# 第十一届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会　　网址：www.tzmcm.cn
电话：0471-4969085　　邮编：010021　　Email：service@tzmcm.cn


## 第十一届“认证杯”数学中国

## 数学建模网络挑战赛
## 承　诺　书

我们仔细阅读了第十一届“认证杯”数学中国数学建模网络挑战赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们接受相应处理结果。

我们允许数学中国网站(www.madio.net)公布论文，以供网友之间学习交流，数学中国网站以非商业目的的论文交流不需要提前取得我们的同意。

**我们的参赛队号为：1013**

**参赛队员　(签名)　：**

队员 1：

队员 2：

队员 3：

**参赛队教练员 (签名)：**

**参赛队伍组别（例如本科组）：研究生组**

地址：数学中国数学建模网络挑战赛组委会　　网址：www.tzmcm.cn
电话：0471-4969085　　邮编：010021　　Email：service@tzmcm.cn

# 第十一届“认证杯”数学中国

# 数学建模网络挑战赛
# 编 号 专 用 页

参赛队伍的参赛队号：（请各个参赛队提前填写好）：

1013

竞赛统一编号（由竞赛组委会送至评委团前编号）：

竞赛评阅编号（由竞赛评委团评阅前进行编号）：

第十一届数学中国数学建模网络挑战赛

地址：数学中国数学建模网络挑战赛组委会　　网址：www.tzmcm.cn
电话：0471-4969085　　邮编：010021　　Email：service@tzmcm.cn

# 2018 年第十一届“认证杯”数学中国数学建模网络挑战赛第一阶段论文

题　　目　基于规则反应函数的鱼群运动模型

关 键 词　自组织 鱼群建模 规则反应函数 捕食交互模型 离散化

## 摘　　要：

沙丁鱼群具有一种典型的自组织复杂运动过程，其个体间通过较为简单的规则聚集在一起，使得群体表现出协调性、智能性，大大降低了被捕食的概率。本文针对沙丁鱼群被一条海豚捕食时的情况，建立了**基于规则反应函数的沙丁鱼群和海豚捕猎交互运动模型**，并进行仿真验证。本文将建模和仿真过程大体分为三个步骤：

首先，本文建立了海域三维坐标系，借鉴粒子群算法的思想，创建随机分布的、质点化的沙丁鱼个体，建立了沙丁鱼群在无海豚威胁情况下的自组织运动模型，满足以下基本规则：**惯性、防撞、聚中、模仿及逃离**。模型中，每条规则相对应于特殊的**反应函数**，以速度来度量基于规则的反应，将不同规则的反应转化为合成速度，从而迭代出沙丁鱼在一段时间内的完整的运动轨迹。通过 Python 和 Matlab 对模型进行仿真分析和可视化，结果可得：鱼群在开始的一段时间内呈现出无明显规律的集群运动，但到某个时刻鱼群会慢慢聚集起来，呈现一定的运动规律（即产生了**自组织行为**），此时整个鱼群有较小的波动，但大致上呈现出类球体的形态，且鱼群规模越大越明显，符合实际情况。

之后，本文给出了海豚在追逐鱼群时的最优选择运动规律模型。模型中海豚满足以下规则：**惯性、捕猎**。规定捕猎规则是指海豚向沙丁鱼群的中心游动，利用规则得出未接近沙丁鱼群时的海豚运动轨迹，从而建立完整的沙丁鱼群与海豚的捕猎交互运动模型。此时，沙丁鱼的逃离规则会被显著体现。针对海豚的捕食过程，设计海豚的**必定捕食距离**以及沙丁鱼个体的**消亡**过程。通过仿真结果可得：①海豚接触鱼群时，鱼群形态有明显的**“炸裂式”反应**，从接触点处向整个鱼群传播。在这种形态变化的过程中，沙丁鱼个体仍旧表现出一定的聚集效应，同时尽量远离海豚位置。②若海豚在一定时间后离开，鱼群能够慢慢恢复自组织形态，否则鱼群呈现出反复的**震荡形态**；同时，单只海豚在持续性捕猎时，其捕猎效率随时间的增长而减小，与客观实际比较吻合。

本文的创新点在于通过建立规则反应函数来表示沙丁鱼群交互运动模型的基本规则，通过模型仿真实现了沙丁鱼群在遇到一条海豚捕食时的运动规律。不足之处在于未考虑海豚的其他捕食方案，在一些细节上做了简化，一定程度上影响了模型精度。在进一步工作中将会考虑沙丁鱼群如何应对海豚不同的捕食方案，优化反应函数，扩展个体差异性。

**参赛队号：** 1013

**所选题目：** A **题**

参赛密码 _____

（由组委会填写）

# 英文摘要（选填）

Sardines have a typical self-organized complex motion process, and their individuals gather together through simple rules. It makes the population show coordination and intelligence, which greatly reduces the probability of prey. In order to solve the problem of sardines being preyed by a dolphin, a model of the interaction between sardines and dolphins based on regular response function is established， then verified by simulation. In this paper, the modeling and simulation process is divided into three steps:

Firstly, the 3D coordinate system of sea area is established. Based on the idea of particle swarm optimization, the stochastic distribution and particle size of sardines are created. A self-organizing movement model of sardines without dolphin threat is established, which satisfies the following basic rules: inertia, collision prevention, clustering, imitation and escape. In the model, each rule corresponds to a special response function. The response based on the rule is measured by the velocity, and the reaction of different rules is transformed into the synthesis speed, then the complete trajectory of the sardines is iterated over a period of time. Through the simulation and visualization of the model by Python and Matlab, it can be seen that the fish herd shows an irregular cluster motion for a period of time, but at some point, the fish flock slowly, showing a certain pattern of movement (that is, self-organizing behavior). At this point, the whole fish stock has a small fluctuation, but generally presents the shape of a spherical body, and the larger the fish population, the more obvious it is.

After that, this paper gives the model of the optimal choice movement of dolphin in pursuit of fish herd. Dolphins in the model meet the following rules: inertia, hunting. The rule of hunting refers to the dolphin swimming toward the center of sardine fish group. The movement track of dolphin which is not close to sardine group is obtained by using the rule, and a complete interactive model of hunting between sardine group and dolphin is established. At this point, the escape rules of sardines are clearly reflected.Aiming at the predation process of dolphin, the paper designs the distance of dolphin's predation and the process of sardine's individual extinction.The simulation results show that when the dolphin contact with the fish herd, the fish form has an obvious "burst" reaction, which propagates from the contact point to the whole fish herd. In the course of this morphological change, sardines still exhibit a certain aggregation effect, while keeping away from dolphin position as far as possible. If the dolphin leaves after a certain period of time, the fish flocks can slowly return to self-organized form, otherwise, the fish herd presents a repeatedly oscillating form. At the same time, the hunting efficiency of a single dolphin decreases with the increase of time when the dolphin continues to hunt. It coincides with the objective reality.

The innovation of this paper is to establish regular response functions to represent the basic rules of the interactive movement model of sardines, and to realize the movement law of sardines when they encounter a dolphin prey through model simulation. The shortcoming lies in the fact that other predation schemes of dolphins are not considered, and some details have been simplified which affects the precision of the model. In the further work, the different predation schemes between sardines and dolphin will considered， and optimizing response function to expand individual difference.

# 目录

一、 问题重述 2

二、模型假设及符号说明 2

2.1 模型假设 2

2.2 符号说明 2

三、问题分析 3

四、建模求解 4

4.1 建模基本思路 4

4.2 沙丁鱼群自组织运动模型 5

4.2.1 防撞规则 6

4.2.2 聚中规则 7

4.2.3 模仿规则 8

4.2.4 逃离规则 8

4.3 未接近沙丁鱼群的海豚运动建模 9

4.4 沙丁鱼与海豚捕猎交互模型 10

4.5 模型分析与仿真 11

4.5.1 模型的结果预测 11

4.5.2 模型仿真 11

4.5.3 模型结果分析 15

五、模型的评价与推广 15

5.1 模型的优点与缺点 15

5.2 模型的改进与推广 16

参考文献 17

附录 18

# 一、问题重述

沙丁鱼以聚成大群的方式来对抗海豚的捕食。由于水下光线很暗，所以在距离较远时，海豚只能使用回声定位方法来判断鱼群的整体位置，难以分辨每个个体。鱼群的行动是有协调性的，在没有外部威胁或障碍物时，鱼群常常会聚成接近球形的形态。而当海豚接触甚至冲进鱼群，鱼群则会进行协同的躲避，所以不易在大鱼群中追踪一个目标。沙丁鱼的这种群体行为降低了其被海豚捕食的概率。

请你建立合理的数学模型来描述沙丁鱼群在遇到一条海豚捕食时的运动规律。

# 二、模型假设及符号说明

## 2.1 模型假设

**在整个求解过程中，假设：**

1）鱼群和海豚都活动在足够大的水域中，不受地形和海平面的影响；

2）所有的沙丁鱼个体具有相同的物理特性（包括运动能力）；

3）单个沙丁鱼的感知范围为以自己为中心，半径为 R 的球域。考虑到沙丁鱼体型较小，可将其视为质点处理；

4）每条沙丁鱼都能与处在其感知域中的其他任意一条沙丁鱼进行有效的信息交互，以传递威胁信息等；

5）暂不考虑某些沙丁鱼个体的领导能力、饥饿、疲劳等因素的影响；

6）考虑到声波传递速度，假设海豚可以时刻感知到整个鱼群；

7）沙丁鱼和海豚的运动能力都是在一定的范围内可调的，即各自有最大运动速度和最低运动速度。

## 2.2 符号说明

| | |
|---|---|
| $m(t)$ | t 时刻沙丁鱼数量 |
| $m_i(t)$ | t 时刻处在第$i$ 条鱼感知域 $R_i$ 内的鱼数量（包含自己） |
| $A_i, A_j$ | 沙丁鱼个体 i，j |
| $P_i(t)$ | 第$i$ 条鱼在 t 时刻的位置 |
| $v_i(t)$ | 第$i$ 条鱼在 t 时刻的速度 |
| $R_i$ | 第$i$ 条鱼的感知范围 |
| $v_i^1(t)$ | 第$i$ 条鱼在 t 时刻基于防撞规则的单一速度响应（矢量） |

| | |
|---|---|
| $v_i^2(t)$ | 第$i$条鱼在t时刻基于聚中规则的单一速度响应（矢量） |
| $v_i^3(t)$ | 第$i$条鱼在t时刻基于模仿规则的单一速度响应（矢量） |
| $v_i^4(t)$ | 第$i$条鱼在t时刻基于逃离规则的单一速度响应（矢量） |
| $v_B^c(t)$ | 海豚在t时刻基于捕猎规则的单一速度响应（矢量） |
| $k_1,k_2,k_3,k_4,k_c$ | 规则相关系数 |
| $P_B(t)$ | 海豚在t时刻的位置 |
| $v_B(t)$ | 海豚在t时刻的速度 |
| $d_{safe}$ | 沙丁鱼间最小舒适距离 |
| $d_{gather}$ | 沙丁鱼间最大舒适距离 |
| $\Delta t$ | 运动时间间隔 |
| $d_{catch}$ | 必定捕杀距离 |

## 三、问题分析

沙丁鱼的集群行为是典型的自组织过程。鱼群中个体间按照一定的规则进行信息交互，使得整个鱼群产生协调性的、智能化的集群运动。**首先**，建立坐标系，借鉴粒子群算法的思想，创建随机分布的、质点化的沙丁鱼个体，给出个体当前运动状态的描述。分析这些鱼群个体间的交互规则，解析并在模型中实现规则。个体能够按照交互运动规则，将下一步的运动状态表示出来。鱼群的自组织过程可以借助计算机仿真来进行模拟，以检查模型的准确性。

**其次**，在描述沙丁鱼群无海豚威胁时的集群行为后，添加海豚的个体，以类似于沙丁鱼个体的方法来描述其运动。海豚除了自身的运动规则之外，还会和沙丁鱼产生捕猎的交互。海豚的位置将在远离沙丁鱼群的一定范围内出现，通过仿真来模拟整个沙丁鱼群与海豚的运动规律，将整个运动过程可视化。

**整个问题的难点在于沙丁鱼个体间、沙丁鱼与海豚间的交互规则的分析和实现**。一般而言，沙丁鱼间的交互效应众多因素的影响。本文中主要探讨包括防撞、聚中、模仿、逃离和惯性这几个因素的影响，对于洋流方向、食物分布、环境等其他将不予考虑，从而简化问题。

沙丁鱼的感知能力是有限度的，假定感知域为球域，中心为自身。对鱼群中的某一个体而言，它会受所感知到的其他个体的影响。本文认为影响值与其距离有关系，且主要来自于防撞、聚中、模仿、逃离四个因素。为方便处理，采用速度来描述影响作用，

将各条规则所产生的速度反应矢量合成为最终速度。下一时刻的运动状态将取决于当前时刻的运动状态和当前时刻的速度。

严格来说，海豚捕猎沙丁鱼的过程是极为复杂的，一次捕杀成功与否取决于双方的位置、速度、生理状态乃至时机。为了使问题清晰化，本文认为海豚应当具有对处在其必定捕杀距离内的沙丁鱼一击必杀的能力，并且其猎杀过程的时间与整个追逐沙丁鱼群的时间相比可忽略。同时，海豚能够利用回声定位对沙丁鱼群保持时刻的整体感知，其感知所耗费的时滞几乎不计。

考虑到物理实际，鱼和海豚的速度都将是有限制的。

# 四、建模求解

鱼群是典型的具有自组织特性的复杂系统。一定数量的个体按照某些简单规则或方式调节自身状态，会产生出群体具有而部分或部分总和所不具有的属性、特征、行为、功能等，也就是说个体间通过相互作用产生出某种群体行为。鱼群的自组织行为是生物进化的结晶，也是生物圈中的普遍现象。研究它的演化规律，开展仿生学研究，对于生产生活具有重要意义。其中，以数学模型来模拟和探索生物集群自组织运动的内在规律和相应机制是重点内容[1,2]。

## 4.1 建模基本思路

第一阶段问题是需要建立合理的数学模型来描述沙丁鱼群在遇到一条海豚捕食时的运动规律，本文分两步来解决。

首先是处理题中沙丁鱼群本身的描述问题。为简化问题，我们假设了沙丁鱼和海豚的运动范围足够大，不受限于地形或者海平面的影响。沙丁鱼的体长不会超过 0.3 米，因此在广阔的大海中我们可以将其进行质点化处理。在海中建立坐标系，对于每个沙丁鱼个体，其最初位置确定（在一定范围内随机分布，初速度为 0），其下一时刻的位置取决于当前位置与本文所制定的交互规则。这些规则通过速度来体现。通过查阅资料，鱼群运动的规则主要有防撞、聚中、模仿、逃离及惯性等[2,3,4]。因此，通过不断迭代，鱼群的在无海豚威胁情况下的自组织过程将得以完全描述。

在此基础上，添加海豚的模型。海豚的初始位置确定（随机出现在远离鱼群的适当位置），其运动规则主要是惯性和捕食规则（海豚只能确定鱼群整体位置，难以分辨单独个体），同样的利用速度来更新后续时刻的运动状态。

综上所诉，整个沙丁鱼群与海豚的运动规律即可通过数学模型来描述，上述过程可用图 4-1 表示：

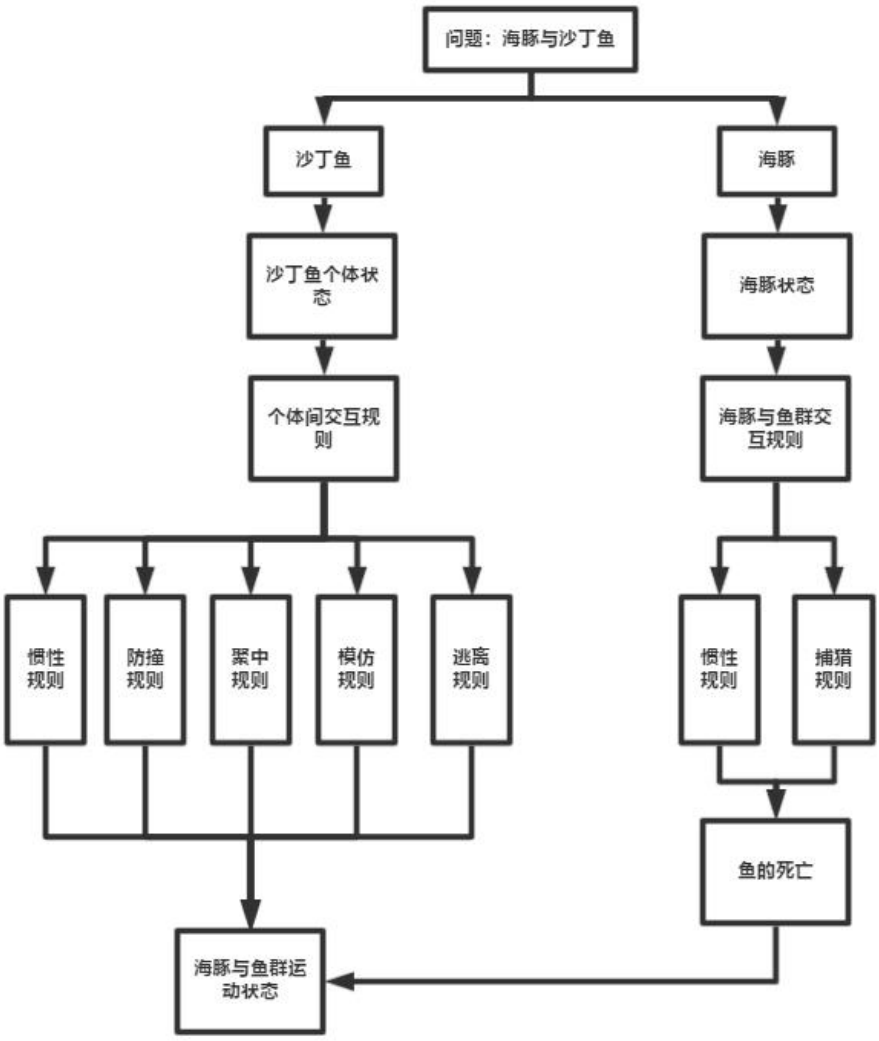


图 4.1 建模基本思路

## 4.2 沙丁鱼群自组织运动模型

在海中建立三维笛卡尔坐标系，按条件随机生成一定数量的鱼群个体。某时刻海豚尚未威胁到鱼群，鱼群中的某条沙丁鱼个体 $A_i$ 的位置可表示为点 $P_i(t)=[x_i(t),y_i(t),z_i(t)]$，速度可表示为向量 $v_i(t)$，则鱼 $A_i$ 为元素 $[P_i(t),v_i(t)]$。因此，整个鱼群的状态可表示为集合：

$$\Omega=\{[P_1,v_1],[P_2,v_2],[P_3,v_3],...,[P_{m_i(t)},v_{m_i(t)}]\} \quad \text{（式 1）}$$

式中 $m_i(t)$ 为鱼群总数。

当鱼 $A_i$ 死亡时，令对应的元素 $[P_{m_i(t)},v_{m_i(t)}]=\varnothing$，$m_i(t)$ 相应减少。在海中，沙丁鱼只会对处在它感知域范围内的个体产生交互影响，由于沙丁鱼已质点化，故其感知域为以自身为中心，半径为 $R$ 的球域，即为 $R_i$。而鱼群在运动过程中要满足一定的交互规则，这些规则**通过对应的单一速度反应矢量来体现**。通过查阅资料，鱼群运动的规则主要有防撞、聚中、模仿、感知逃离及惯性等。因此，沙丁鱼 $A_i$ 在下一时刻的运动速度可表示为：

$$v_i(t+\Delta t)=v_i(t)+v_i^1(t)+v_i^2(t)+v_i^3(t)+v_i^4(t) \quad \text{（式 2）}$$

式中，$v_i(t)$ 表示鱼的惯性；$v_i^1(t)$ 表示鱼基于防撞规则的速度反应；$v_i^2(t)$ 表示鱼基于聚中规则的速度反应；$v_i^3(t)$ 表示鱼基于模仿规则的速度反应；$v_i^4(t)$ 表示鱼基于逃离规则的速度反应。应当注意，沙丁鱼的实际运动能力是有一定范围限制的，即 $v_{min}\le v_i(t)\le v_{max}$，$\forall i,t$。

因此，沙丁鱼 $A_i$ 在下一时刻的位置可表示为：

$$P_i(t+\Delta t)=\begin{cases}x_i(t)+\int_t^{t+\Delta t}v_{ix}(t+\Delta t)d\Delta t\\ y_i(t)+\int_t^{t+\Delta t}v_{iy}(t+\Delta t)d\Delta t\\ z_i(t)+\int_t^{t+\Delta t}v_{iz}(t+\Delta t)d\Delta t\end{cases}=\begin{cases}x_i(t)+\int_t^{t+\Delta t}coscosv_i(t+\Delta t)d\Delta t\\ y_i(t)+\int_t^{t+\Delta t}sincosv_i(t+\Delta t)d\Delta t\\ z_i(t)+\int_t^{t+\Delta t}sinv_i(t+\Delta t)d\Delta t\end{cases} \quad \text{（式 3）}$$

因此，在无海豚威胁情况下，整个鱼群的自组织过程的状态模型即为：

$$\Omega=\{[P_1,v_1],[P_2,v_2],[P_3,v_3],...,[P_{m_i(t)},v_{m_i(t)}]\}$$

下面，我们将分别讨论沙丁鱼间的几条交互规则。

### 4.2.1 防撞规则

沙丁鱼在鱼群中游动时，相互之间要保持一定的距离，记最短安全距离为 $d_{safe}$。一般来说，对于距离越近的个体，沙丁鱼基于防撞规则而产生的应激反应越强烈，但在一定距离之外又几乎毫无反应，即沙丁鱼的反应相对于距离来说是非线性的，如图 4.2 所示。

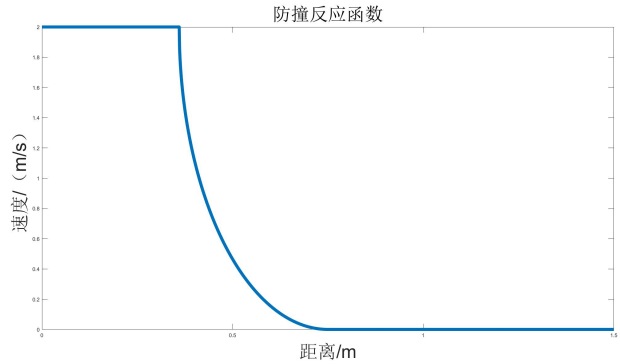


图 4.2 沙丁鱼的基于防撞规则反应相对于距离的关系

沙丁鱼 $A_i$ 对沙丁鱼 $A_j$ 的基于防撞规则的速度反应描述为：

$$v_{ij}^{1}(t)=\begin{cases} v_{i\max}\cdot e_{ij}^{1}, \left|P_i(t)-P_j(t)\right|<d_{safe}; \\ v_{i\max}\cdot e_{ij}^{1}\cdot\left(1-\sqrt{1-\left(\frac{\left|P_i(t)-P_j(t)\right|-d_{safe}}{d_{gather}-d_{safe}}\right)^2}\right), d_{safe}\le\left|P_i(t)-P_j(t)\right|<d_{gather}; \\ 0, \left|P_i(t)-P_j(t)\right|\ge d_{gather} \end{cases}$$

式中，$d_{safe}$ 表示沙丁鱼的最小舒适距离，沙丁鱼间距小于这个值时表现出明显的互相排斥、以防止碰撞的倾向；$d_{gather}$ 表示最大舒适距离，超越这个距离时沙丁鱼对防撞规则的反应大幅度降低。

基于此，将沙丁鱼 $A_i$ 基于防撞规则的速度反应描述表示为：

$$v_i^1(t)=k_1 e_{ij}^1\cdot\sum_{j\in R_i}v_{ij}^1(t) \qquad \text{(式 4)}$$

式中，$e_{ij}^1=\frac{P_i(t)-P_j(t)}{\left|P_i(t)-P_j(t)\right|}$，表示该速度是由鱼 $A_i$ 感知域 $R_i$ 内的鱼 $A_j$ 指向鱼 $A_i$ 自身的。$k_1$ 表示防撞相关系数，一般取 1。

### 4.2.2 聚中规则

沙丁鱼群具有简单的趋向于群体中心的倾向。鱼组成鱼群，通过协同躲避，在一定程度上干扰了掠食者对目标的选择，与单独个体相比，鱼群的这种群体行为提高了生存率。同时，群体性的鱼类也更加容易发现事物或捕食者，以及更高的成功繁衍率。沙丁鱼群的数量往往可以聚集到一个极大的数字。

类似于防撞规则，沙丁鱼对于聚中规则的反应相对于距离来说也是非线性的。如图 4.3 所示。

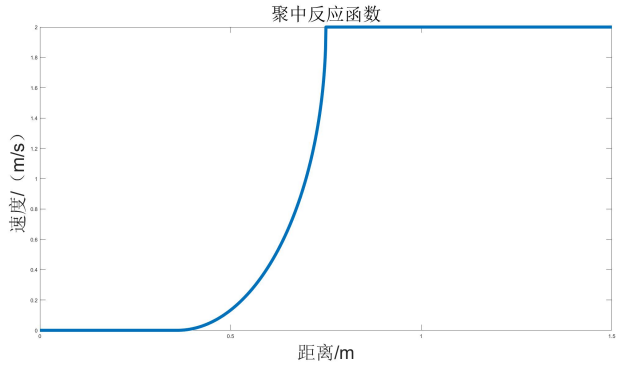


图 4.3 沙丁鱼的基于聚中规则反应相对于距离的关系

沙丁鱼 $A_i$ 对沙丁鱼 $A_j$ 的基于聚中规则的速度反应描述为：

$$v_{ij}^{2}(t)=\begin{cases} 0, \left|P_i(t)-P_j(t)\right|<d_{safe}; \\ v_{i\max}\cdot e_{ij}^{1}\cdot\left(1-\sqrt{1-\left(\frac{\left|P_i(t)-P_j(t)\right|-d_{safe}}{d_{gather}-d_{safe}}\right)^2}\right), d_{safe}\le\left|P_i(t)-P_j(t)\right|<d_{gather}; \\ v_{i\max}\cdot e_{ij}^{1}, \left|P_i(t)-P_j(t)\right|\ge d_{gather} \end{cases}$$

可以看出，表示沙丁鱼的最小舒适距离，沙丁鱼间距大于最大舒适距离 $d_{gather}$ 时表现出明显的互相吸引、聚中效应；间距小于最小舒适距离 $d_{safe}$ 时沙丁鱼对聚中规则几乎无反应。

基于此，将沙丁鱼 $A_i$ 基于聚中规则的速度反应描述表示为：

$$v_i^2(t) = k_2 e_{ij}^2 \cdot \sum_{j \in R_i} v_{ij}^2(t) \qquad \text{(式 5)}$$

式中，$e_{ij}^2 = \dfrac{P_j(t) - P_i(t)}{\left| P_i(t) - P_j(t) \right|}$，表示该速度是由鱼 $A_i$ 指向感知域 $R_i$ 内的鱼 $A_j$ 的。$k_2$ 表示聚中相关系数，一般取 1。通过这种加权的形式的累加，能够使得最后求出的速度描述指向感知域内的反应聚中规则关系的中心点。

### 4.2.3 模仿规则

沙丁鱼通过视觉和侧线来感知其他个体，在鱼群中会明显性的跟随前方和周围的个体，以保持运动的一致性。这种一致性使得个体仅需要较低的速度来保证鱼群整体的较高的稳定性，降低鱼群产生碰撞等的几率。

沙丁鱼对于在其感知域内的其他个体的模仿可表示为其他个体的速度在它身上的映射合成量。

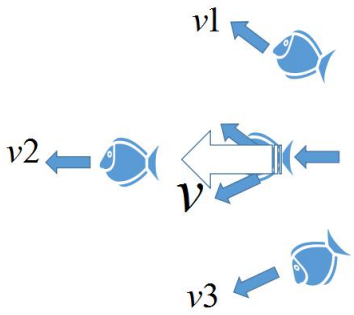


图 4.4 沙丁鱼的模仿规则

对沙丁鱼 $A_i$ 而言，它基于模仿规则的速度反应可表示为：

$$v_i^3(t) = \frac{k_3}{m_i(t)} \sum_{j \in R_i} v_j(t) \qquad \text{(式 6)}$$

式中，$k_3$ 表示模仿相关系数，一般取 1.

### 4.2.4 逃离规则

沙丁鱼是典型的小型小型植食性鱼类，天然处在肉食性的海豚食物链中。当每个沙丁鱼在感知范围内发现海豚的踪迹时，最佳的逃离方向是与海豚追赶的方向相同。沙丁鱼距离海豚越近时，逃离的急迫性越高，即沙丁鱼对于逃离规则的速度反应相对于距离来说也是非线性的。如图 4.5 所示。

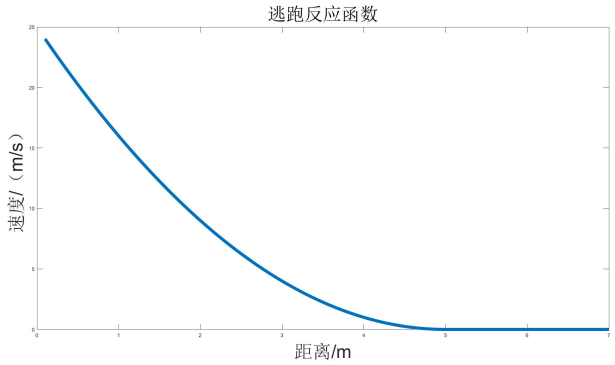


图 4.5 沙丁鱼的基于逃离规则反应相对于距离的关系

上述关系可表示为：

$$v_i^4(t)=\begin{cases}\dfrac{k_4}{|P_i(t)-P_B(t)|}\cdot e_i^4, |P_i(t)-P_B(t)|\le R_i\\ 0, |P_i(t)-P_B(t)|>R_i\end{cases} \quad (式\ 7)$$

式中，$e_i^4=\dfrac{P_i(t)-P_B(t)}{|P_i(t)-P_B(t)|}$ 表示该速度是由鱼 $A_i$ 感知域内的海豚指向鱼 $A_i$ 自身的。$k_4$ 表示逃离相关系数，一般取 1。

### 4.3 未接近沙丁鱼群的海豚运动建模

对于海豚的描述类似于沙丁鱼个体，将其质点化，记海豚的位置可表示为：$P_B(t)=[x_B(t),y_B(t),z_B(t)]$，速度可表示为$v_B(t)$，则海豚状态为集合：

$$\Omega_{\mathrm{B}}=\{[P_B(t),v_B(t)]\} \quad (式\ 8)$$

在海中，海豚能利用声波定位十几公里外的鱼群。由于在海中声波传递速度较快，在接近鱼群时海豚本身的感知域可视为覆盖全鱼群。海豚的定位能判断鱼群的整体位置，但是难以分辨每个个体，这影响了它的捕猎策略。因此，海豚在下一时刻的运动速度可表示为：

$$v_B(t+\Delta t)=v_B(t)+v_B^c(t) \quad (式\ 9)$$

式中，$v_B(t)$表示海豚的惯性，$v_B^c(t)$表示海豚基于捕猎规则的速度反应。

同样的，海豚的实际运动能力是有一定范围限制的，即 $v_{Bmin}\le v_B(t)\le v_{Bmax},\ \forall t$。

因此，海豚在下一时刻的位置可表示为：

$$P_B(t+\Delta t)=\begin{cases}x_B(t)+\int_t^{t+\Delta t}v_{Bx}(t+\Delta t)d\Delta t\\ y_B(t)+\int_t^{t+\Delta t}v_{By}(t+\Delta t)d\Delta t\\ z_B(t)+\int_t^{t+\Delta t}v_{Bz}(t+\Delta t)d\Delta t\end{cases}=\begin{cases}x_B(t)+\int_t^{t+\Delta t}coscosv_B(t+\Delta t)d\Delta t\\ y_B(t)+\int_t^{t+\Delta t}sincosv_B(t+\Delta t)d\Delta t\\ z_B(t)+\int_t^{t+\Delta t}sinv_B(t+\Delta t)d\Delta t\end{cases} \quad (式\ 10)$$

下面，讨论下海豚的捕猎策略以及由此引发的与沙丁鱼群之间的交互。

### 4.4 沙丁鱼与海豚捕猎交互模型

由于海豚只能针对整个鱼群来决定捕猎策略，它的捕猎规则是倾向于直接追寻鱼群的中心（质点化时即几何中心），其速度反应可表示为：

$$v_{catch}(t)=\frac{k_c}{m(t)}\sum_{i=1}^{m(t)}(P_i(t)-P_B(t)) \qquad (式\ 11)$$

式中，$k_c$ 为捕猎相关系数，一般取 1.

对于海豚而言，假定当它距离沙丁鱼一定距离内，记为捕杀距离 $d_{catch}$ 时，它将必定捕杀该条沙丁鱼。因此，对于沙丁鱼 $A_i$，当它的位置处在捕杀距离内时，$A_i$ 死亡，即 $[P_{m_i(t)},v_{m_i(t)}]=\varnothing$，同时鱼群总数量 $m(t)$ 相应减少。上述关系可表示为：

$$[P_{m_i(t)},v_{m_i(t)}]=\varnothing,\ \exists|P_i(t)-P_B(t)|<d_{catch} \qquad (式\ 12)$$

综上所述，建立完整的**海豚与沙丁鱼捕猎交互模型**为：

$$\begin{cases}\Omega=\{[P_1,v_1],[P_2,v_2],[P_3,v_3],...,[P_{m_i(t)},v_{m_i(t)}]\}\\ \Omega_B=\{[P_B,v_B]\}\end{cases}$$

s.t.：

$$v_i(t+\Delta t)=v_i(t)+v_i^1(t)+v_i^2(t)+v_i^3(t)+v_i^4(t),\quad v_{min}\le v_i(t)\le v_{max},\quad \forall i,t$$

$$P_i(t+\Delta t)=\begin{cases}x_i(t)+\int_t^{t+\Delta t}v_{ix}(t+\Delta t)d\Delta t\\ y_i(t)+\int_t^{t+\Delta t}v_{iy}(t+\Delta t)d\Delta t\\ z_i(t)+\int_t^{t+\Delta t}v_{iz}(t+\Delta t)d\Delta t\end{cases}=\begin{cases}x_i(t)+\int_t^{t+\Delta t}coscosv_i(t+\Delta t)d\Delta t\\ y_i(t)+\int_t^{t+\Delta t}sincosv_i(t+\Delta t)d\Delta t\\ z_i(t)+\int_t^{t+\Delta t}sinv_i(t+\Delta t)d\Delta t\end{cases}$$

$$v_i^1(t)=k_1e_{ij}^1\cdot\sum_{j\in R_i}v_{ij}^1(t)\qquad e_{ij}^1=\frac{P_i(t)-P_j(t)}{|P_i(t)-P_j(t)|}$$

，

$$v_i^2(t)=k_2e_{ij}^2\cdot\sum_{j\in R_i}v_{ij}^2(t)\qquad e_{ij}^2=\frac{P_j(t)-P_i(t)}{|P_i(t)-P_j(t)|}$$

，

$$v_i^3(t)=\frac{k_3}{m_i(t)}\sum_{j\in R_i}v_j(t)$$

$$v_i^4(t)=\begin{cases}\frac{k_4}{|P_i(t)-P_B(t)|}\cdot e_i^4,|P_i(t)-P_B(t)|\le R_i\\ 0,|P_i(t)-P_B(t)|>R_i\end{cases}\qquad e_i^4=\frac{P_i(t)-P_B(t)}{|P_i(t)-P_B(t)|}$$

，

$$\Omega_B=\{[P_B(t),v_B(t)]\}$$

$$v_B(t+\Delta t)=v_B(t)+v_B^c(t),\quad v_{Bmin}\le v_B(t)\le v_{Bmax},\quad \forall t$$

$$P_B(t+\Delta t)=\begin{cases}x_B(t)+\int_t^{t+\Delta t}v_{Bx}(t+\Delta t)d\Delta t\\ y_B(t)+\int_t^{t+\Delta t}v_{By}(t+\Delta t)d\Delta t\\ z_B(t)+\int_t^{t+\Delta t}v_{Bz}(t+\Delta t)d\Delta t\end{cases}=\begin{cases}x_B(t)+\int_t^{t+\Delta t}coscosv_B(t+\Delta t)d\Delta t\\ y_B(t)+\int_t^{t+\Delta t}sincosv_B(t+\Delta t)d\Delta t\\ z_B(t)+\int_t^{t+\Delta t}sinv_B(t+\Delta t)d\Delta t\end{cases}$$

$$v_{catch}(t)=\frac{k_c}{m(t)}\sum_{i=1}^{m(t)}(P_i(t)-P_B(t))$$

$$[P_{m_i(t)},v_{m_i(t)}]=\varnothing,\ \exists|P_i(t)-P_B(t)|<d_{catch}$$

### 4.5 模型分析与仿真

#### 4.5.1 模型的结果预测

根据经验，在没有海豚威胁时，鱼群自身会根据交互规则进行自组织运动，最终会形成较为稳定的类球形的形态。

当有海豚接近鱼群时，最外侧最接近海豚的鱼群将会率先产生较大的逃离反应，进而由于相互间的交互规则，呈现出以接触点为中心的“炸裂式”鱼群分离现象，其程度取决于沙丁鱼与海豚的运动能力差。若海豚的猎杀有上限，即海豚在猎杀到一定数量的沙丁鱼后离开，剩余的沙丁鱼将仍有机会重新进行自组织，其几率取决于存货率。

但是，若海豚的猎杀是不受限的，那么整个沙丁鱼群的形态将发生反复性的震荡状态，呈现个体逃离带来的不规则变化。当存活率低于一定阈值，整个鱼群将丧失再次的自组织能力。

#### 4.5.2 模型仿真

根据模型可知，模型中包含了较多数量的积分操作，执行起来极为复杂。为简化过程，需要对其进行线性化。在求解过程中以下面的式子

$$P_i(t+\Delta t)=\begin{cases}x_i(t)+v_{ix}(t+\Delta t)\cdot\Delta t\\ y_i(t)+v_{iy}(t+\Delta t)\cdot\Delta t\\ z_i(t)+v_{iz}(t+\Delta t)\cdot\Delta t\end{cases}=\begin{cases}x_i(t)+\text{coscos}v_i(t+\Delta t)\cdot\Delta t\\ y_i(t)+\text{sincos}v_i(t+\Delta t)\cdot\Delta t\\ z_i(t)+\text{sin}v_i(t+\Delta t)\cdot\Delta t\end{cases}\quad\text{(式 12)}$$

$$P_B(t+\Delta t)=\begin{cases}x_B(t)+v_{Bx}(t+\Delta t)\cdot\Delta t\\ y_B(t)+v_{By}(t+\Delta t)\cdot\Delta t\\ z_B(t)+v_{Bz}(t+\Delta t)\cdot\Delta t\end{cases}=\begin{cases}x_B(t)+coscosv_B(t+\Delta t)\cdot\Delta t\\ y_B(t)+sincosv_B(t+\Delta t)\cdot\Delta t\\ z_B(t)+sinv_B(t+\Delta t)\cdot\Delta t\end{cases}\quad\text{(式 13)}$$

来分别近似拟合（式 3）和（式 10）。令 $\Delta t=1$，则可根据两式不断迭代出沙丁鱼与海豚的运动状态。这种线性化的原理在于，沙丁鱼和海豚在针对外界的刺激（遵守交互规则）下做出相应的反应动作是有一定的反应时间的，在反应时间内会根据惯性继续前进；反应时间 $\Delta t$ 相对于整个鱼群的运动时间极小，因此可以用这种近似的线性过程来拟合反应动作本身。由于生物体物理限制，沙丁鱼和海豚的运动轨迹都不可能有极大导数。

本文中使用 Python 进行编程仿真，并将结果导出至 Matlab 来图形展示。模型初始运行时，鱼群所有个体在满足最小碰撞距离 $d_{safe}$ 为前提下随机产生，每个个体具有各自的状态。取 $\Delta t$ 为单位时间 1，感知域半径 R=5（为简化处理，忽略单位），沙丁鱼最大

运动速度 $v_{\max}$=2/s,海豚最大运动速度 $v_{B\max}$=10/s,初始鱼群数量为 250，$d_{safe}$ 为 0.36，$d_{gather}$ 为 0.75，其他各量取合适值（见附录程序）。随着时间的变化，模型中的个体间相互作用，使得整个系统继续往前运行。

**当无海豚接近鱼群时，整个模型的仿真结果如下：**

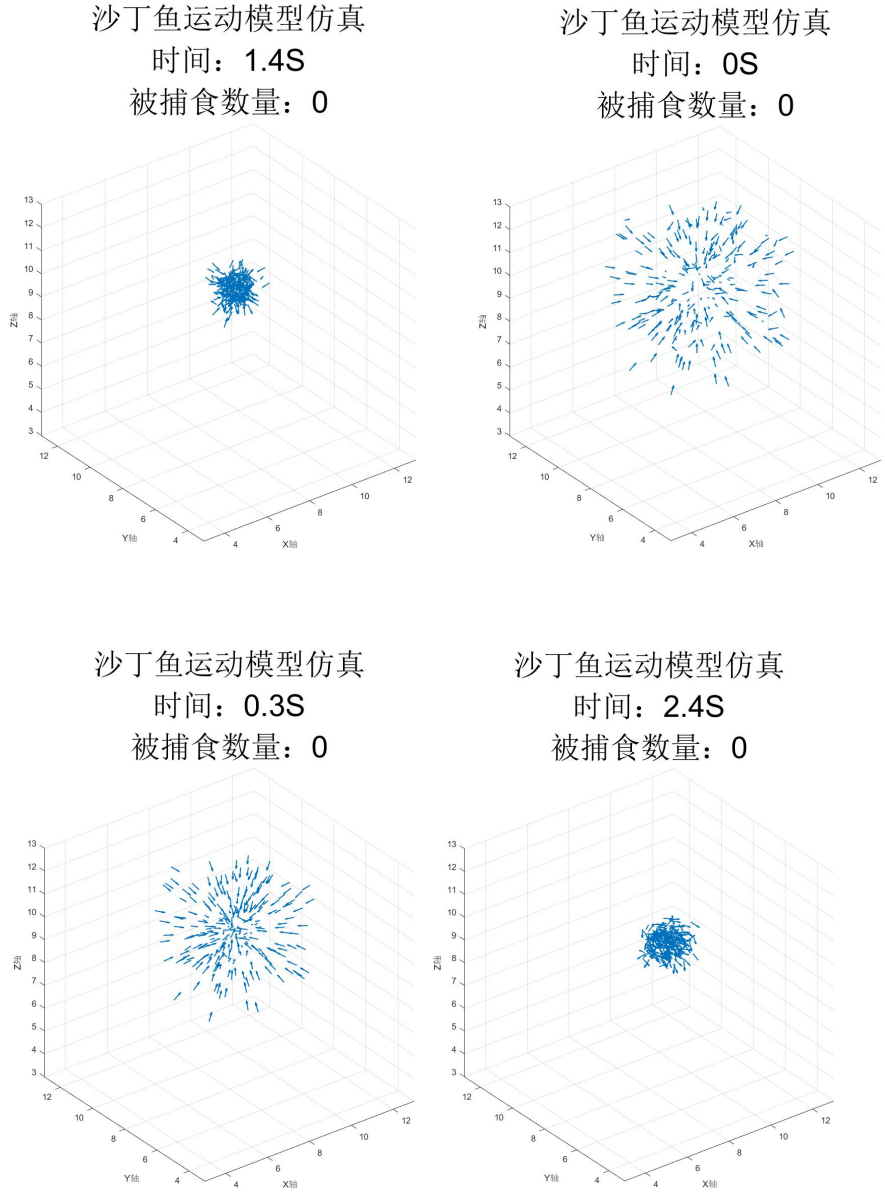


图 4.6 无海豚时鱼群的自组织行为

上图详细的反映了沙丁鱼群在无海豚刺激的情况下，由初始状态处出发的自组织过程。

**当海豚（图中黄色箭头）接近鱼群，并在猎杀到一定数量沙丁鱼后离开，整个模型的仿真结果如下：**

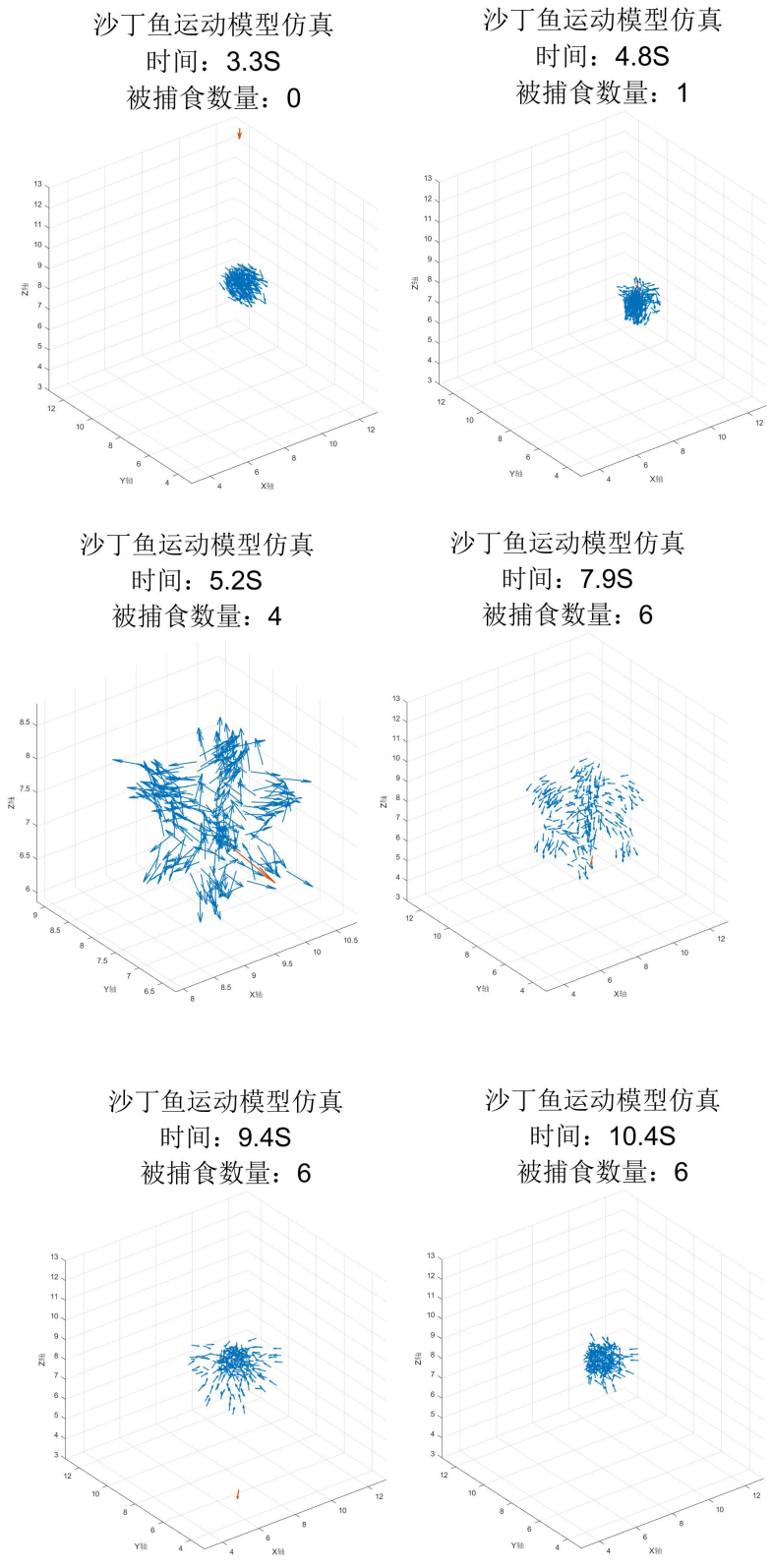


图 4.7 海豚接近鱼群、猎杀并离开

应当注意，在图 4.7 中，当 t=5.2s 时，图像展示的是放大的细节，图中鱼群初始总数为 250。可以看出沙丁鱼群对于海豚有明显的、扩散式的反应，并在海豚离开后能够恢复自组织过程。整个过程中，鱼群的离散程度变化情况如下：

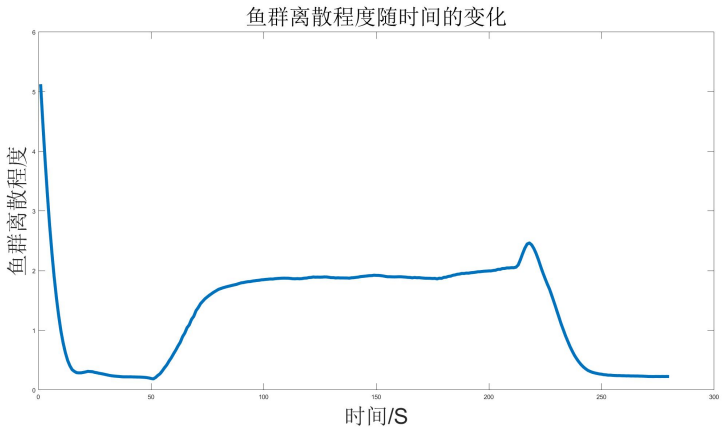


图 4.8 鱼群离散程度的变化

一开始，鱼群的离散程度较大，通过短时间的自组织运动，离散程度显著减小且稳定，当海豚接近时，鱼群就会开始逃逸，离散程度又会变大，但鱼群不会完全散开，当海豚离开后，鱼群的离散值又恢复到最低值。

**当海豚（图中黄色箭头）反复在沙丁鱼群中捕猎时，整个模型的仿真结果如下：**

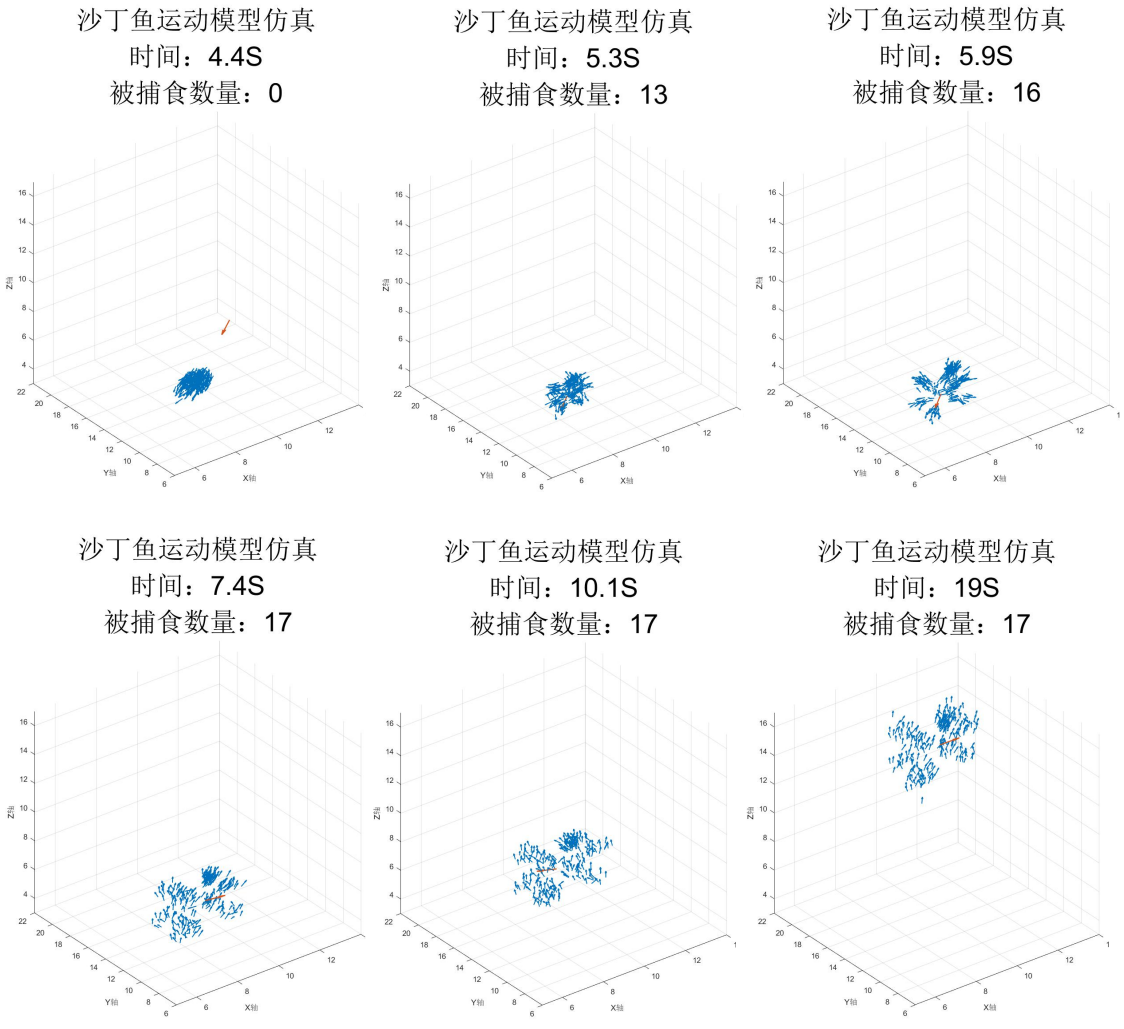


图 4.9 海豚反复捕猎鱼群

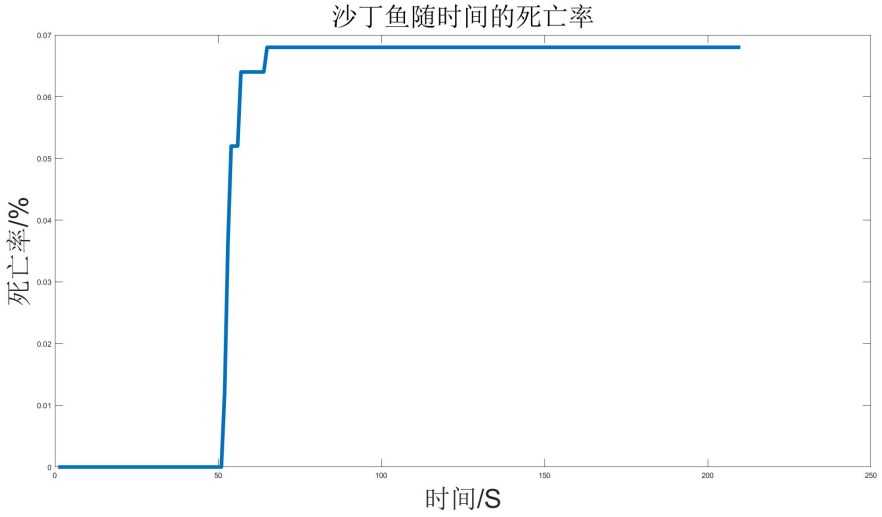


图 4.10 沙丁鱼的死亡率

明显可见，海豚不离开，整个沙丁鱼群仍将处于不稳定的震荡形态；海豚在最初的一段时间内有捕获沙丁鱼，但是随着时间的推移，沙丁鱼的死亡率会逐渐上升到 0.07%且基本维持在一个较稳定的水平（见图 4.10），也就是说单只海豚的捕猎效率随时间的增长而减小。

### 4.5.3 模型结果分析

1）鱼群在最开始的一段时间内呈现出无明显规律的集群运动，但到某个时刻整个鱼群在个体间相互作用下慢慢聚集起来，呈现一定的运动规律（即产生了自组织行为），此时整个鱼群有较小的波动，但大致上呈现出类球体的形态，且鱼群规模越大越明显。

2）海豚接触鱼群时，鱼群形态有明显的“炸裂式”反应，从接触点处向整个鱼群传播。在这种形态变化的过程中，沙丁鱼个体仍旧表现出一定的聚集效应，同时尽量远离海豚位置。

3）若海豚在一定时间后离开，鱼群能够慢慢恢复自组织形态，否则鱼群呈现出反复的震荡形态。鱼群规模小于一定数量时，鱼群难以恢复自组织形态。另一方面，单只海豚在持续性捕猎时，其捕猎效率随时间的增长而减小，这与客观实际比较吻合。

## 五、模型的评价与推广

### 5.1 模型的优点与缺点

#### 1.模型的优点

1）在查阅沙丁鱼群相关资料的基础上，给出了沙丁鱼对于具体规则的反应情况（包括仿碰撞、聚中，模仿、逃离、惯性、被捕食），从而建立了基于规则反应函数的各种鱼群模型，对客观事实的反应更为准确。

2）本文通过三维笛卡尔模型建立模型，相较于目前比较普遍的二维平面的模型更加精确有效，利用速度与维坐标系来描述运动，将复杂系统的自组织过程转化为在三维空间内的质点运动位移的关系能够有效的追踪每一条鱼的运动状态。

3）在动作反应相对鱼群总运动时间极小的情况下，将模型进行离散化处理，简化

了问题。利用编程（Python 和 Matlab）对问题进行了有效地分析、仿真；给出了沙丁鱼群在无海豚情况下和有一只海豚情况下的运动规律仿真，仿真的结果精确度高，和理论实际符合度好。

4）沙丁鱼群状态的描述方式可以很方便地推广的其他数量的海豚，乃至于更多不同种类的海洋生物，模型的可扩展性高。

**2.模型的缺点**

1）模型在建立过程中对于海豚与沙丁鱼进行了质点化处理，在实际情况中，海豚与鱼群的物理体积是无法忽视的；另外，海豚在近距离内对于沙丁鱼的捕杀成功率也是不确定的（模型默认一定距离内必定成功）。同时，模型未考虑地形和海平面的问题。

2）在对鱼群建模时，默认了鱼群中所有个体是同质体，但实际情况下群体间的个体差异（领导能力、体重、运动能力、感知力、饥饿疲劳度）有很大影响；同时，未考虑海洋中的水体流向、实物分布、天敌分布等。对于海豚和沙丁鱼的感知域，默认海豚能全覆盖鱼群，沙丁鱼是球体感知域，实际中由于地形、视野阻挡、光线等原因，感知域的实际情况也是有很大差异的[5,6]。另外，海豚对于鱼群的追逐方向是否直接指向所有现存鱼群的中心，以及海豚对于沙丁鱼的捕杀成功率都还值得考虑。

3）受限于仿真条件（Python3 和普通 PC），不能对大数量的鱼群进行较高效率的仿真，一般来说数量越多，仿真结果和实际情况更加符合；时间紧迫、问题复杂，在考虑问题时难免忽略部分细节。

## 5.2 模型的改进与推广

**1.模型的改进**

1）加入地形、体积、洋流速度、食物和天敌的分布等因素的考虑，对沙丁鱼群和海豚的各项能力属性（领导力、体积、质量、感知能力、交互能力等）按一定分布随机化处理，以更好地符合客观实际；

2）海豚在接近鱼群时，鱼群会明显呈分裂散开现象。将鱼群进行聚类，以各自的聚类中心来作为海豚对鱼群的追逐方向的考虑因素；

3）本文中的沙丁鱼个体对于规则的反应函数关系仍有提升的；对沙丁鱼进行详细的实验研究将有助于改进函数关系，使得模型准确度再次提高。

**2. 模型的推广**

本文通过理想化的假设，完成了海豚与沙丁鱼群的运动规律的建模描述。事实上，本文的模型可以推广到各类海洋生物在更大范围、更复杂环境下的海洋环境中。本文中仅有一只海豚，实际情况下海豚是成群活动的；在捕食大规模的沙丁鱼群或者其他类似场合，往往出现的不只有海豚一种捕食者，金枪鱼、鲨鱼、鲸类都会参与。对这些运动的参与者，按照本文中的建模方式，可以将其复杂的集群行为转化为合适的数学模型来描述。

## 参考文献

[1] 黄天云. 群体机器人系统的群集自组织运动控制[D]. 大连理工大学, 2015.
[2] 柳玲飞. 鱼群结构的数学建模-以红鼻鱼为例[D]. 上海海洋大学, 2015..
[3] 班晓娟, 吴崇浩, 王晓红,等. 基于多 Agent 的人工鱼群自组织行为算法[J]. 计算机工程, 2007, 33(23):182-184.
[4] 田明伟. 三维空间内鱼群复杂群体行为的建模与仿真研究[D]. 燕山大学, 2013.
[5] 班晓娟, 彭立, 王晓红,等. 人工鱼群高级行为的自组织算法与实现[J]. 计算机科学, 2007, 34(7):193-196.
[6] 王联国. 人工鱼群算法及其应用研究[D]. 兰州理工大学, 2009.
[7] 鄢喜爱,杨金民,田华. Matlab 在数据处理和绘图中的应用[J]. 科学技术与工程,2006,22:3631-3633.
[8] 姜启源, 谢金星, 叶俊. 数学模型.第 3 版[M]. 高等教育出版社, 2003.
[9] 菜鸟教程 Python 教程，http://www.runoob.com/python3/python3-module.html,2018/4/15
[10]赫特兰. Python 算法教程[M]. 人民邮电出版社, 2016.

# 附录

## 1. 鱼群仿真 Python 代码

### Main.py

```
from Sardines.sardines import Sardine
from Sardines.dolphins import Dolphin
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

kill = 0      # 被捕食沙丁鱼数
kill_history = [0]      # 被捕食沙丁鱼数量历史记录
n = 250      # 沙丁鱼数量
showtime = 25      # 海豚出现时间
gotime = 60      # 海豚离开时间
t = 0.05      # 时间间隔
sardines = []      # 保存沙丁鱼实例用
dolphin = Dolphin()      # 海豚实例化
for i in range(n):
    sardines.append(Sardine())      # 将实例化的沙丁鱼保存到 list 里面

for k in range(90):
    x = []      # 保存 x 坐标用
    y = []      # 保存 y 坐标用
    z = []      # 保存 z 坐标用
    center_position = np.array([0.0, 0.0, 0.0])      # 计算沙丁鱼几何中心
    all_position = 0      # 计算所有沙丁鱼和几何中心的距离之和
    for j in range(n):
        if showtime < k < gotime:      # 海豚出现时改变时间间隔
            sardines[j].t = t
            dolphin.t = t
        sardines[j].find_neighborhood(sardines)      # 根据规则更新沙丁鱼的速度
        sardines[j].set_speed_escape_now(dolphin)
        sardines[j].set_speed_all_now()

    if showtime < k < gotime:
        dolphin.set_speed_all_now(sardines)      # 更新海豚速度
    elif gotime <= k:
        dolphin.go_away()      # 使海豚离开
    for i in range(n):
```

```
            sardines[i].set_position_now()      # 更新沙丁鱼位置
            x.append(sardines[i].position_now[0][0])      # 保存沙丁鱼位置
            y.append(sardines[i].position_now[0][1])
            z.append(sardines[i].position_now[0][2])
        if showtime < k:
            dolphin.set_position_now()      # 更新海豚位置

        for i in range(n):      # 判断沙丁鱼是否被吃掉
            if (np.sqrt(np.sum(np.square(sardines[i].position_now[0]-dolphin.position_now))) < 0.15) &
(sardines[i].alive == 1):
                kill += 1
                sardines[i].alive = 0
            center_position += sardines[i].position_now[0]
        center_position = center_position / n
        kill_history.append(kill)
        for i in range(n):
            all_position = np.sqrt(np.sum(np.square(sardines[i].position_now[0] - center_position)))

        # 画图
        fig = plt.figure(1)
        ax = Axes3D(fig)
        ax.scatter(x, y, z, '*')
        if k > showtime:
            ax.scatter(dolphin.position_now[0], dolphin.position_now[1], dolphin.position_now[2], '*r')
        plt.pause(0.00001)
        fig.clear()

# 保存各种数据到本地
for i in range(n):
    np.savetxt('C:/Users/LittleWhite/Desktop/fish/%d.csv'%i, np.column_stack((sardines[i].position_history,
sardines[i].speed_history)), delimiter=',')
np.savetxt('C:/Users/LittleWhite/Desktop/fish/dolphin.csv', np.column_stack((dolphin.position_history,
dolphin.speed_history)), delimiter=',')
np.savetxt('C:/Users/LittleWhite/Desktop/fish/kill.csv', kill_history, delimiter=',')
```

## Sardines.py

```
import numpy as np

class Sardine(object):

    def __init__(self):
        super(Sardine, self).__init__()
```

```
        # 构造鱼的各种数据
        self.speed_gather_now = np.array([0.0, 0.0, 0.0])
        self.speed_imitate_now = np.array([0.0, 0.0, 0.0])
        self.speed_escape_now = np.array([0.0, 0.0, 0.0])
        self.speed_inertia_now = np.array([0.0, 0.0, 0.0])
        self.speed_anticollision_now = np.array([0.0, 0.0, 0.0])
        self.speed_all_last = np.array([0.0, 0.0, 0.0])
        self.speed_all_now = np.array([0.0, 0.0, 0.0])
        self.position_now = np.random.rand(1, 3) * 6 + 6
        self.position_last = self.position_now
        self.position_history = self.position_now
        self.speed_history = self.speed_all_now
        self.neighborhood = []
        self.perception_range = 5       # 感知大小
        self.max_speed = 2       # 最大速度
        self.too_close = 0.36       # 最小舒适距离
        self.too_far = 0.75       # 最大舒适距离
        self.k2 = 1       # 聚中系数
        self.k3 = 0.4       # 模仿系数
        self.k1 = 1       # 防撞系数
        self.k4 = 1      # 逃离系数
        self.k5 = 0.5       # 惯性系数
        self.t = 0.1
        self.alive = 1

    def set_position_now(self):
        # 更新位置
        self.position_last = self.position_now
        self.position_now = self.position_last + self.speed_all_now * self.t
        self.position_history = np.row_stack((self.position_history, self.position_now[0]))

    def find_neighborhood(self, sardines):
        # 寻找感知域内的所有鱼
        self.neighborhood = []
        for sardine in sardines:
            if np.sqrt(np.sum(np.square(self.position_now - sardine.position_now))) <= self.perception_range:
                self.neighborhood.append(sardine)

    def set_speed_inertia_now(self):
        # 惯性原则
        self.speed_inertia_now = self.k5 * self.speed_all_last

    def set_speed_escape_now(self, dolphin):
```

```
        # 逃跑原则
        self.speed_escape_now = np.array([0.0, 0.0, 0.0])
        dir = self.position_now - dolphin.position_now
        dis = np.sqrt(np.sum(np.square(dir)))
        if dis <= self.perception_range:
            # 反函数
            self.speed_escape_now = self.k4/dir

    def set_speed_imitate(self):
        # 模仿原则
        self.speed_imitate_now = np.array([0.0, 0.0, 0.0])
        for sardine in self.neighborhood:
            self.speed_imitate_now += sardine.speed_all_last[0]
        if len(self.neighborhood) == 0:
            self.speed_imitate_now = np.array([0.0, 0.0, 0.0])
            return 1
        self.speed_imitate_now = self.k3 * (self.speed_imitate_now / len(self.neighborhood))

    def set_speed_gather(self):
        # 聚集原则
        self.speed_gather_now = np.array([0.0, 0.0, 0.0])
        # 找所领域内有鱼的几何中心
        for sardine in self.neighborhood:
            self.speed_gather_now += sardine.position_now[0]
        if len(self.neighborhood) == 0:
            self.speed_gather_now = np.array([0.0, 0.0, 0.0])
            return 1
        self.speed_gather_now = self.speed_gather_now / len(self.neighborhood) - self.position_now
        # 用的自创函数
        speed = self.gather_function(np.sqrt(np.sum(np.square(self.speed_gather_now))))
        self.speed_gather_now = speed * self.speed_gather_now / np.sqrt(
            np.sum(np.square(self.speed_gather_now)))

    def set_speed_anticollision(self):
        # 避撞原则
        self.speed_anticollision_now = np.array([0.0, 0.0, 0.0])
        # 领域内所有鱼都是防撞关键鱼
        for sardine in self.neighborhood:
            if list(sardine.position_now[0]) == list(self.position_now[0]):
                continue
            distance = self.position_now - sardine.position_now
            speed = self.anticollision_function(np.sqrt(np.sum(np.square(distance)))) * distance / np.sqrt(
                np.sum(np.square(distance)))
```

```
            self.speed_anticollision_now = speed + self.speed_anticollision_now
        if np.sqrt(np.sum(np.square(self.speed_anticollision_now))) > self.max_speed:
            self.speed_anticollision_now = self.k1 * self.max_speed * self.speed_anticollision_now / np.sqrt(
                np.sum(np.square(self.speed_anticollision_now)))

    def set_speed_all_now(self):
        # 更新速度
        self.set_speed_anticollision()
        self.set_speed_gather()
        self.set_speed_inertia_now()
        self.set_speed_imitate()
        self.speed_all_last = self.speed_all_now
        self.speed_all_now = self.speed_inertia_now + self.speed_gather_now + self.speed_imitate_now +
self.speed_anticollision_now + self.speed_escape_now
        if np.sqrt(np.sum(np.square(self.speed_all_now))) > self.max_speed:
            self.speed_all_now = self.speed_all_now / np.sqrt(np.sum(np.square(self.speed_all_now))) *
self.max_speed
        self.speed_history = np.row_stack((self.speed_history, self.speed_all_now[0]))

    def anticollision_function(self, x):
        # 避撞原则自定义分段函数
        y = 0
        if x < self.too_close:
            y = self.max_speed
        elif self.too_close <= x < self.too_far:
            y = self.max_speed - self.max_speed * np.sqrt((1 - (x - self.too_far) ** 2 /
(self.too_far-self.too_close) ** 2))
        elif self.too_far <= x:
            y = 0
        return y

    def gather_function(self, x):
        # 聚中原则自定义分段函数
        y = 0
        if x < self.too_close:
            y = 0
        elif self.too_close <= x < self.too_far:
            y = self.max_speed - self.max_speed * np.sqrt((1 - (x - self.too_close) ** 2 /
(self.too_far-self.too_close) ** 2))
        elif self.too_far <= x:
            y = self.max_speed
        return y
```

## Dolphin.py

```
import numpy as np
class Dolphin(object):

    def __init__(self):
        super(Dolphin, self).__init__()
        # 构造各种鱼的属性
        self.speed_catch_now = np.array([0.0, 0.0, 0.0])
        self.speed_inertia_now = np.array([0.0, 0.0, 0.0])
        self.speed_all_last = np.array([0.0, 0.0, 0.0])
        self.speed_all_now = np.array([0.0, 0.0, 0.0])
        self.position_now = np.array([15.0, 15.0, 15.0])
        self.position_last = self.position_now
        self.position_history = self.position_now
        self.speed_history = self.speed_all_now
        self.max_speed = 10
        self.t = 1

    def set_position_now(self):
        # 更新位置
        self.position_last = self.position_now
        self.position_now = self.position_last + self.speed_all_now * self.t
        self.position_history = np.row_stack((self.position_history, self.position_now))

    def set_speed_catch(self, sardines):
        # 捕捉原则
        self.speed_catch_now = np.array([0.0, 0.0, 0.0])
        for sardine in sardines:
            self.speed_catch_now += sardine.position_now[0]
        speed = self.speed_catch_now / len(sardines) - self.position_now + np.random.rand(1,3)[0]*2-1
        self.speed_catch_now = speed / np.sqrt(np.sum(np.square(speed))) * self.max_speed

    def set_speed_inertia_now(self):
        # 惯性原则
        self.speed_inertia_now = self.speed_all_last * 2.5

    def set_speed_all_now(self, sardines):
        # 更新速度
        self.set_speed_inertia_now()
        self.set_speed_catch(sardines)
```

```
            self.speed_all_last = self.speed_all_now
            self.speed_all_now = self.speed_catch_now + self.speed_inertia_now + np.random.rand(1,3)[0]
            if np.sqrt(np.sum(np.square(self.speed_all_now))) > self.max_speed:
                self.speed_all_now = self.speed_all_now / np.sqrt(np.sum(np.square(self.speed_all_now))) *
self.max_speed
            self.speed_history = np.row_stack((self.speed_history,self.speed_all_now))

    def go_away(self):
        # 离开
        dir = np.array([-15.0, -15.0, -15.0]) - self.position_now
        self.speed_all_now = dir / np.sqrt(np.sum(np.square(dir))) * self.max_speed
        self.speed_history = np.row_stack((self.speed_history, self.speed_all_now))
```

## 2. Matlab 可视化代码

```
clc;
clear;
close all;
file=dir('C:\Users\LittleWhite\Desktop\fish\*.csv');
file_name=cell(1,1);
file_data=cell(6,1);
for i=1:size(file,1)
    file_name{i,1}=file(i).name;
    file_data{i,1}=csvread(['C:\Users\LittleWhite\Desktop\fish\',file_name{i,1}]);
end
[a,~] = size(file_data);
[b,~] = size(file_data{1,1});
[c,~] = size(file_data{a-1,1});
for i = 2:b
    x_start = [];
    y_start = [];
    z_start = [];
    x_end = [];
    y_end = [];
    z_end = [];
    figure(1);
    clf;
    for j = 1:a-2
        position = file_data{j,1}(i,1:3);
        speed = 0.2*file_data{j,1}(i,4:6);
        x_start = [x_start,position(1)];
        y_start = [y_start,position(2)];
        z_start = [z_start,position(3)];
        x_end = [x_end,speed(1)];
```

```
            y_end = [y_end,speed(2)];
            z_end = [z_end,speed(3)];
        end

quiver3(x_start,y_start,z_start,x_end,y_end,z_end,'MaxHeadSize',10,'Linewidth',1.2,'AutoScaleFactor',1.0,'AutoScale','off');
        hold on;
        if i>b-c+1
            position = file_data{a-1,1}(i-(b-c+1),1:3);
            speed = 0.09*file_data{a-1,1}(i-(b-c+1),4:6);

quiver3(position(1),position(2),position(3),speed(1),speed(2),speed(3),'MaxHeadSize',10,'Linewidth',1.5,'AutoScaleFactor',1.0,'AutoScale','off');
            hold on;
        end
        axis([5 14 6 22 3 17]);
        axis square;
        xlabel('X 轴');
        ylabel('Y 轴');
        zlabel('Z 轴');
        title({['沙丁鱼运动模型仿真'];['时间：',num2str(double(0.1*(i-2))),'S'];['被捕食数量：',num2str(file_data{a,1}(i,1))]},'FontSize',36);
        pause(0.1);
end
delete(gcp('nocreate'));
```