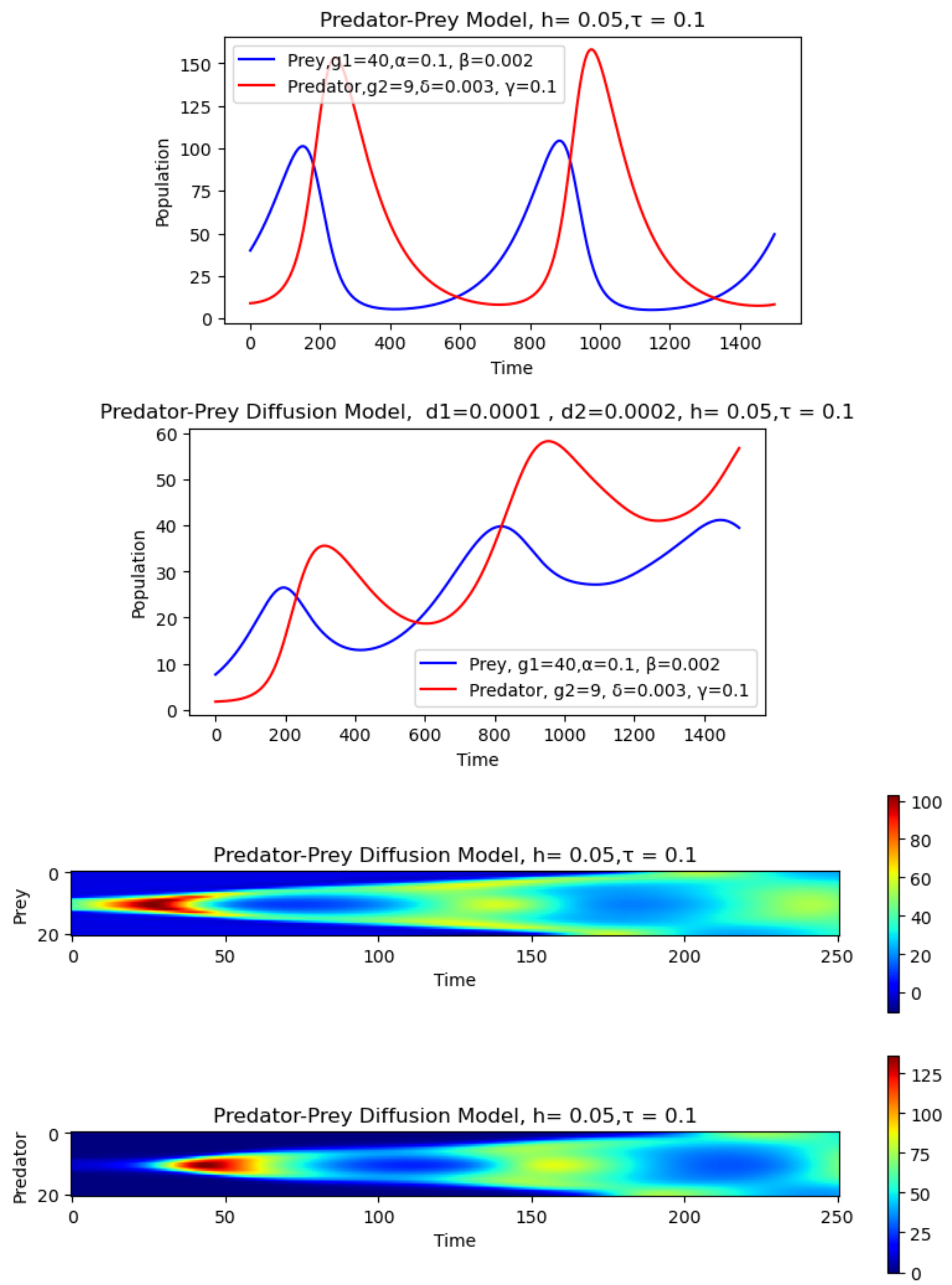

Figure 2: Two species model, effect of diffusion.

Case-3:

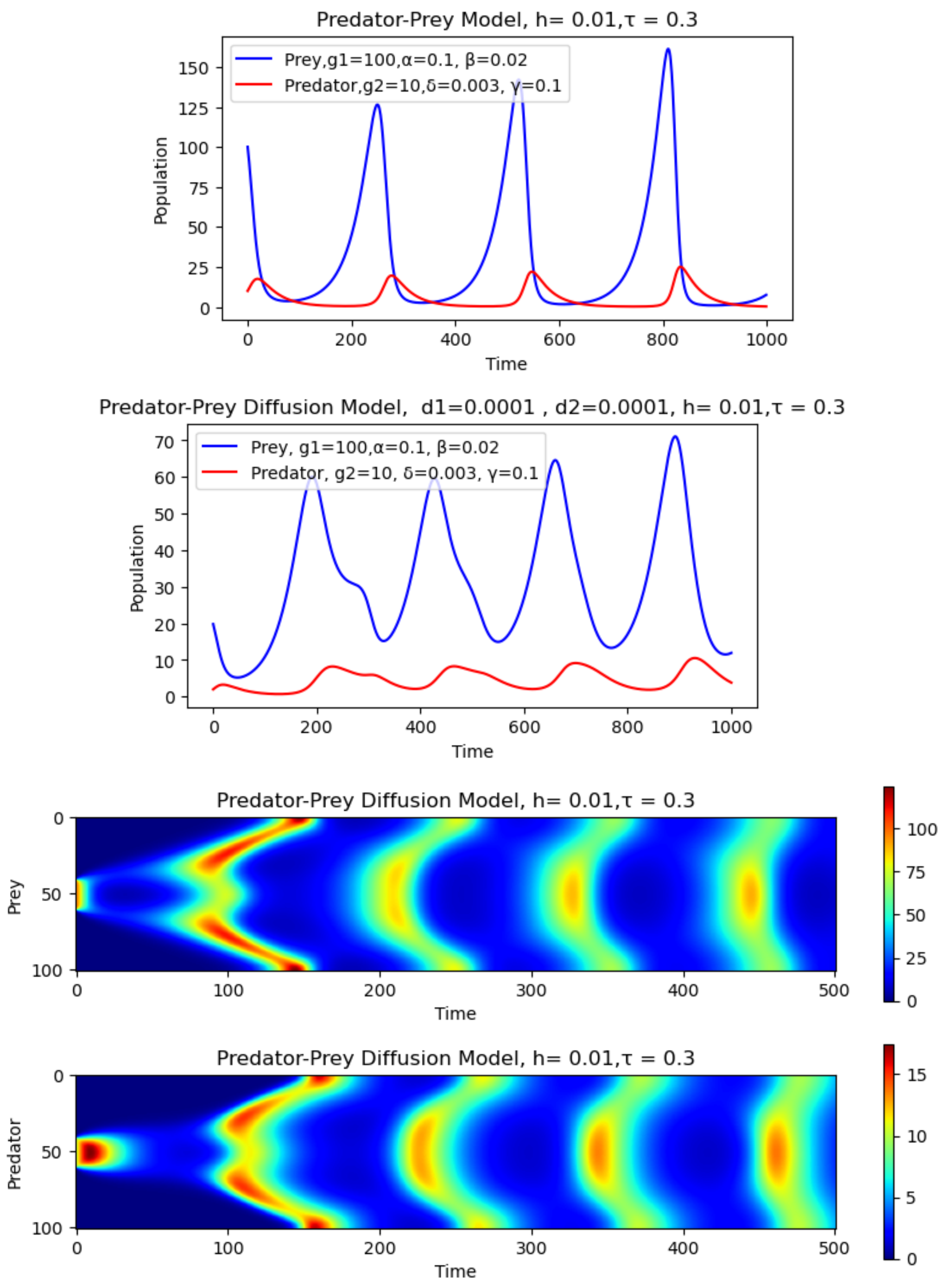

Figure 3: Two species model, effect of diffusion.

5 Appendix 1 (system with 4 unknowns)

We can get numeric results for our mass and stiffness matrix using the Simpson's Rule:

$$\int_{x_{k-1}}^{x_k} g(x) = \frac{h}{6}[g(x_{k-1}) + 4g(x_{k-\frac{1}{2}}) + g(x_k)]$$

We have,

$$M^k = \begin{bmatrix} \int_{e_k} \Phi_{k-1}(x)\Phi_{k-1}(x)dx & \int_{e_k} \Phi_{k-1}(x)\Phi_k(x)dx \\ \int_{e_k} \Phi_k(x)\Phi_{k-1}(x)dx & \int_{e_k} \Phi_k(x)\Phi_k(x)dx \end{bmatrix} = \frac{h}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$A^k = \begin{bmatrix} \int_{e_k} \frac{\partial\Phi_{k-1}(x)}{\partial x}\frac{\Phi_{k-1}(x)}{\partial x}dx & \int_{e_k} \frac{\partial\Phi_{k-1}(x)}{\partial x}\frac{\partial\Phi_k(x)}{\partial x}dx \\ \int_{e_k} \frac{\partial\Phi_k(x)}{\partial x}\frac{\partial\Phi_{k-1}(x)}{\partial x}dx & \int_{e_k} \frac{\partial\Phi_k(x)}{\partial x}\frac{\partial\Phi_k(x)}{\partial x}dx \end{bmatrix} = \frac{1}{h}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$b^{k,n} = \begin{bmatrix} \int_{e_k} f\Phi_{k-1}(x)dx \\ \int_{e_k} f\Phi_k(x)dx \end{bmatrix}$$

Therefore, using the global matrix A, M and b we can formulate system of equations for any number of unknowns. To avoid unnecessary repetition I choose only one equation from given system of equation.

In general case of explicit scheme with four unknown-

$$\frac{h}{6\tau}\begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}\begin{bmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \end{bmatrix} - \frac{h}{6\tau}\begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}\begin{bmatrix} u_0^{n} \\ u_1^{n} \\ u_2^{n} \\ u_3^{n} \end{bmatrix} + \frac{d}{h}\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}\begin{bmatrix} u_0^{n} \\ u_1^{n} \\ u_2^{n} \\ u_3^{n} \end{bmatrix} = \begin{bmatrix} \int_{e_k} f\Phi_{k-1}(x)dx \\ \int_{e_k} f(\Phi_{k-1}(x) + \Phi_k(x))dx \\ \int_{e_k} f(\Phi_k(x) + \Phi_{k-1}(x))dx \\ \int_{e_k} f\Phi_k(x)dx \end{bmatrix}$$

And general form of the system of equation with four unknown in case of implicit scheme-

$$\frac{h}{6\tau}\begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}\begin{bmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \end{bmatrix} - \frac{h}{6\tau}\begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}\begin{bmatrix} u_0^{n} \\ u_1^{n} \\ u_2^{n} \\ u_3^{n} \end{bmatrix} + \frac{d}{h}\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}\begin{bmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \end{bmatrix} = \begin{bmatrix} \int_{e_k} f\Phi_{k-1}(x)dx \\ \int_{e_k} f(\Phi_{k-1}(x) + \Phi_k(x))dx \\ \int_{e_k} f(\Phi_k(x) + \Phi_{k-1}(x))dx \\ \int_{e_k} f\Phi_k(x)dx \end{bmatrix}$$

6 Appendix 2 (code)

Code-1:

```
#Implicit ODE

import numpy as np
import matplotlib.pyplot as plt


Ne=60 #20,100
h =1.0/Ne
N=Ne+1
print(N)

D = np.zeros((N, N))
M = np.zeros((N, N))
f = np.zeros(N)
for ei in range(Ne):
    #diffusion
    D[ei][ei] += 1/h
    D[ei][ei+1] += -1/h
    D[ei+1][ei] += -1/h
    D[ei+1][ei+1] += 1/h

    #mass matrix
    M[ei][ei] += 2*h/6
    M[ei][ei+1] += h/6
    M[ei+1][ei] += h/6
    M[ei+1][ei+1] += 2*h/6
    #rhs
    f[ei]+=h/2
    f[ei+1]+=h/2

alpha = 0.1 #prey increase rate
beta =  0.02 #prey decrease rate
delta = 0.003 # #predator increase
gamma = 0.1  #predator decrease rate

g1, g2 = 40, 9
Nt = 1000
tau = 0.4

u1 = np.zeros(Nt)
u2 = np.zeros(Nt)

u1[0]= g1
u2[0]= g2

for t in range(1, Nt):
    u1[t] = u1[t-1] + tau *(alpha*u1[t-1] - beta*u1[t-1]*u2[t-1])
    u2[t] = u2[t-1] + tau *(delta*u1[t-1]*u2[t-1] - gamma*u2[t-1])

plt.figure(figsize=(6,3))
plt.plot(u1,label= 'Prey,g1='+str(g1)+',\u03B1='+str(alpha)+',
\u03B2='+str(beta)+'', c='b')
plt.plot(u2,label= 'Predator,g2='+str(g2)+',\u03B4='+str(delta)+',
\u03B3='+str(gamma)+'', c='r')
plt.xlabel('Time')

plt.ylabel('Population')
plt.title('Predator-Prey Model, h= '+str(h)+',\u03C4 = '+str(tau)+'')
plt.legend()
plt.show()
```

Code-2: Lotka–Volterra diffusion-reaction model.

```
def plot_sol(uu1,uu2,Nt):
    plt.figure(figsize=(6,3))
    plt.plot(av1,label= 'Prey, g1='+str(g1)+',\u03B1='+str(alpha)+',
    \u03B2='+str(beta)+'', c='b')
    plt.plot(av2,label= 'Predator, g2='+str(g2)+', \u03B4='+str(delta)+',
    \u03B3='+str(gamma)+'', c='r')
    plt.title('Predator-Prey Diffusion Model, d1='+str(d1)+' , d2='+str(d2)+',
    h= '+str(h)+',\u03C4 = '+str(tau)+'')
    plt.xlabel('Time')
    plt.ylabel('Population')

    plt.legend()
    plt.show()
    plt.figure(figsize=(10,5))
    plt.subplot(2,1,1)
    plt.imshow(uu1[::2, :].T, cmap= 'jet')
    plt.colorbar()
    plt.title('Predator-Prey Diffusion Model, h= '+str(h)+',\u03C4 = '+str(tau)+'')
    plt.xlabel('Time')
    plt.ylabel('Prey')
    plt.subplot(2,1,2)
    plt.imshow(uu2[::2, :].T, cmap= 'jet')
    plt.colorbar()
    plt.title('Predator-Prey Diffusion Model, h= '+str(h)+',\u03C4 = '+str(tau)+'')
    plt.xlabel('Time')
    plt.ylabel('Predators')
    plt.show()

#parameter

alpha = 0.1   #prey increase rate
beta =  0.02  #prey decrease rate
delta = 0.003 #predator increase
gamma = 0.1   #predator decrease rate

g1, g2 = 100, 10
Nt = 1000
tau = 0.3
d1, d2 = 1.0e-4, 1.0e-4

#some other parameters:
#g1, g2 = 40, 9
#Nt = 1500
#tau = 0.1
#tau = 0.4
#alpha = 0.1   #prey increase rate
#beta =  0.002 #prey decrease rate
#delta = 0.003 #predator increase
#gamma = 0.1   #predator decrease rate
#d1, d2 = 1.0e-4, 2*1.0e-4


u1 = np.zeros(N)
u2 = np.zeros(N)

for i in range(N):
    if i < 3*N/5 and i > 2*N/5:
        u1[i]= g1
        u2[i]= g2

A1 = M/tau + d1*D
A2 = M/tau + d2*D

av1, av2 = [] ,[]
uu1 = np.zeros((Nt+1, N))
uu2 = np.zeros((Nt+1, N))
uu1[0,:] =u1[:]
uu2[0,:] =u2[:]

av1.append(u1.sum()/N)
av2.append(u2.sum()/N)
```

```
for ti in range(1, Nt+1):
    b1 = (M/tau)@u1
    b2 = (M/tau)@u2
    for ei in range(Ne):
        #1
        b1[ei] += alpha*h/2*u1[ei] - beta*h/2*u1[ei]*u2[ei]
        b1[ei+1] += alpha*h/2*u1[ei+1] - beta*h/2*u1[ei+1]*u2[ei+1]
        #2
        b2[ei] += delta*h/2*u1[ei]*u2[ei] - gamma*h/2*u2[ei]
        b2[ei+1] += delta*h/2*u1[ei+1]*u2[ei+1] - gamma*h/2*u2[ei+1]
    u1 = np.linalg.solve(A1,b1)
    u2 = np.linalg.solve(A2,b2)

    uu1[ti,:] = u1[:]
    uu2[ti,:] = u2[:]

    av1.append(u1.sum()/N)
    av2.append(u2.sum()/N)

plot_sol(uu1, uu2, Nt)
```

Code-3:

```
#parameter
alpha = 0.1   #prey increase rate
beta =  0.002 #prey decrease rate
delta = 0.003 #predator increase
gamma = 0.1   #predator decrease rate

d1, d2 = 1.0e-4, 2*1.0e-4

u1 = np.zeros(N)
u2 = np.zeros(N)

for i in range(N):
    if i<N*3/5 and i> N*2/5:
        u1[i]= g1
        u2[i]= g2

av1, av2 = [] ,[]
uu1 = np.zeros((Nt+1, N))
uu2 = np.zeros((Nt+1, N))
uu1[0,:] =u1[:]
uu2[0,:] =u2[:]

av1.append(u1.sum()/N)
av2.append(u2.sum()/N)
for ti in range(1, Nt+1):
    A1 = M/tau + d1*D
    A2 = M/tau + d2*D
    for ei in range(Ne):
        u1_1, u1_12, u1_2 = u1[ei] , (u1[ei]+ u1[ei+1])/2, u1[ei+1]
        u2_1, u2_12, u2_2 = u2[ei] , (u2[ei]+ u2[ei+1])/2, u2[ei+1]

        A1[ei][ei] -= h/6 * (alpha -beta*u2_1 + alpha - beta*u2_12)
        A1[ei][ei+1] -= h/6 * (alpha -beta *u2_12)
        A1[ei+1][ei] -= h/6 * (alpha -beta*u2_12)
        A1[ei+1][ei+1] -= h/6 *(alpha -beta*u2_2 +alpha - beta*u2_12)

        A2[ei][ei] -= h/6 * (delta*u1_1 - gamma + delta*u1_2 -gamma)
        A2[ei][ei+1] -= h/6 * (delta*u1_12 - gamma)
        A2[ei+1][ei] -= h/6 * (delta*u1_1 - gamma)
        A2[ei+1][ei+1] -= h/6 *(delta*u1_2 - gamma + delta *u1_12 -gamma)

    b1 = (M/tau)@u1
    b2 = (M/tau)@u2
    u1 = np.linalg.solve(A1,b1)
    u2 = np.linalg.solve(A2,b2)

    uu1[ti,:] = u1[:]
    uu2[ti,:] = u2[:]

    av1.append(u1.sum()/N)
    av2.append(u2.sum()/N)

plot_sol(uu1, uu2, Nt)
```

[Wan+23] [Bac21] [JWX14]

References

[Bac21] Mahboub Baccouch. “A Brief Summary of the Finite Element Method for Differential Equations”. In: *Finite Element Methods and Their Applications*. Ed. by Mahboub Baccouch. Rijeka: IntechOpen, 2021. Chap. 1. DOI: 10.5772/intechopen.95423. URL: https://doi.org/10.5772/intechopen.95423.

[JWX14] Yunfeng Jia, Jianhua Wu, and Hong-Kun Xu. “Positive solutions of a Lotka–Volterra competition model with cross-diffusion”. In: *Computers Mathematics with Applications* 68.10 (2014), pp. 1220–1228. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2014.08.016. URL: https://www.sciencedirect.com/science/article/pii/S0898122114004088.

[Wan+23] Youwen Wang et al. “Numerical Investigation and Factor Analysis of Two-Species Spatial-Temporal Competition System after Catastrophic Events”. In: *WSEAS TRANSACTIONS ON SYSTEMS* 22 (May 2023), pp. 423–436. DOI: 10.37394/23202.2023.22.45.