# Numerical Methods for the One-Dimensional Nonlinear Reaction Diffusion Equation

Samuele Rebecchi 10781703

05/09/2024

## Introduction

I have chosen the project 9.7 of the Lecture Notes: "Numerical methods for the one-dimensional nonlinear reaction diffusion equation". To find an equation where to apply these methods, I have referred to the article "Nonlinear Dynamics of a Marine Phytoplankton-Zooplankton System" by Pengfei Wang et al,2016 [Wan+16].

The interaction between phytoplankton and zooplankton is a fundamental component of marine ecosystems influencing primary production and nutrient cycling. Mathematical modeling of these interactions helps us understand and predict ecological dynamics. In this project, we aim to numerically solve the reaction-diffusion equations describing the phytoplankton-zooplankton system using different numerical methods to evaluate their accuracy and efficiency.

I focus on discretizing the equations using finite difference approximations for the second derivative and Runge-Kutta methods for time integration. Specifically, I implement a second-order Runge-Kutta method combined with a second-order finite difference scheme and a fourth-order Runge-Kutta method combined with a fourth-order finite difference scheme.

The model is based on the reaction-diffusion equations that describe the temporal and spatial evolution of phytoplankton (P) and zooplankton (Z) densities:

$$\frac{\partial P}{\partial t} = r_1 P(1 - \frac{P}{K_1}) - \frac{\beta_1 P Z}{m + P} + D_1 \Delta P$$
$$\frac{\partial Z}{\partial t} = r_2 Z(1 - \frac{Z}{K_2}) + \frac{\beta_2 P Z}{m + P} + D_2 \Delta Z$$

where:

- $r_1, r_2$ are the intrinsic growth rates
- $K_1, K_2$ are the carrying capacities
- $\beta_1, \beta_2$ are interaction coefficients

- $m$ is the half-saturation constant
- $D_1, D_2$ are the diffusion coefficients for phytoplankton and zooplankton respectively.

## Part 1: Parameters and Conditions

Based on the reference article I define all necessary parameters including growth rates, carrying capacities, interaction coefficients, and diffusion coefficients.

Matlab code:

```
L = 40; % Length of the domain
T = 200; % Total simulation time
r1 = 0.8; % Intrinsic growth rate for phytoplankton
r2 = 0.7; % Intrinsic growth rate for zooplankton
K1 = 1.0; % Carrying capacity for phytoplankton
K2 = 0.47; % Carrying capacity for zooplankton
beta1 = 0.8; % Predation rate of zooplankton on phytoplankton
beta2 = 0.7; % Conversion efficiency of ingested biomass to zooplankton
m = 0.5; % Half-saturation constant
D1 = 0.02; % Diffusion coefficient for phytoplankton
D2 = 0.1; % Diffusion coefficient for zooplankton

Nx = 100; % Number of spatial points
Nt = 1000; % Number of time points
dx = L / Nx; % Spatial step size
dt = T / Nt; % Time step size
x = (0:dx:L-dx)'; % Spatial grid
```

## Part 2: Boundary Conditions

I assume Dirichlet boundary conditions and set initial conditions based on page 166 of the Lecture notes.

```
P0 = @(x) 1/( exp(D1*sqrt(2)*(x-L/3)/2)); % Initial condition for
                                            phytoplankton
Z0 = @(x) 1/( exp(D2*sqrt(2)*(x-L/3)/2)); % Initial distribution of
                                            Zooplankton
gP0 = @(t) 0; % Left boundary condition for phytoplankton
gPL = @(t) 0; % Right boundary condition for phytoplankton
gZ0 = @(t) 0; % Left boundary condition for zooplankton
gZL = @(t) 0; % Right boundary condition for zooplankton
```

## Part 3: Initialize Solution Matrices

I initialize matrices for P and Z for both 2nd-order and 4th-oder Runge-Kutta methods. We defined P_RK2, P_RK4, Z_RK2 and Z_RK4 dimensions as the final answers. P_RK2 and P_RK4 are phytoplankton densities using second-order Runge Kutta and fourth-order Runge Kutta, and Z_RK2 and Z_RK4 are zooplankton densities using second-order Runge Kutta and fourth-order Runge Kutta.

```
P_RK2 = zeros(Nx, Nt); % Phytoplankton density using RK2
Z_RK2 = zeros(Nx, Nt); % Zooplankton density using RK2
P_RK2(:, 1) = P0(0);
Z_RK2(:, 1) = Z0(0);

P_RK4 = zeros(Nx, Nt); % Phytoplankton density using RK4
Z_RK4 = zeros(Nx, Nt); % Zooplankton density using RK4
P_RK4(:, 1) = P0(0);
Z_RK4(:, 1) = Z0(0);
```

## Part 4: Defining Functions

The first part of the project, I calculate the diffusion-reaction function with the central second finite difference approximation of the second derivative. The finite difference approximation is a numerical approach for calculating derivatives, go to page 33 of the lecture, its formula is:

$$\delta^2 f_i = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} \quad i = 1, \ldots, n-1.$$

I applied this formula to the underlined part of our main function (the diffusion term), as it is the second-order derivative of our function.

$$\frac{\partial c}{\partial t} = \epsilon \frac{\partial^2 c}{\partial x^2} + f(c)$$

It is showed in the code below how I have coded the notions above. Also, I defined fP and fZ which are Reaction terms for phytoplankton and zooplankton, including logistic growth and loss due to predation by zooplankton and phytoplankton (based on the lecture page 166 and the reference paper).

I calculated the diffusion term in three steps because it asked for the second-order finite difference approximation, also I have the zero-state boundary condition (gP0) and the final boundary condition(gPL).

The second part of the project calls for determining the reaction-diffusion function with the fourth-order approximation of the second derivative based on page 125 of the lecture notes:

$$\delta^{2,4} c_i = \frac{-c(x_{i+2}) + 16c(x_{i+1}) - 30c(x_i) + 16c(x_{i-1}) - c(x_{i-2})}{12\Delta x^2}.$$

According to the equation above, I defined the diffusion function. Here I calculated the fourth-order approximation of the second derivative in five steps (Diffusion terms using fourth-order central difference and correcting for boundaries using second-order central difference). Having in mind that I cannot use the fourth-order equation on the boundary points (refer to the point on page 125 of the lecture), I corrected the function in the boundaries (m is the half-saturation constant).

dPdt = Sum of reaction and diffusion terms for phytoplankton

dZdt = Sum of reaction and diffusion terms for zooplankton

```
%% Define the reaction-diffusion system function
function [dPdt, dZdt] = reaction_diffusion(P, Z, r1,
   r2, K1, K2, beta1, beta2, m, D1, D2, dx, order, gP0
   , gPL, gZ0, gZL, t)
  % Reaction terms
  fP = r1 * P .* (1 - P / K1) - beta1 * P .* Z ./ (m +
    P);
  fZ = r2 * Z .* (1 - Z / K2) + beta2 * P .* Z ./ (m +
    P);

  % Diffusion terms
  if order == 2
    % Diffusion terms using second-order central
   difference
    d2Pdx2 = ([P(2:end); P(end)] - 2 * P + [P(1); P(1:
   end-1)]) / dx^2;
    d2Zdx2 = ([Z(2:end); Z(end)] - 2 * Z + [Z(1); Z(1:
   end-1)]) / dx^2;
  elseif order == 4
    % Diffusion terms using fourth-order central
   difference
    d2Pdx2 = (-[P(3:end); P(end); P(end)] + 16*[P(2:
   end); P(end)] - 30*P + 16*[P(1); P(1:end-1)] - [P
   (1); P(1); P(1:end-2)]) / (12*dx^2);
    d2Zdx2 = (-[Z(3:end); Z(end); Z(end)] + 16*[Z(2:
   end); Z(end)] - 30*Z + 16*[Z(1); Z(1:end-1)] - [Z
   (1); Z(1); Z(1:end-2)]) / (12*dx^2);

    % Correcting for boundaries using second-order
   central difference
    d2Pdx2(1) = (P(2) - 2 * P(1) + gP0(t)) / dx^2;
    d2Pdx2(2) = (P(3) - 2 * P(2) + P(1)) / dx^2;
    d2Pdx2(end-1) = (P(end) - 2 * P(end-1) + P(end-2))
    / dx^2;
    d2Pdx2(end) = (gPL(t) - 2 * P(end) + P(end-1)) /
   dx^2;
```

```
        d2Zdx2(1) = (Z(2) - 2 * Z(1) + gZ0(t)) / dx^2;
        d2Zdx2(2) = (Z(3) - 2 * Z(2) + Z(1)) / dx^2;
        d2Zdx2(end-1) = (Z(end) - 2 * Z(end-1) + Z(end-2))
        / dx^2;
        d2Zdx2(end) = (gZL(t) - 2 * Z(end) + Z(end-1)) /
      dx^2;
    end

    % Total derivatives
    dPdt = fP + D1 * d2Pdx2;
    dZdt = fZ + D2 * d2Zdx2;
end
```

## Part 5: Implementation of Numerical Methods

Now, I move on to the next part of the project which has asked to use the finite difference approximation for the second derivative (which we defined above) and 2nd-order Runge Kutta.

```
% Time-stepping using Runge-Kutta 2nd order method
for n = 1:Nt-1
    % Compute the RK2 coefficients
    [k1P, k1Z] = reaction_diffusion(P_RK2(:, n), Z_RK2
      (:, n), r1,
    r2, K1, K2, beta1, beta2, m, D1, D2, dx, 2, gP0, gPL
      , gZ0, gZL, n*dt);
    [k2P, k2Z] = reaction_diffusion(P_RK2(:, n) + dt *
      k1P, Z_RK2(:,
    n) + dt * k1Z, r1, r2, K1, K2, beta1, beta2, m, D1,
      D2, dx, 2, gP0, gPL, gZ0, gZL, n*dt + dt);

    % Update densities using Runge-Kutta 2nd order
      method
    P_RK2(:, n+1) = P_RK2(:, n) + dt / 2 * (k1P + k2P);
    Z_RK2(:, n+1) = Z_RK2(:, n) + dt / 2 * (k1Z + k2Z);

    % Applying boundary conditions
    P_RK2(1, n+1) = gP0(n*dt);
    P_RK2(end, n+1) = gPL(n*dt);
    Z_RK2(1, n+1) = gZ0(n*dt);
    Z_RK2(end, n+1) = gZL(n*dt);
end
```

The 2nd-order Runge Kutta formula is on page 65 of the lecture:

$$
\begin{aligned}
f_1 &= f(u_k, t_k) \\
f_2 &= f(u_k + chf_1, t_k + ch) \\
u_{k+1} &= u_k + h(b_1 f_1 + b_2 f_2)
\end{aligned}
$$

According to this formula, I have coded lines below. To discretize the time domain, I have defined a loop where $k1P$ and $k1Z$ are $f_1$, and $k2P$ and $k2Z$ are $f_2$ in the formula (4.7). Additionally, $P\_RK2$ and $Z\_RK2$ represent $u_{k+1}$. Furthermore, dt is equal to $h$, and the coefficients $b1$, $b2$, and $c$ are defined by the Heun method ($b1 = \frac{1}{2}$, $b2 = \frac{1}{2}$, $c = 1$) as described on page 66 of the lecture.

The following part of the project asked to use the forth-order finite difference approximation for the second derivative and the 4th-order Runge-Kutta (page 66 of the lecture notes) method to evaluate the diffusion-reaction function.

$$
\begin{aligned}
f_1 &= f(u_k, t_k) \\
f_2 &= f\left(u_k + \frac{h}{2} f_1, t_k + \frac{h}{2}\right) \\
f_3 &= f\left(u_k + \frac{h}{2} f_2, t_k + \frac{h}{2}\right) \\
f_4 &= f(u_k + hf_3, t_k + h) \\
u_{k+1} &= u_k + \frac{h}{6}\left(f_1 + 2f_2 + 2f_3 + f_4\right) \quad k = 0, \ldots, N-1.
\end{aligned}
$$

I have defined the 4th-order Runge-Kutta with the help of a reaction-diffusion function:

```
for n = 1:Nt-1
  % Compute the RK4 coefficients
  [k1P, k1Z] = reaction_diffusion(P_RK4(:, n), Z_RK4
   (:, n), r1,
  r2, K1, K2, beta1, beta2, m, D1, D2, dx, 4, gP0, gPL
   , gZ0, gZL, n*dt);
  [k2P, k2Z] = reaction_diffusion(P_RK4(:, n) + dt *
   k1P / 2,
  Z_RK4(:, n) + dt * k1Z / 2, r1, r2, K1, K2, beta1,
   beta2, m, D1,
  D2, dx, 4, gP0, gPL, gZ0, gZL, n*dt + dt/2);
  [k3P, k3Z] = reaction_diffusion(P_RK4(:, n) + dt *
   k2P / 2,
  Z_RK4(:, n) + dt * k2Z / 2, r1, r2, K1, K2, beta1,
   beta2, m, D1,
  D2, dx, 4, gP0, gPL, gZ0, gZL, n*dt + dt/2);
  [k4P, k4Z] = reaction_diffusion(P_RK4(:, n) + dt *
   k3P, Z_RK4(:,
  n) + dt * k3Z, r1, r2, K1, K2, beta1, beta2, m, D1,
   D2, dx, 4,
  gP0, gPL, gZ0, gZL, n*dt + dt);

  % Update densities using Runge-Kutta 4th order
   method
  P_RK4(:, n+1) = P_RK4(:, n) + dt / 6 * (k1P + 2*k2P
   + 2*k3P + k4P);
  Z_RK4(:, n+1) = Z_RK4(:, n) + dt / 6 * (k1Z + 2*k2Z
   + 2*k3Z + k4Z);

  % Applying boundary conditions
  P_RK4(1, n+1) = gP0(n * dt);
  P_RK4(end, n+1) = gPL(n * dt);
  Z_RK4(1, n+1) = gZ0(n * dt);
  Z_RK4(end, n+1) = gZL(n * dt);
end
```

Here k1, k2, k3, k4 are equal to f1, f2, f3, f4 and `P_RK4` , `Z_RK4` are $u_{k+1}$ in the 4th-order Runge-Kutta formula. Initial conditions are set for the first time step.

## Part 6: Exact solution

As the model is really complex, there are no accurate methods or approaches to find exact solutions [PBK22].

The problem is that, to determine the accuracy of the approximated solutions, I need the exact solution.

Since I do not have that solution, I considered an alternative method to still achieve good enough accuracy. This approach consist in using another approximate solution, with demonstrated better accuracy than the ones I want to test, as exact solution. For the 2nd-order and 4th-order Runge-Kutta methods, a proven better method [AYA20] is the 5th-order Runge-Kutta method.

The iterative formula for the 5th-order Runge-Kutta method is given by [LK65]:

$$u_{n+1} = u_n + \frac{k_1 + 5k_3 + 5k_5 + k_6}{12}$$

Where $k_i$ values are defined as:

$$k_1 = hf(t_n, u_n)$$

$$k_2 = hf\left(t_n + \frac{h}{2}, u_n + \frac{k_1}{2}\right)$$

$$k_3 = hf\left(t_n + \frac{(5-\sqrt{5})h}{10}, u_n + \frac{2k_1 + (3-\sqrt{5})k_2}{10}\right)$$

$$k_4 = hf\left(t_n + \frac{h}{2}, u_n + \frac{k_1 + k_2}{4}\right)$$

$$k_5 = hf\left(t_n + \frac{(5+\sqrt{5})h}{10}, u_n + \frac{(1-\sqrt{5})k_1 - 4k_2 + (5+3\sqrt{5})k_3 + 8k_4}{20}\right)$$

$$k_6 = hf\left(t_n + h, u_n + \frac{(\sqrt{5}-1)k_1 + (2\sqrt{5}-2)k_2 + (5-\sqrt{5})k_3 - 8k_4 + (10-2\sqrt{5})k_5}{4}\right)$$

The Matlab implementation follows the same structure of the other two methods (for the approximation of the second derivative we have used a 4th-order approximation, like for the 4th-order Runge-Kutta method, because the 6-th order has not showed a noticeable improvement of the solution):

```
P_RK5 = zeros(Nx, Nt); % Phytoplankton density using
   RK5
Z_RK5 = zeros(Nx, Nt); % Zooplankton density using RK5
P_RK5(:, 1) = P0(0);
Z_RK5(:, 1) = Z0(0);

for n = 1:Nt-1
  [k1P, k1Z] = reaction_diffusion(P_RK5(:, n), Z_RK5
   (:, n), r1, r2, K1, K2, beta1, beta2, m, D1, D2, dx
   , 4, gP0, gPL, gZ0, gZL, n*dt);

  [k2P, k2Z] = reaction_diffusion(P_RK5(:, n) + dt *
   k1P / 2, Z_RK5(:, n) + dt * k1Z / 2, r1, r2, K1, K2
   , beta1, beta2, m, D1, D2, dx, 4, gP0, gPL, gZ0,
   gZL, n*dt + dt/2);

  [k3P, k3Z] = reaction_diffusion(P_RK5(:, n) + dt *
   (2 * k1P + (3 - sqrt(5)) * k2P)/10, Z_RK5(:, n) +
   dt * (2 * k1Z + (3 - sqrt(5)) * k2Z)/10, r1, r2, K1
   , K2, beta1, beta2, m, D1, D2, dx, 4, gP0, gPL, gZ0
   , gZL, n*dt + (5 - sqrt(5)) * dt / 10);

  [k4P, k4Z] = reaction_diffusion(P_RK5(:, n) + dt *
   ((k1P + k2P) / 4), Z_RK5(:, n) + dt * ((k1Z + k2Z)
   / 4), r1, r2, K1, K2, beta1, beta2, m, D1, D2, dx,
   4, gP0, gPL, gZ0, gZL, n*dt + dt / 2);

  [k5P, k5Z] = reaction_diffusion(P_RK5(:, n) + dt *
   ((1 - sqrt(5)) * k1P - 4 * k2P + (5 + 3 * sqrt(5))
   * k3P + 8 * k4P)/20, Z_RK5(:, n) + dt * ((1 - sqrt
   (5)) * k1Z - 4 * k2Z + (5 + 3 * sqrt(5)) * k3Z + 8
   * k4Z)/20, r1, r2, K1, K2, beta1, beta2, m, D1, D2,
     dx, 4, gP0, gPL, gZ0, gZL, n*dt + (5 + sqrt(5)) *
    dt / 10);

  [k6P, k6Z] = reaction_diffusion(P_RK5(:, n) + dt *
    (((sqrt(5) - 1) / 4 * k1P + (2 * sqrt(5) - 2) / 4 *
     k2P + (5 - sqrt(5)) / 4 * k3P - 8 / 4 * k4P + (10
    - 2 * sqrt(5)) / 4 * k5P)), Z_RK5(:, n) + dt * (((
    sqrt(5) - 1) / 4 * k1Z + (2 * sqrt(5) - 2) / 4 *
    k2Z + (5 - sqrt(5)) / 4 * k3Z - 8 / 4 * k4Z + (10 -
     2 * sqrt(5)) / 4 * k5Z)), r1, r2, K1, K2, beta1,
    beta2, m, D1, D2, dx, 4, gP0, gPL, gZ0, gZL, n*dt +
     dt);

  % Update densities using Runge-Kutta 5th order
    method
  P_RK5(:, n+1) = P_RK5(:, n) + dt / 12 * (k1P + 5 *
    k3P + 5 * k5P + k6P);
  Z_RK5(:, n+1) = Z_RK5(:, n) + dt / 12 * (k1Z + 5 *
    k3Z + 5 * k5Z + k6Z);

  % Applying boundary conditions
  P_RK5(1, n+1) = gP0(n * dt);
  P_RK5(end, n+1) = gPL(n * dt);
  Z_RK5(1, n+1) = gZ0(n * dt);
  Z_RK5(end, n+1) = gZL(n * dt);
end
```

Therefore, from this moment I will use the 5th-order Runge-Kutta method as the exact solution.

## Part 7: Results

The density profiles of phytoplankton and zooplankton at the end of the simulation time T, using the Runge-Kutta 2nd and 4th-order methods, are shown below.

I expect the results of the Runge-Kutta 2nd and 4th-order methods to show similar trends, but the 4th-order method must have higher accuracy and stability due to the use of fourth-order central difference within the domain and second-order at the boundaries.

This is verified calculating the relative error between the exact solution and the two methods, as showed in Table 1.

We see that for both the 2nd-order and 4th-order Runge-Kutta methods the relative error decreases as you increase the number of spatial points Nx and time points Nt.

However, the error remains significantly larger in the 2nd-order method, which is expected because it has a lower order of accuracy.

For the 4th-order the relative error is not only much smaller compared to the 2nd-order but also decreases more rapidly as you increase Nx and Nt. This is typical because higher-order methods converge faster than lower-order methods.

In the 4th-order Runge-Kutta, the error should decreases proportional to $\mathcal{O}(\Delta t^4)$ this is not precisely what happens during these simulations. This can be explained because I did not have the true exact solution available but only a very accurate numerical method, the 5th-order Runge-Kutta method to make the comparison. Anyway, the results are still pretty coherent with what I should get.

Table 1: Relative errors at time T for RK2 and RK4 methods with different values of $N_x$ and $N_t$.

| $N_x$ | $N_t$ | **Relative Error RK2** | **Relative Error RK4** |
|---|---|---|---|
| 100 | 1000 | 0.0136 | 6.4594e-06 |
| 200 | 10000 | 0.0104 | 1.6891e-07 |
| 500 | 100000 | 0.0026 | 1.3241e-08 |

Table 2: Computation times for RK2 and RK4 methods with Nx=100, Nt=1000

| **Method** | **Computation Time (seconds)** |
|---|---|
| RK2 | 0.07 |
| RK4 | 0.33 |

```
Relative_Error_RK2 = norm(abs(P_RK2(:,end) -
    Exact_Solution(:,end)), inf) / norm(Exact_Solution
    (:,end), inf)
Relative_Error_RK4 = norm(abs(P_RK4(:,end) -
    Exact_Solution(:,end)), inf) / norm(Exact_Solution
    (:,end), inf)
```

We conclude:

- Central Difference and Second-Order Runge-Kutta method: This method is simpler and faster but may have less accuracy compared to higher-order methods.
- Fourth-Order approximation and Fourth-Order Runge-Kutta method: This method offers higher accuracy but is more complex and time-consuming.

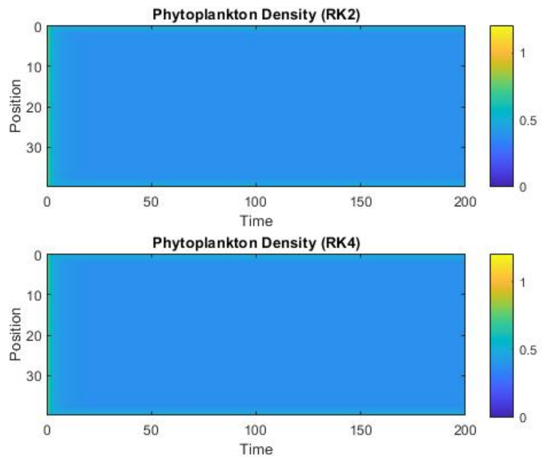

Figure 1: Phytoplankton and Zooplankton Density (RK2)

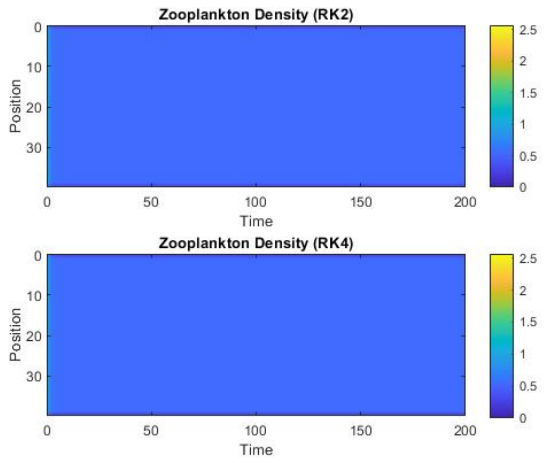

Figure 2: Phytoplankton and Zooplankton Density (RK4)

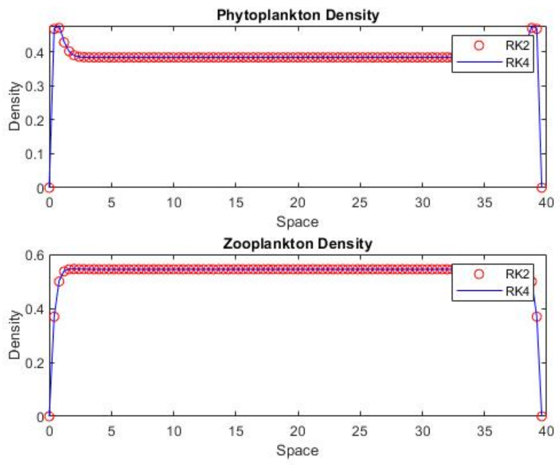

Figure 3: Phytoplankton and Zooplankton Density at end time T

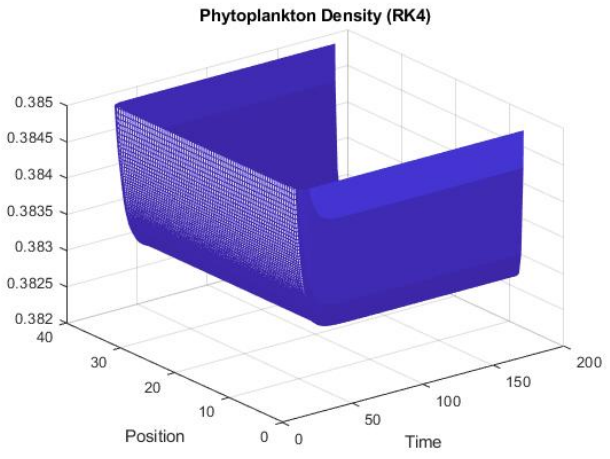

Figure 4: Phytoplankton density using 4th-order Runge-Kutta method

## References

[LK65] H. A. Luther and H. P. Konen. “Some Fifth-Order Classical Runge-Kutta Formulas”. In: *SIAM Review* 7.4 (1965), pp. 551–558.

[Wan+16] P. Wang et al. “Nonlinear dynamics of a marine phytoplankton-zooplankton system”. In: *Advances in Difference Equations* 2016.212 (2016).

[AYA20] Z. A. Adegboye, Y. A. Yahaya, and U. I. Ahmed. “Direct Integration of General Fourth Order Ordinary Differential Equations Using Fifth Order Runge-Kutta Method”. In: *Journal of the Nigerian Mathematical Society* 39.1 (2020), pp. 69–78.

[PBK22] R. N. Premakumari, Chandrali Baishya, and Mohammed K. A. Kaabar. “Dynamics of a fractional plankton-fish model under the influence of toxicity, refuge, and combine-harvesting efforts”. In: *Journal of Inequalities and Applications* 2022.137 (2022).