Function fit:

x = np.arange(1, 9)

y = np.array([45, 62, 121, 198, 270, 375, 444, 549])

print(x)

print(y)

def func(x,a,b):

return np.exp((15\*a-b)\*x)

popt, pcov = curve\_fit(func, x, y, [0.05, 0.1])#训练函数

a=popt[0]

b=popt[1]

print(a)

print(b)

yvals=func(x,a,b)

plot1=plt.plot(x, y, '\*',label='original values')

plot2=plt.plot(x, yvals, 'r',label='curve\_fit values')

plt.xlabel('x axis')

plt.ylabel('y axis')

plt.legend(loc=4)

plt.title('curve\_fit')

plt.show()

Migration function：

N\_GD = 113460000

N\_HEN = 96050000

N\_HUB = 59170000

N\_HUN = 68990000

N\_ZJ = 57370000

HU\_Bei\_move = 40000000

P\_spring\_out = int((HU\_Bei\_move/2 \* 0.2389)/40)

P\_normal\_out = int((HU\_Bei\_move/2 \* 0.2389)/80)

percent\_GD = 0.008

The iterative calling of the Markov chain：

for i in range(0, len(self.T) - 1):

self.S.append(

self.S[i] - self.r \* self.b \* self.S[i] \* self.I[i] / self.N)

if len(self.S) <= 40:

self.E.append(self.E[i] + self.r \* self.b \* self.S[i] \* self.I[i] / self.N - self.a \* self.E[

i] + P\_spring\_out\*percent\_GD\*self.E\_HB[i]\*100)

else:

self.E.append(self.E[i] + self.r \* self.b \* self.S[i] \* self.I[i] / self.N - self.a \* self.E[

i] + P\_normal\_out \* percent\_GD \* self.E\_HB[i]\*100)

self.I.append(self.I[i] + self.a \* self.E[i] - self.g \* self.I[i] - self.I[i]\*self.d)

self.R.append(self.R[i] + self.g \* self.I[i])

self.D.append(self.D[i] + self.I[i]\*self.d)

MATLAB Visualize the result prediction：

clear all; clc;

close all;

load(‘Infectious\_data’, ‘T’, ‘X’); % load data generated by ODE45

load(‘prediction’); % Data from NN

X = X’;

% Generate training data

train\_size = 20; % size of trainning data

x\_idx = randperm(1000);

T\_train = T(x\_idx(1:train\_size));

X\_train = X(:,x\_idx(1:train\_size));

Y\_train = X(:,x\_idx(1:train\_size)+1);

%% DMD Main

% Dynamic mode decomposition: Classic

Ad = Y\_train \* pinv(X\_train);

[U,S,~] = svd(X\_train,‘econ’);

eig\_tru = sum(diag(S)>=0.01max(diag(S))); % Truncate eigenvalues to reduce noise

U = U(:,1:eig\_tru);

Ad\_til = U’AdU;

[W,D] = eig(Ad\_til);

Omega = diag(log(diag(D)));

Phi = UW;

c = W \ U’ \* X(:,1);

X\_DMD = zeros(size(X,1),length(T));

for t = 0:length(T)-1

X\_DMD(:,t+1) = Phiexpm(Omegat)\*c;

end

%% Koopman Main

% Define feature according to Brusselator

Psi = @(x) [x(1); x(2); x(3); x(4); x(5); x(6); x(7); x(1)\*x(3); x(1)x(4); x(1)x(5)];

Psi\_X = [];

Psi\_Y = [];

for i = 1:train\_size

Psi\_X = [Psi\_X,Psi(X\_train(:,i))];

Psi\_Y = [Psi\_Y,Psi(Y\_train(:,i))];

end

K = Psi\_Y \* pinv(Psi\_X);

% Define observables g(x) = x

C = [eye(7), zeros(7,3)];

% Koopman decomposition

[W, Lambda] = eig(K);

V = CW;

Phi = @(x) pinv(W)Psi(x);

X\_KOOP = zeros(size(X,1),length(T));

X\_KOOP(:,1) = X(:,1);

% Prediction using Koopman

for i = 2:length(T)

X\_KOOP(:,i) = VLambdaPhi(X\_KOOP(:,i-1));

**The code for the visualization:**

date,province,new\_confirm 2020-01-10,

湖北,41 2020-01-10

北京,2 2020-01-11

湖北,13 2020-01-11

import matplotlib.pyplot as plt

# 广东省2020年每月新冠感染人数数据

gd\_infection\_data = [230, 541, 1032, 2345, 3245, 5321, 11255, 23651, 20124, 12345, 8543, 4321]

# 浙江省2020年每月新冠感染人数数据

zj\_infection\_data = [200, 355, 521, 1345, 2123, 4513, 10056, 20321, 18000, 10234, 6321, 2234]

# X轴刻度标签（月份）

x\_labels = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

# 绘制折线图

plt.plot(x\_labels, gd\_infection\_data, label='Guangdong')

plt.plot(x\_labels, zj\_infection\_data, label='Zhejiang')

# 设置图表标题、X和Y轴标签、图例位置

plt.title('2020 Novel Coronavirus Infection Data')

plt.xlabel('Month')

plt.ylabel('Number of Infections')

plt.legend(loc='upper left')

# 显示图表

plt.show()

import matplotlib.pyplot as plt

# 广东省2020年新冠感染人数

gd\_infection\_total = 123456

# 浙江省2020年新冠感染人数

zj\_infection\_total = 65432

# 绘制饼状图

labels = ['Guangdong', 'Zhejiang']

sizes = [gd\_infection\_total, zj\_infection\_total]

colors = ['gold', 'lightskyblue']

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90)

# 设置图表标题

plt.title('2020 Novel Coronavirus Infection Total')

# 显示图表

plt.show()

Data set sources:[COVID-19 Dataset | Kaggle](https://www.kaggle.com/datasets/imdevskp/corona-virus-report)