《系统仿真与matlab》综合试题

题 目：弱肉强食问题——Volterra问题

编 号：15

姓 名 马韬

班 级 人工智能2203班

学 号 U202215201

联系方式 15939391069

得分项	创新性	工作量	代码可读性	报告写作	总分
分数					

目录

一、实验题目

处于同一自然环境中两个种群之间的关系除了相互竞争和相互依存之外，还有一种更为有趣的生存方式：种群甲靠丰富的天然资源生长，而种群乙则靠掠食甲为生．地中海里的食用鱼与鲨鱼，加拿大森林中的美洲兔与山猫，阿尔卑斯山中的落叶松与芽虫等都是这种生存方式的典型.生态学上种群甲称为食饵 (Prey)，种群乙称为捕食者(Predator)，二者共处组成食饵一捕食者系统(简称P—P系统).

模型假设

食饵和捕食者在时刻t的数量分别记作$x_1(t)$和$x_2(t)$，因为大海中资源丰富．可以假设如果食饵独立生存则将以增长率r_1按指数规律增长，即有$\dot{x}_1 = r_1 x_1$．捕食者的存在使食饵的增长率降低，设降低的程度与捕食者数量成正比．于是$x_1(t)$满足方程

$\dot{x}_1(t) = x_1(r_1 - \lambda_1 x_2)$

比例系数λ_1，反映捕食者掠取食饵的能力。

捕食者离开食饵无法生存，若设它独自存在时死亡率为r_2，即$\dot{x}_2 = -r_2 x_2$，而食饵为它提供食物的作用相当于使死亡率降低、或使之增长．设这个作用与食饵数量成正比，于是$x_1(t)$满足$\dot{x}_2(t) = x_2(-r_2 + \lambda_2 x_1)$比例系数$\lambda_2$，反映食饵对捕食者的供养能力.

上述方程是在没有人工捕获情况下自然环境中食饵与捕食者之间的制约关系，是Volterra提出的最简单的模型．可以看出这个模型没有考虑自身的阻滞作用.

仿真要求　系统输入为仿真时间T，食饵和捕食者在初始时刻的数量$x_1(0)$和$x_2(0)$，食饵独立生存增长率r_1，捕食者掠取食饵的能力λ_1，捕食者独自存在时死亡率r_2，食饵对捕食者的供养能力λ_2，系统输出为食饵和捕食者在时刻t的数量$x_1(t)$和$x_2(t)$。要求有输入、输出界面及仿真过程。

二、建模分析

这个问题描述的是经典的食饵—捕食者模型，是由生态学家 **Volterra** 提出的，其中两个物种（食饵和捕食者）相互影响，模型使用微分方程描述它们之间的动态关系。下面分析相关建模：

2.1 食饵建模

食饵数量x(t)在没有捕食者的情况下按照指数规律增长。增长率为 r 。但是，捕食者的存在会降低食饵的增长速率，假设这种降低程度与捕食者的数量成正比。因此，食饵的变化率可以表示为：

$$\frac{dx}{dt} = rx(t) - \lambda x(t)y(t) \tag{1}$$

其中，x(t)为食饵数量，y(t)为捕食者数量，λ 是捕食者对食饵的掠夺能力（比例系数），它反映了捕食者对食饵种群的影响。

2.2 捕食者建模

捕食者离开食饵无法生存，只有捕食食饵才能生存。捕食者死亡率为ϑ，当捕食者数量y(t)大于零时，它们的死亡率减少，且与食饵的数量成正比。

因此，捕食者的变化率可以表示为：

$$\frac{dy}{dt} = \beta x(t)\mathrm{y}(t) - \vartheta y(t) \tag{2}$$

其中，β是食饵对捕食者的供养能力（比例系数），表示食饵数量对捕食者数量增长的影响。δ 是捕食者的自然死亡率。

而本文实现了两个捕食者的建模。

2.3 Lotka-Volterra 竞争模型建模

在查阅相关资料后，我将上述基本模型进行了一定改进，添加了一个同级的竞争捕食者，并且在公式中添加竞争系数α_{ij} 代表捕食者i和捕食者j在该环境下由于竞争导致的数量变化参数。因此上述公式变化为：

$$\frac{dx}{dt} = rx(t) - \lambda x(t)y(t) \tag{1}$$

$$\frac{dy}{dt} = \beta x(t)\mathrm{y}(t) - \vartheta y(t) - \alpha y'(t) \tag{3}$$

在本次中我使用ode45( )函数对上述微分方程组进行运算，得到捕食者1、2和食饵的数量变化。并且通过不断地刷新更新界面得到动态的画面，从而实现动画的效果。

2.4 拓展：捕食者-食物链模型建模

捕食者-食物链模型是描述生态系统中不同物种之间相互依存与相互作用的一个重要数学模型。特别是在食物链中，捕食者与食物（或称食饵）之间的相互作用直接影响其种群的增长与衰退。食物链通常描述了食物（资源）从低级消费者（如植物）到高层消费者（如捕食者）如何在物种之间流动。可以构建模型如下：

$$\frac{dx}{dt} = r \cdot x(t) - \alpha x(t) y(t) \tag{4}$$

$$\frac{dy}{dt} = \beta x(t) \cdot y(t) - \gamma \cdot y(t) \tag{5}$$

其中，r 是食饵的增长率，α 是捕食者捕食食饵的率，β 是捕食者的繁殖率，γ 是捕食者的死亡率。

在本次中我使用ode45( )函数对上述微分方程组进行运算，得到捕食者和食饵的数量变化。并且通过不断地刷新更新界面得到动态的画面，从而实现动画的效果。

2.5 拓展：捕食-竞争模型建模

在这个新模型中，将添加一个捕食者与食饵的竞争关系，其中捕食者和食饵之间不仅存在捕食互动，还考虑到捕食者之间的竞争效应。这个模型的核心思想是两个捕食者共享同一资源（食饵），但它们之间还存在一定的竞争关系。我们可以通过调整捕食者之间的竞争系数来模拟不同的生态情景。

食饵的动态：食饵的数量变化取决于食饵的生长率和两类捕食者的捕食行为，但还需要考虑食物资源供给。

$$\frac{dN_1}{dt} = r_1 N_1 (1 - \frac{N_1}{K_1}) - \alpha_1 N_1 P_1 - \alpha_2 N_1 P_2 + S \tag{6}$$

其中，r_1是食饵的增长率。K_1是食饵的环境承载力（最大承载量）。α_1和α_2是捕食者1和捕食者2对食饵的捕食系数。P_1和P_2分别是捕食者1和捕食者2的种群数量。S是食物供给率。

捕食者1的动态：捕食者1的种群变化由食饵的数量、捕食者与食饵之间的捕食行为，以及捕食者之间的竞争影响。

$$\frac{dP_1}{dt} = \beta_2 \frac{N_1 P_2}{1 + h_2 P_2} - d_2 P_2 - c_2 P_1 P_2 \tag{7}$$

其中，β_1是捕食者1的捕食效率，d_1是捕食者1的死亡率。C_1是捕食者1和捕食者2之间的竞争系数。h_1是捕食者1对食饵的响应系数。

捕食者2的动态：与捕食者1类似，但包含不同的竞争效应。

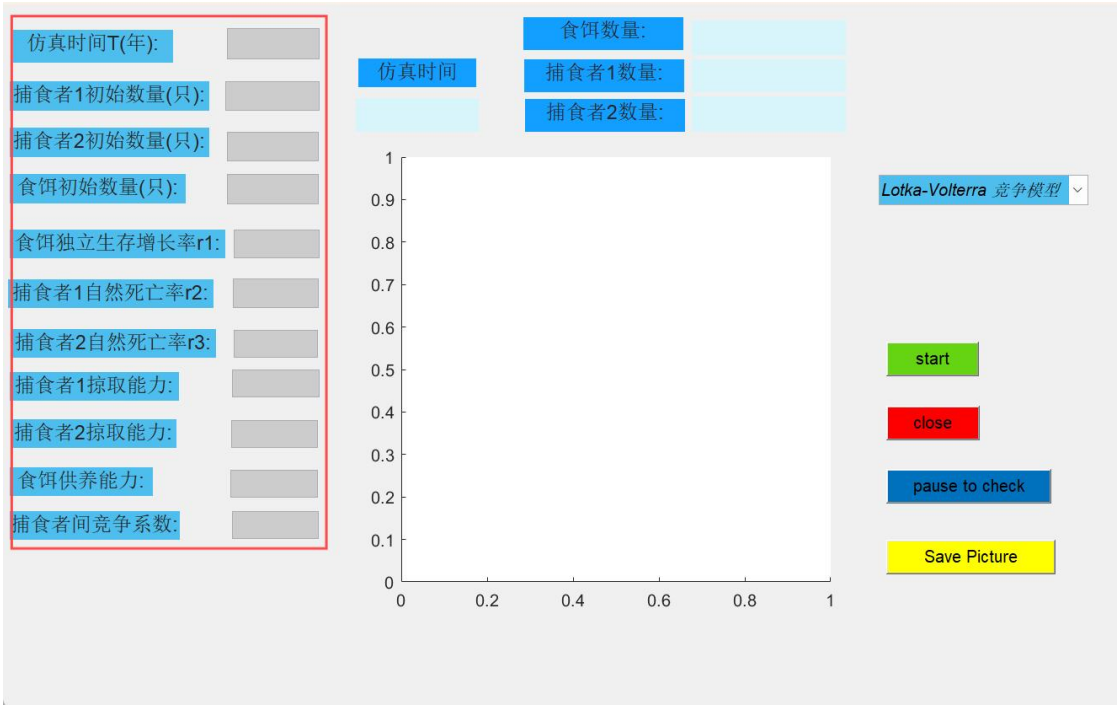
$$\frac{dP_2}{dt} = \beta_2 \frac{N_1 P_2}{1 + h_2 P_2} - d_2 P_2 - c_2 P_1 P_2 \tag{8}$$

其中，β_2是捕食者2的捕食效率。d_2是捕食者2的死亡率. c_2是捕食者1与捕食者2之间的竞争系数。h_2是捕食者2对食饵的响应系数。

在本次中我使用ode45( )函数对上述微分方程组进行运算，得到捕食者1、2和食饵的数量变化。并且通过不断地刷新更新界面得到动态的画面，从而实现动画的效果。

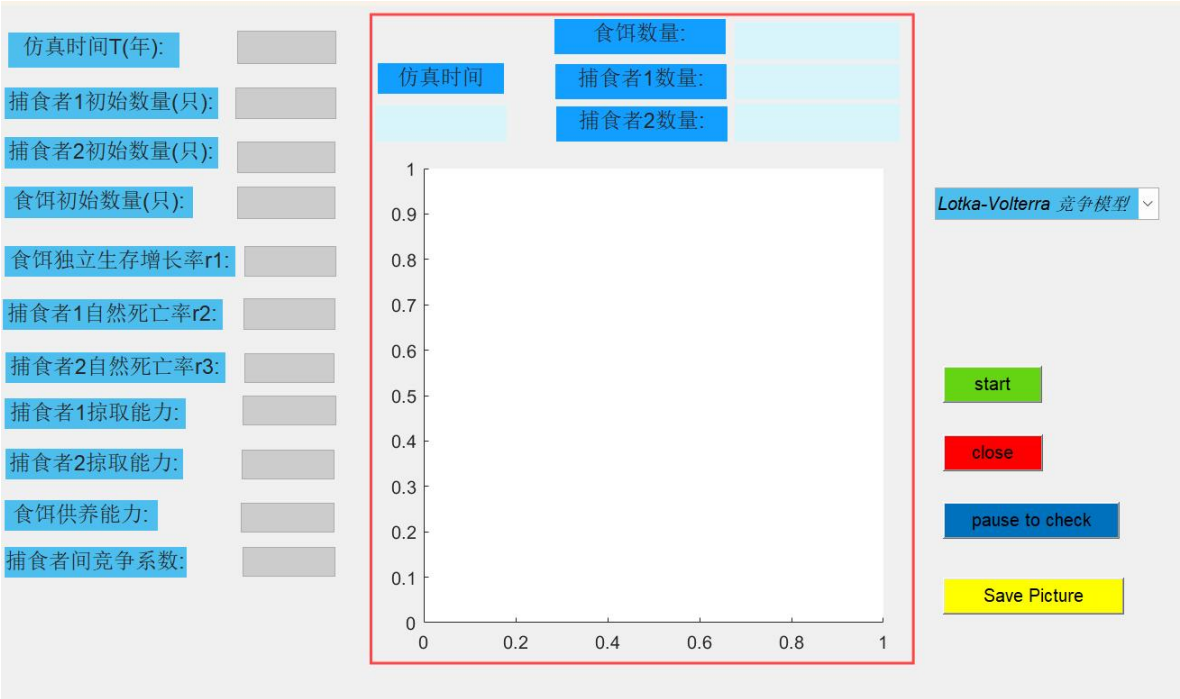
三、功能分析

3.1 输入功能



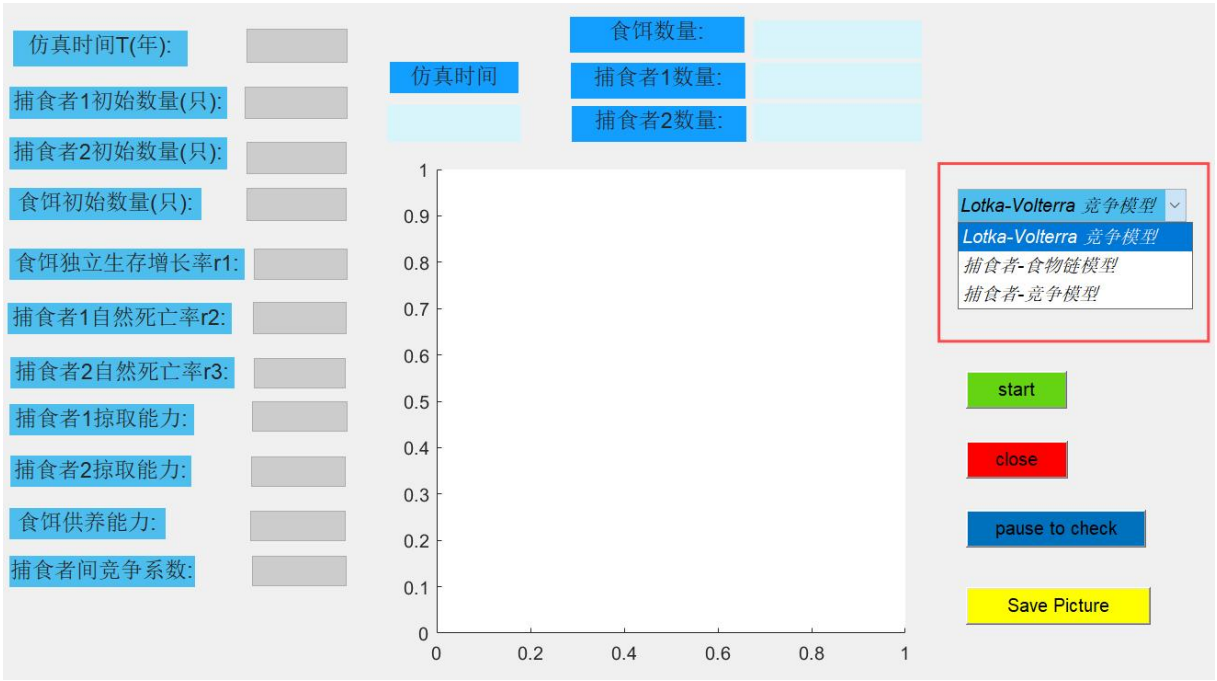
红框即为GUI界面的输入部分，点击灰框输入相关参数即可。

3.2 显示功能



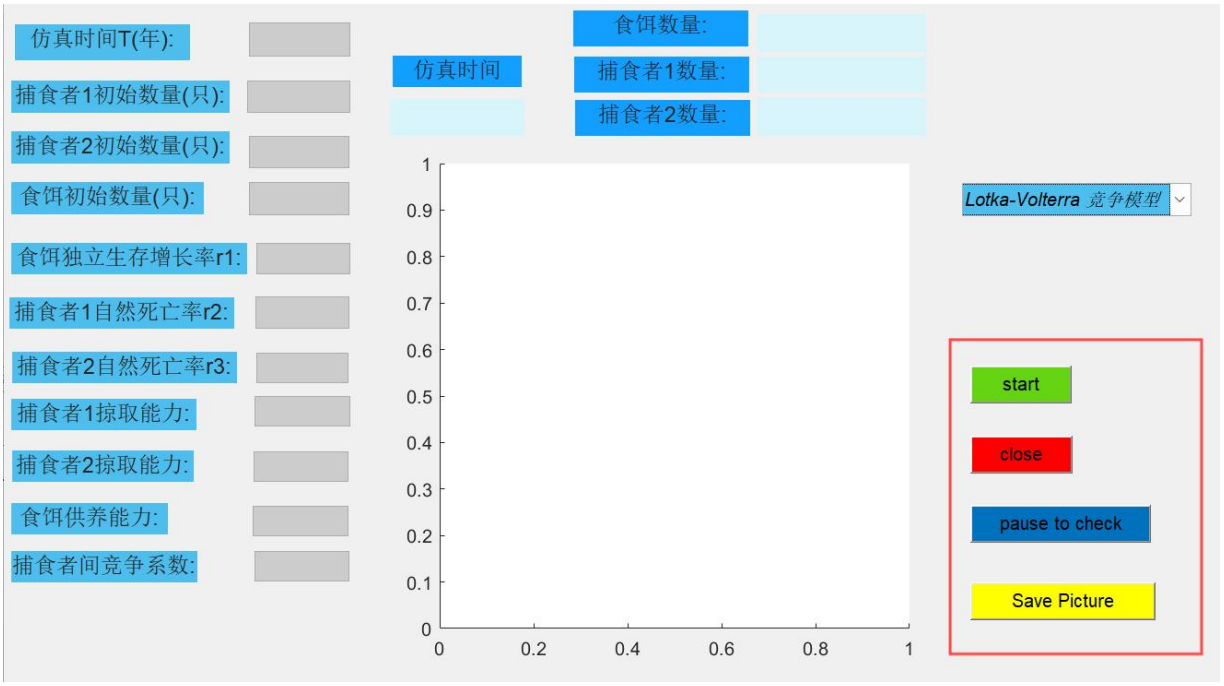
红框处即为结果显示模块，上方会实时显示随着仿真时间变化的数值，包括食饵数量、捕食者1数量、捕食者2数量。下方会显示绘图，也是随着时间进行的动画展示。可视化丰富，可以明显看出仿真结果

3.3 模型选择功能



红框标识处即为模型选择模块，共有三个模型进行选择，分别是Lotka-Volterra竞争模型、捕食者-食物链模型、捕食者-竞争模型。模型的详细建模在上方已详细阐述，通过模型的选择然后利用输入的参数进行结果的可视化和仿真。

3.4 按钮功能



四个按钮对应了四个不同的功能：

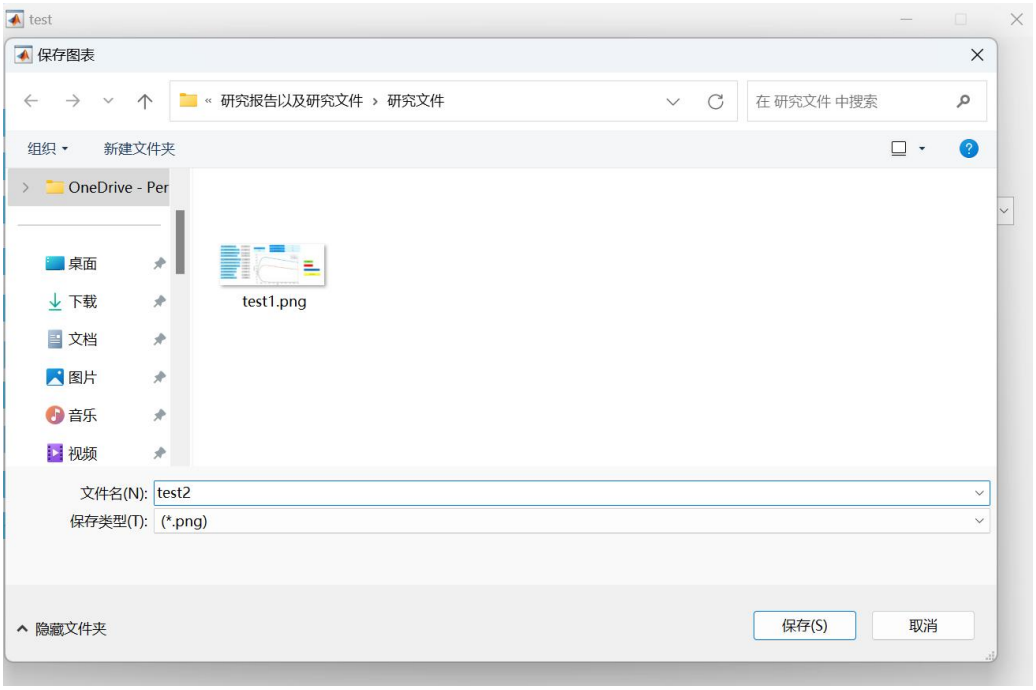
start按钮：开始仿真，当参数输入完毕，选好模型后，就可以按下该按钮开始仿真。

close按钮:不论何时按下该按钮都可以结束仿真，然后重新修改参数或者选择模型。

Pause to check按钮：当仿真开始时，按下按钮可以暂停仿真查看当时的所有仿真结果和相关参数。

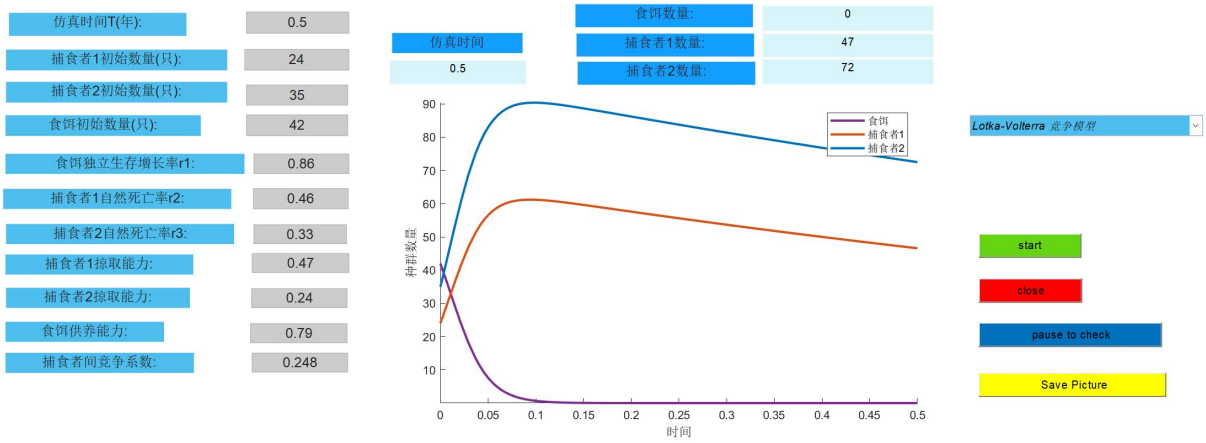
Save picture按钮：当仿真结束后，可以按下该按钮然后会弹出文件夹管理器，输入保存的图片名称和路径即可保存这次仿真的相关参数和最终结果。

保存时的界面：



保存后提交显示：

保存后的图像：



四、关键难点

4.1 建模难点

该模型的建模是一组耦合的非线性微分方程，系统的解可能表现出复杂的动态行为。例如，捕食者和食物种群的数量可能会呈现周期性波动、稳定态或甚至混沌行为。如何准确分析和理解这些非线性动态行为，当时我不知道matlab有什么好的方法，直到我上网查阅之后，才掌握了ode45( )函数。

4.2 代码难点

当建模完成之后就是代码的实现，首先刚开始操作的时候还不太清楚GUI界面的绘制和函数回调导致没办法让代码和GUI界面很好的连接，但之后查阅了PPT和相关资料后，这方面的难点已经攻克。

然后就是主函数的实现，我在start按钮的回调函数定为主函数，然后定义了众多变量，由于我区分了三个模型，因此我需要分别构建微分模型然后实现，并且在绘图以及模型选择方面都要进行分类逻辑，写完后，发现动画效果一般，因为横轴纵轴的尺度不是跟随变化的，因此动画效果大打折扣，于是我又在动画绘图模块添加了一个动态调整的模块，最终动画效果实现成功。

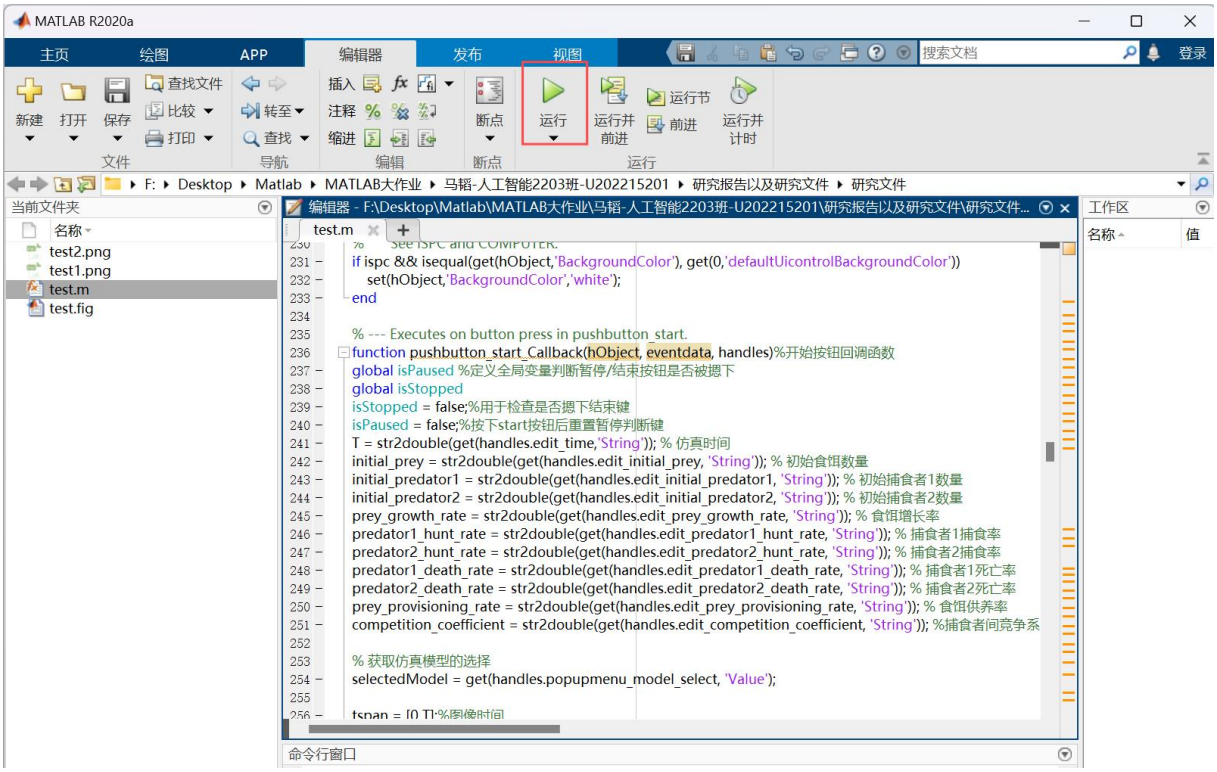
之后就是按钮的嵌入，每个按钮有不同的功能，因此要嵌入程序不同的位置，所以导致我多次嵌入无效，只有理清了逻辑最后才能攻克这个难点。通过改变全局变量的参数对图像显示进行控制的时候，一开始采用了在初始化GUI界面的参数设定中设定句柄参数，但是在检测过程中发现在摁动按钮后该句柄中的参数并没有随着摁动的变化而变化，在尝试debug多次后我选择了另外一种方式，即放弃在初始化函数中设定额外的句柄参数而改用全局变量。在采用全局变量的方法后，成功的实现了按钮的功能。

4.3 绘图难点

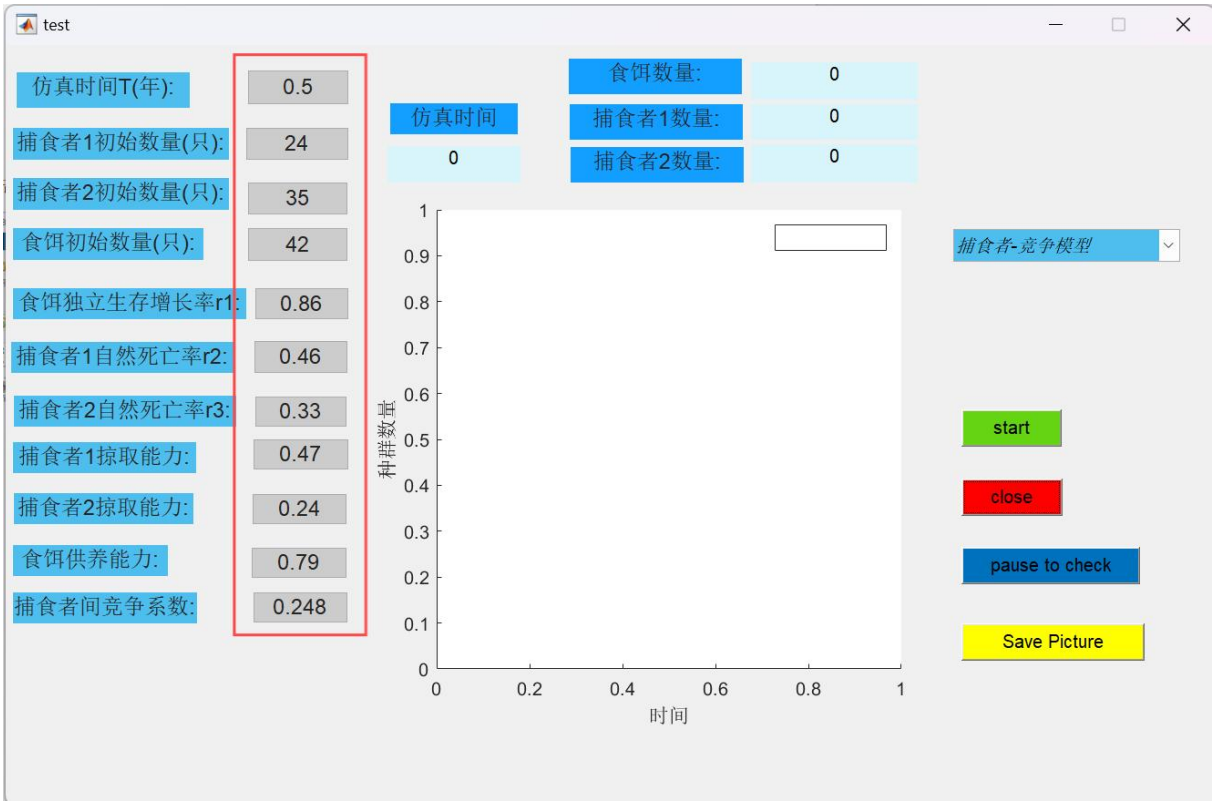
绘图的其他方面没有什么问题，主要的难点在于首先我认为那个界面空间不大，后续发现可以调整，解决了第一个难点，第二个难点就是下拉式菜单，关于设置多个旋向的问题，试了多次没有设置成功，后续通过网上查阅资料才明白如何分类。

五、程序运行指南

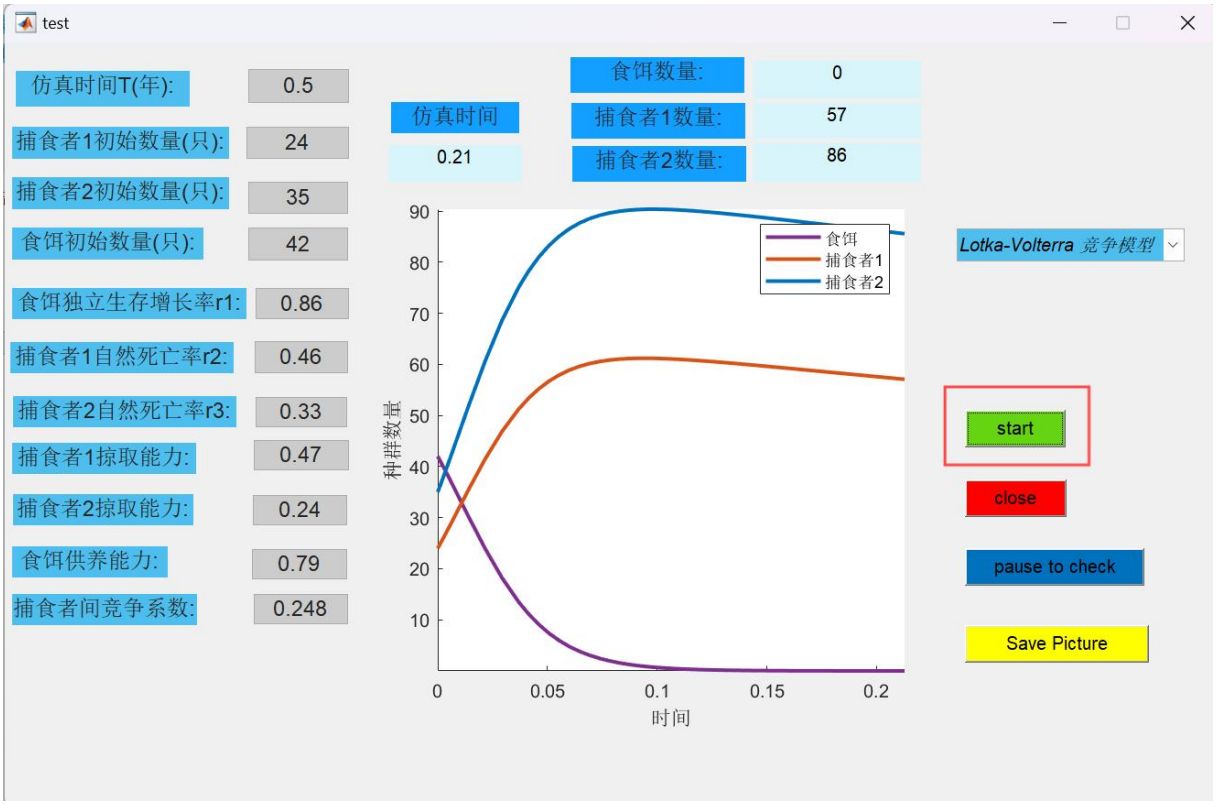
1. 点击test.m，让源路径在该文件夹下，点击运行



2. 输入相关参数，可以自己定义，这里我给一个示例

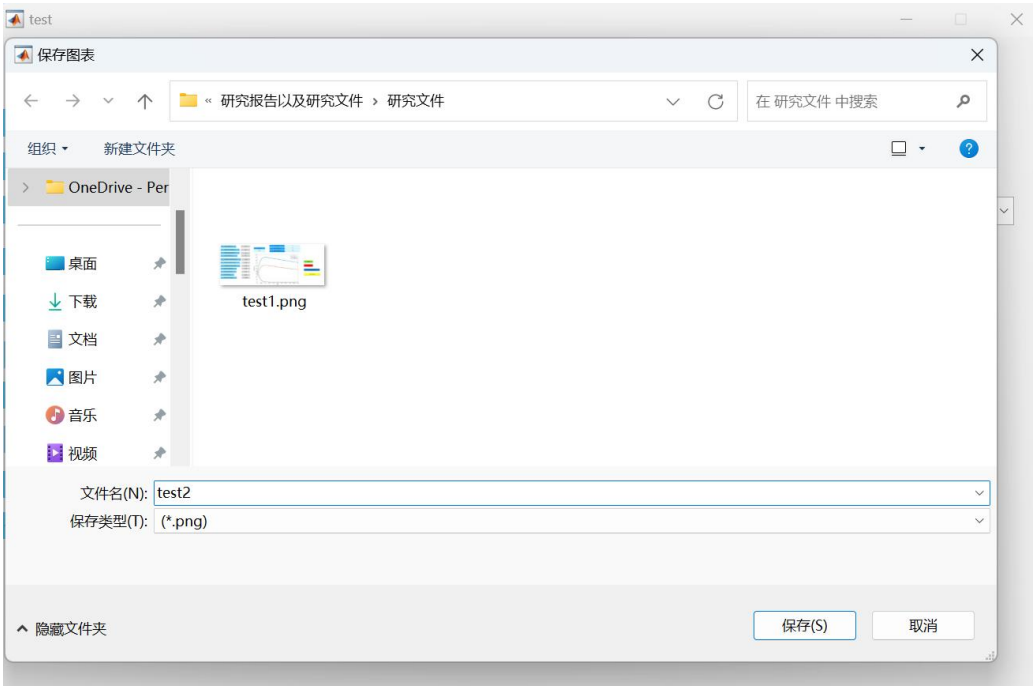


3. 点击start，开始仿真



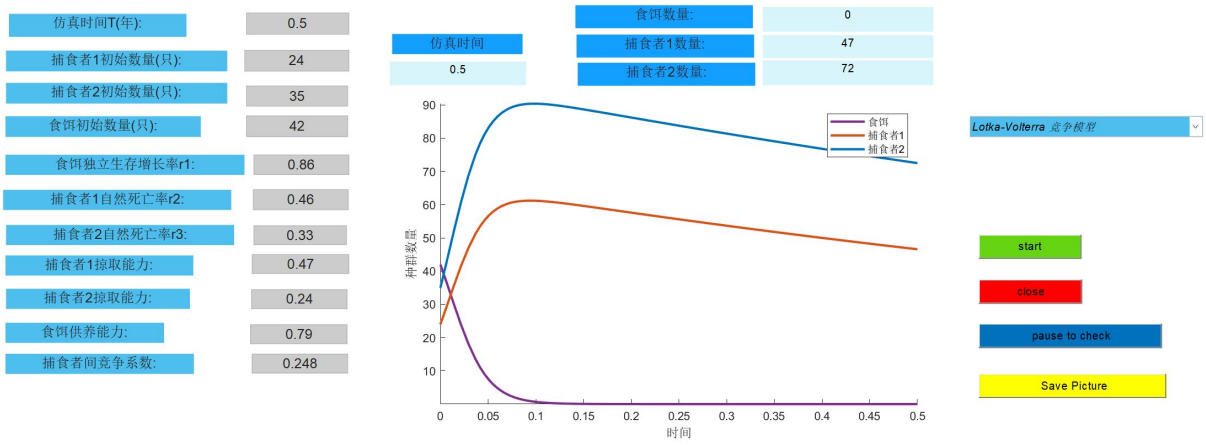
4. 仿真结束后，可以点击Save Picture,保存图片

保存时的界面：

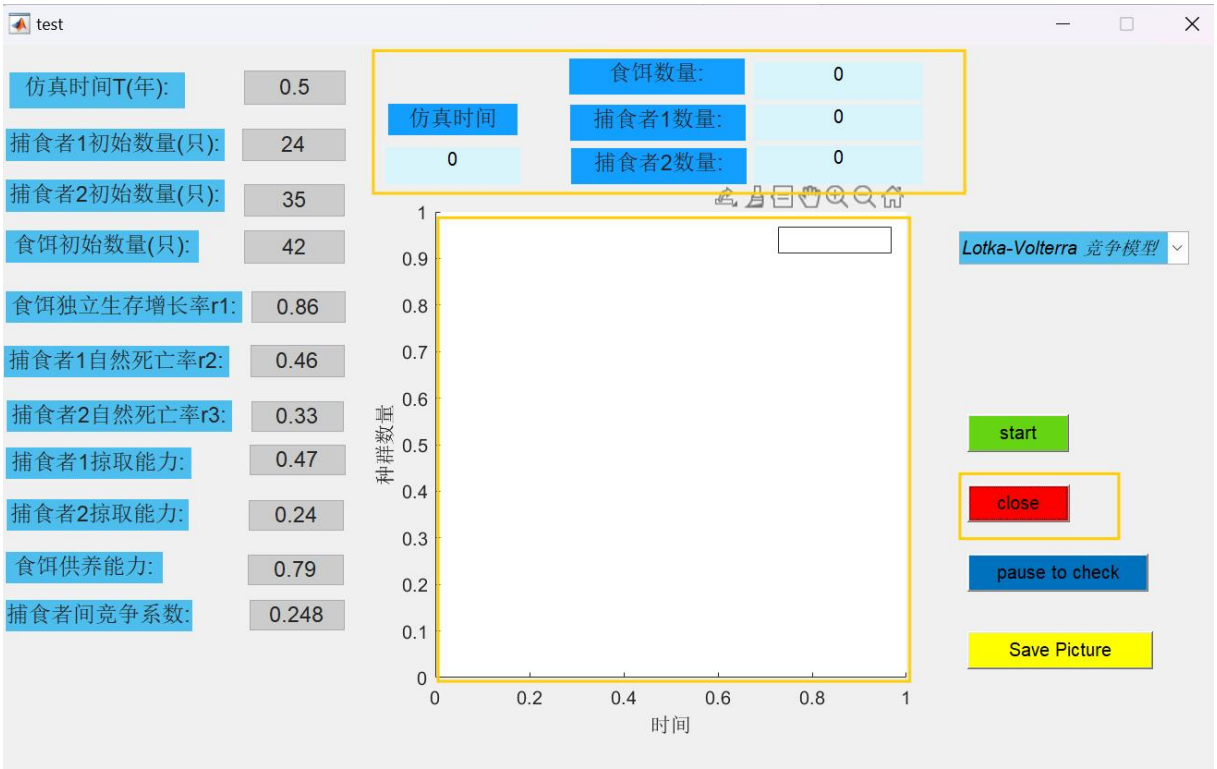


保存后提交显示：

保存后的图像：



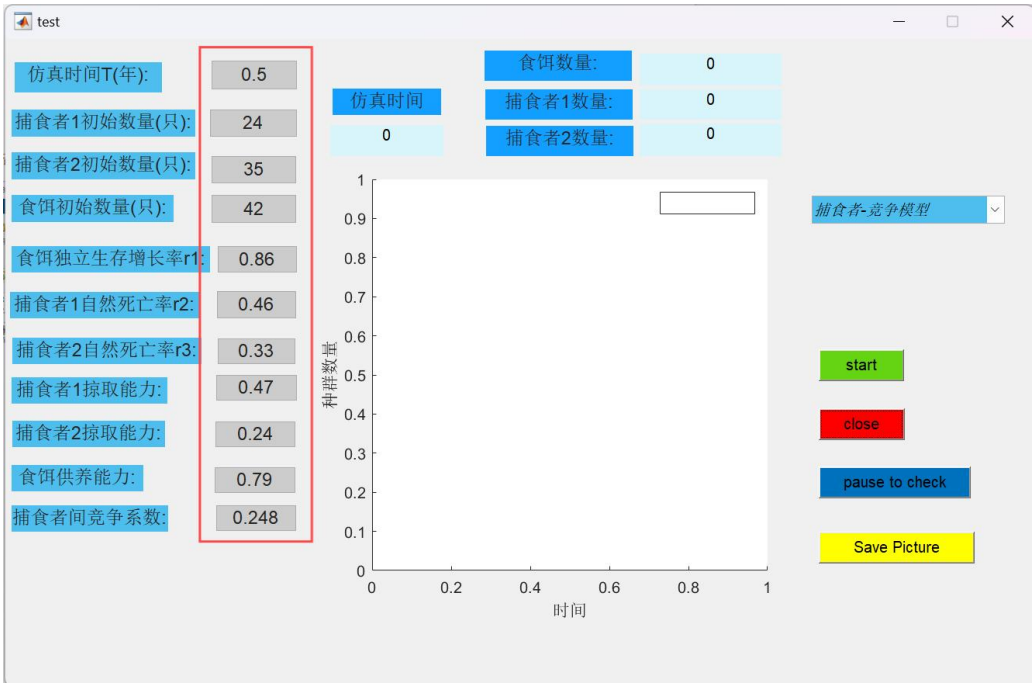
5. 点击close，清除参数和仿真



6. 点击pause to check,可以暂停10秒，然后继续运行

六、程序运行实例分析

分别用这个参数实例对三个模型进行分析：



对模型一：Lotka-Volterra竞争模型：

$$\frac{dN_1}{dt} = r_1 N_1 - a_1 N_1 N_2 - b_1 N_1 N_3$$

$$\frac{dN_2}{dt} = -r_2 N_2 + c_1 N_1 N_2 - d_{12} N_2 N_3$$

$$\frac{dN_3}{dt} = -r_3 N_3 + c_2 N_1 N_3 - d_{21} N_3 N_2$$

N_1：食饵数量，初始值为 42。

N_2：捕食者 1 数量，初始值为 24。

N_3：捕食者 2 数量，初始值为 35。

$r_1 = 0.86$：食饵的独立增长率。

$r_2 = 0.46, r_3 = 0.33$：捕食者的自然死亡率。

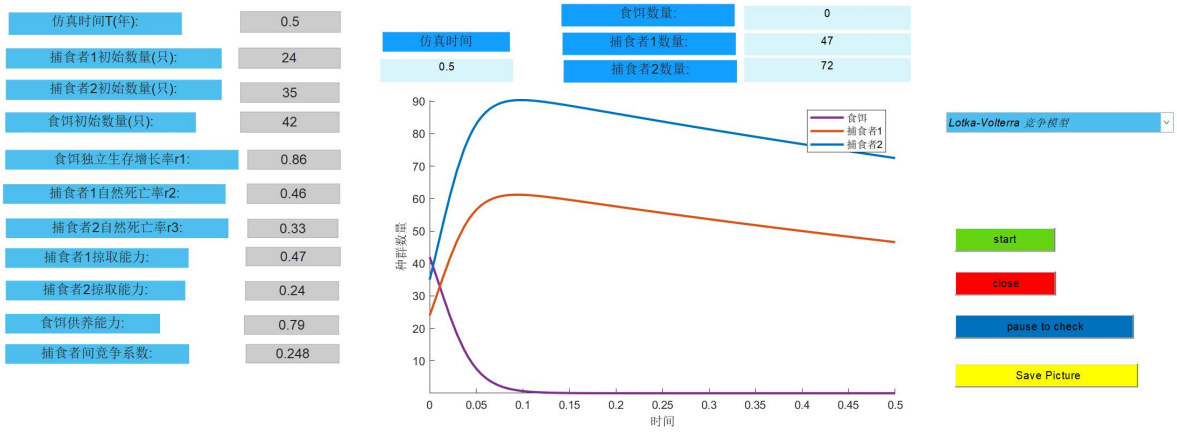
$a_1 = 0.47, b_1 = 0.24$：捕食者掠取能力。

$d_{12} = d_{21} = 0.248$：捕食者间的竞争系数。

$c_1 = c_2 = 0.79$：食饵对捕食者的供养能力。

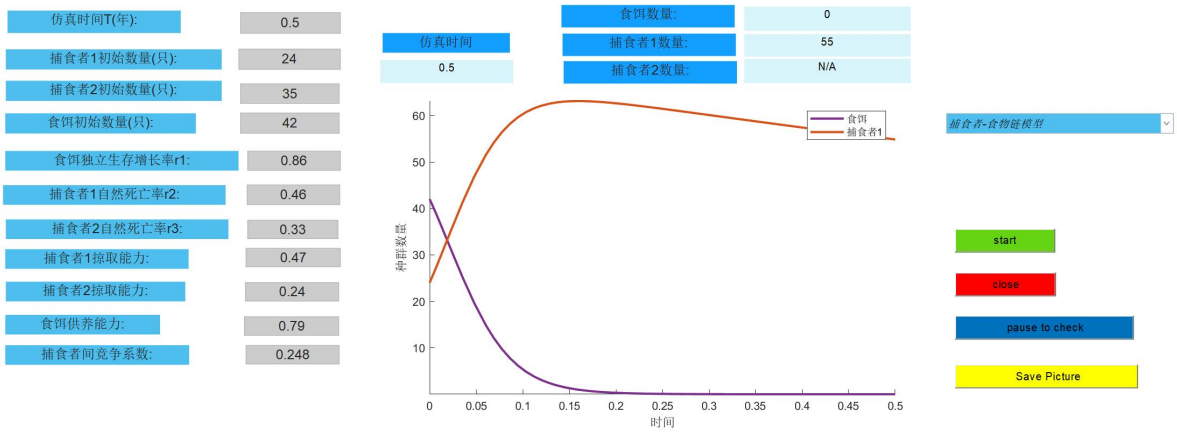
食饵数量快速增长后受到捕食者掠取能力的抑制。

捕食者 1 和捕食者 2 的数量随食饵数量的变化而波动，同时因彼此竞争而进一步限制各自种群增长。因此有如下结果：



对模型二：捕食者-食物链模型：

只考虑了捕食者1和食饵的参数，使用上面提出的捕食者-食物链模型：捕食者1和食饵是一个反比例关系，如下：



对模型三：捕食者-竞争模型：

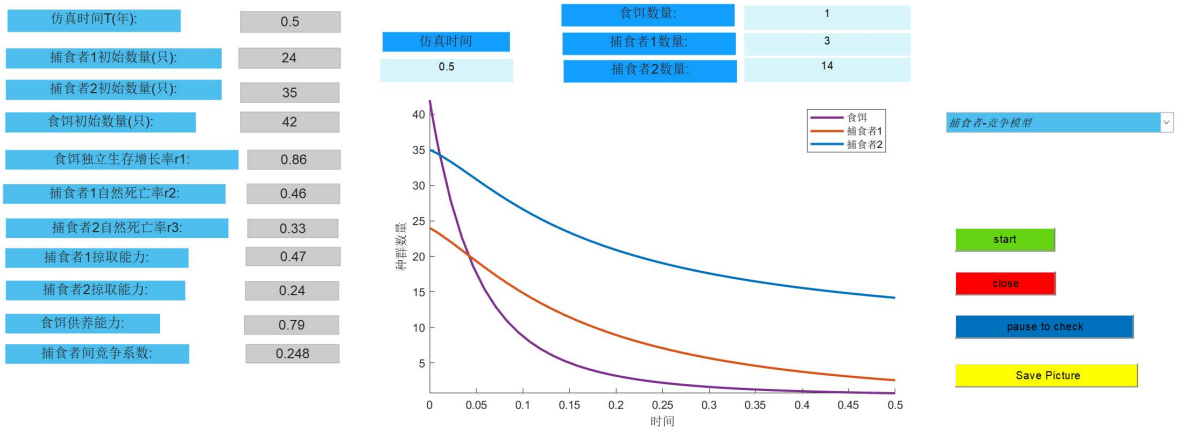
在该模型中，捕食者 1 和捕食者 2 对食饵的掠取能力直接影响各自的种群数量

$$\frac{dN_1}{dt} = r_1 N_1 (1 - \frac{N_1}{K_1}) - \alpha_1 N_1 P_1 - \alpha_2 N_1 P_2 + S$$

$$\frac{dP_1}{dt} = \beta_2 \frac{N_1 P_2}{1 + h_2 P_2} - d_2 P_2 - c_2 P_1 P_2$$

$$\frac{dP_2}{dt} = \beta_2 \frac{N_1 P_2}{1 + h_2 P_2} - d_2 P_2 - c_2 P_1 P_2$$

利用上述参数，可得出结果：



七、代码展示

```
%GUI界面
function varargout = test(varargin)
    gui_Singleton = 1;
    gui_State = struct('gui_Name',       mfilename, ...
                       'gui_Singleton',  gui_Singleton, ...
                       'gui_OpeningFcn', @test_OpeningFcn, ...
                       'gui_OutputFcn',  @test_OutputFcn, ...
                       'gui_LayoutFcn',  [] , ...
                       'gui_Callback',   []);
    if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
    end

    if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
end

%start按钮回调函数
function pushbutton_start_Callback(hObject, eventdata, handles)%开始按钮回调函数
    global isPaused %定义全局变量判断暂停/结束按钮是否被摁下
    global isStopped
    isStopped = false;%用于检查是否摁下结束键
    isPaused = false;%按下start按钮后重置暂停判断键
    T = str2double(get(handles.edit_time,'String')); % 仿真时间
    initial_prey = str2double(get(handles.edit_initial_prey, 'String')); % 初始食饵数量
```

```
28.       initial_predator1 = str2double(get(handles.edit_initial_predator1, '
   String')); % 初始捕食者1数量
29.       initial_predator2 = str2double(get(handles.edit_initial_predator2, '
   String')); % 初始捕食者2数量
30.       prey_growth_rate = str2double(get(handles.edit_prey_growth_rate, 'St
   ring')); % 食饵增长率
31.       predator1_hunt_rate = str2double(get(handles.edit_predator1_hunt_rat
   e, 'String')); % 捕食者1捕食率
32.       predator2_hunt_rate = str2double(get(handles.edit_predator2_hunt_rat
   e, 'String')); % 捕食者2捕食率
33.       predator1_death_rate = str2double(get(handles.edit_predator1_death_r
   ate, 'String')); % 捕食者1死亡率
34.       predator2_death_rate = str2double(get(handles.edit_predator2_death_r
   ate, 'String')); % 捕食者2死亡率
35.       prey_provisioning_rate = str2double(get(handles.edit_prey_provisioni
   ng_rate, 'String')); % 食饵供养率
36.       competition_coefficient = str2double(get(handles.edit_competition_co
   efficient, 'String')); %捕食者间竞争系数
37.
38.       % 获取仿真模型的选择
39.       selectedModel = get(handles.popupmenu_model_select, 'Value');
40.
41.       tspan = [0 T];%图像时间
42.
43.       % 根据选择的模型选择ODE方程
44.       switch selectedModel
45.           case 1  % Lotka-Volterra竞争模型
46.               odeFunc = @(t, populations) lotka_volterra_competition(t, po
   pulations, prey_growth_rate, predator1_hunt_rate, predator2_hunt_rate, pre
   dator1_death_rate, predator2_death_rate, prey_provisioning_rate, competiti
   on_coefficient);
47.               % 设定初始种群
48.               initial_populations = [initial_prey, initial_predator1, init
   ial_predator2];
49.           case 2  % 捕食者-食物链模型
50.               odeFunc = @(t, populations) predator_prey_food_chain(t, popu
   lations, prey_growth_rate, predator1_hunt_rate, predator1_death_rate);
51.               % 设定初始种群
52.               initial_populations = [initial_prey, initial_predator1]; %
   只有食饵和捕食者1
53.           case 3  % 捕食者-竞争模型
54.           odeFunc = @(t, populations) predator_competition_model(t, popula
   tions, prey_growth_rate, 1000, predator1_hunt_rate, predator2_hunt_rate, p
   rey_provisioning_rate, 0.5, 0.1, predator1_death_rate, competition_coeffic
   ient, 0.5, 0.1, predator2_death_rate, competition_coefficient);
55.           initial_populations = [initial_prey, initial_predator1, initial_
   predator2];
56.           otherwise
57.               disp('Invalid Model Selected');
58.               return;
59.       end
```

```
60.
61.         % 求解ODE
62.       [t, populations] = ode45(odeFunc, tspan, initial_populations);
63.
64.       % 绘图
65.   for i = 1:length(t)
66.       if ~isPaused && ~isStopped
67.           cla; % 清除当前图像
68.           hold on;
69.
70.           % 根据选择的模型，绘制不同数量的种群图像
71.           if selectedModel == 1  % Lotka-Volterra竞争模型
72.               plot(t(1:i), populations(1:i, 1), 'color', [0.4940 0.1840 0.
   5560], 'LineWidth', 2, 'DisplayName', '食饵'); % 食饵图像
73.               plot(t(1:i), populations(1:i, 2), 'color', [0.8500 0.3250 0.
   0980], 'LineWidth', 2, 'DisplayName', '捕食者1'); % 捕食者1图像
74.               plot(t(1:i), populations(1:i, 3), 'color', [0.000 0.4470 0.7
   410], 'LineWidth', 2, 'DisplayName', '捕食者2'); % 捕食者2图像
75.
76.           elseif selectedModel == 2  % 捕食者-食物链模型
77.               plot(t(1:i), populations(1:i, 1), 'color', [0.4940 0.1840 0.
   5560], 'LineWidth', 2, 'DisplayName', '食饵'); % 食饵图像
78.               plot(t(1:i), populations(1:i, 2), 'color', [0.8500 0.3250 0.
   0980], 'LineWidth', 2, 'DisplayName', '捕食者1'); % 捕食者1图像
79.
80.           elseif selectedModel == 3  % 捕食者-竞争模型
81.               plot(t(1:i), populations(1:i, 1), 'color', [0.4940 0.1840 0.
   5560], 'LineWidth', 2, 'DisplayName', '食饵'); % 食饵图像
82.               plot(t(1:i), populations(1:i, 2), 'color', [0.8500 0.3250 0.
   0980], 'LineWidth', 2, 'DisplayName', '捕食者1'); % 捕食者1图像
83.               plot(t(1:i), populations(1:i, 3), 'color', [0.000 0.4470 0.7
   410], 'LineWidth', 2, 'DisplayName', '捕食者2'); % 捕食者2图像
84.           end
85.
86.           xlabel('时间');
87.           ylabel('种群数量');
88.
89.           % 更新文本数据
90.           set(handles.text_time, 'String', num2str(round(t(i) * 10^2) / 10
   ^2)); % 更新时间数据（保留两位小数）
91.           set(handles.text_prey_data, 'String', num2str(round(populations(
   i, 1)))); % 食饵数据
92.           set(handles.text_predator1_data, 'String', num2str(round(populat
   ions(i, 2)))); % 捕食者1数据
93.
94.           if selectedModel == 1  % 只有Lotka-Volterra模型有捕食者2
95.               set(handles.text_predator2_data, 'String', num2str(round(pop
   ulations(i, 3)))); % 捕食者2数据
96.           elseif selectedModel == 2  % 捕食者-食物链模型没有捕食者2
```

```
                set(handles.text_predator2_data, 'String', 'N/A'); % 捕食者2数据为空
            else  % 捕食者-竞争模型
                set(handles.text_predator2_data, 'String', num2str(round(populations(i, 3)))); % 捕食者2数据
            end

            % 自动调整坐标轴
            axis tight;  % 使坐标轴自动适应数据范围

            drawnow;
            pause(0.05); % 短暂暂停使界面刷新更加流畅
        end

        if isPaused && ~isStopped  % 按下暂停键
            pause(10); % 暂停10s进行数据记录后继续画图
            isPaused = false; % 重置暂停键
        end
        if isStopped && ~isPaused  % 按下结束键
            break;
        end
    end

    legend('show'); % 显示图例


    %Lotka-Volterra竞争模型
    function dpdt = lotka_volterra_competition(t, populations, prey_growth_rate, predator1_hunt_rate, predator2_hunt_rate , predator1_death_rate, predator2_death_rate, prey_provisioning_rate, competition_coefficient)%竞争模型
        prey_populations = populations(1);%食饵
        predator1_populations = populations(2);%捕食者1
        predator2_populations = populations(3);%捕食者2

        dN1dt = prey_populations*(prey_growth_rate - predator1_hunt_rate*predator1_populations - predator2_hunt_rate*predator2_populations);%食饵的微分方程
        dN2dt = predator1_populations*(-predator1_death_rate + prey_provisioning_rate*prey_populations - competition_coefficient);%捕食者1的微分方程
        dN3dt = predator2_populations*(-predator2_death_rate + prey_provisioning_rate*prey_populations - competition_coefficient);%捕食者2的微分方程
        dpdt = [dN1dt; dN2dt; dN3dt];

      % 捕食者-食物链模型
    function dpdt = predator_prey_food_chain(t, populations, prey_growth_rate, predator_hunt_rate, predator_death_rate)
        prey_populations = populations(1); % 食饵数量
        predator_populations = populations(2); % 捕食者数量

        % 捕食者-食物链模型的微分方程
```

```
    dPrey_dt = prey_growth_rate * prey_populations - predator_hunt_rate * prey_populations * predator_populations; % 食饵的变化率
    dPredator_dt = predator_hunt_rate * prey_populations * predator_populations - predator_death_rate * predator_populations; % 捕食者的变化率

    dpdt = [dPrey_dt; dPredator_dt];

    %捕食者-竞争模型
function dpdt = predator_competition_model(t, populations, r1, K1, alpha1, alpha2, S, beta1, h1, d1, c1, beta2, h2, d2, c2)
    N1 = populations(1); % 食饵数量
    P1 = populations(2); % 捕食者1数量
    P2 = populations(3); % 捕食者2数量

    % 食饵种群的变化
    dN1dt = r1 * N1 * (1 - N1 / K1) - alpha1 * N1 * P1 - alpha2 * N1 * P2 + S;

    % 捕食者1种群的变化
    dP1dt = beta1 * (N1 * P1) / (1 + h1 * P1) - d1 * P1 - c1 * P1 * P2;

    % 捕食者2种群的变化
    dP2dt = beta2 * (N1 * P2) / (1 + h2 * P2) - d2 * P2 - c2 * P1 * P2;

    dpdt = [dN1dt; dP1dt; dP2dt];

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)%暂停状态按钮回调函数
global isPaused isStopped
disp('pause Button pressed!');
isPaused = true;%如果当前状态是暂停状态
isStopped = false;%如果当前状态是结束状态

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)%结束按钮回调函数
    global isPaused isStopped
    disp('stop Button pressed!');
    isStopped = true;%如果当前状态是结束状态
    isPaused = false;%非暂停状态
    cla(handles.axes2);%清除所有图像
    set(handles.text_time, 'String', num2str(0));%更新时间数据
    set(handles.text_prey_data, 'String', num2str(0));
    set(handles.text_predator1_data, 'String', num2str(0));
    set(handles.text_predator2_data, 'String', num2str(0));

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
     % 获取当前的图形句柄
```

```
    figHandle = handles.figure1;

    % 弹出文件保存对话框，选择保存位置和文件名
    [fileName, filePath] = uiputfile({'*.png';'*.jpg';'*.fig';'*.pdf'}, '保存图表');

    % 如果选择了保存路径和文件名
    if fileName ~= 0
        fullFileName = fullfile(filePath, fileName);

        % 保存当前图形为PNG格式（可以选择其他格式）
        saveas(figHandle, fullFileName);

        % 如果希望提供用户反馈，可以显示保存成功的消息
        msgbox('图表已保存', '保存成功', 'help');
    end

function pushbutton_start_CreateFcn(hObject, eventdata, handles)
function axes2_CreateFcn(hObject, eventdata, handles)
function edit_initial_predator2_Callback(hObject, eventdata, handles)
function edit_initial_predator2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit_predator2_death_rate_Callback(hObject, eventdata, handles)
function edit_predator2_death_rate_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit_predator2_hunt_rate_Callback(hObject, eventdata, handles)
function edit_predator2_hunt_rate_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit_competition_coefficient_Callback(hObject, eventdata, handles)

function edit_competition_coefficient_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```
function popupmenu_model_select_Callback(hObject, eventdata, handles)

function popupmenu_model_select_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes just before test is made visible.
function test_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = test_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function edit_time_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function edit_time_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit_initial_predator1_Callback(hObject, eventdata, handles)

function edit_initial_predator1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit_initial_prey_Callback(hObject, eventdata, handles)

function edit_initial_prey_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_prey_growth_rate_Callback(hObject, eventdata, handles)

function edit_prey_growth_rate_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
271.      set(hObject,'BackgroundColor','white');
272.  end
273.
274.
275.
276.  function edit_predator1_death_rate_Callback(hObject, eventdata, handles)
277.
278.  function edit_predator1_death_rate_CreateFcn(hObject, eventdata, handles
     )
279.
280.  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontr
     olBackgroundColor'))
281.      set(hObject,'BackgroundColor','white');
282.  end
283.
284.
285.  function edit_predator1_hunt_rate_Callback(hObject, eventdata, handles)
286.
287.  function edit_predator1_hunt_rate_CreateFcn(hObject, eventdata, handles)
288.  %       See ISPC and COMPUTER.
289.  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontr
     olBackgroundColor'))
290.      set(hObject,'BackgroundColor','white');
291.  end
292.
293.  function edit_prey_provisioning_rate_Callback(hObject, eventdata, handle
     s)
294.
295.  function edit_prey_provisioning_rate_CreateFcn(hObject, eventdata, handl
     es)
296.
297.  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontr
     olBackgroundColor'))
298.      set(hObject,'BackgroundColor','white');
299.  end
300.  function text4_CreateFcn(hObject, eventdata, handles)
301.  function text2_DeleteFcn(hObject, eventdata, handles)
302.  function text27_CreateFcn(hObject, eventdata, handles)
303.
304.
```