东南大学

仿真实验大作业报告

课程名称：生物系统建模与分析

小组成员：

_____

_____

_____

_____

实验一：分子动力学模拟

1，实验目的

神经网络模型试图模拟大脑中发生的信息处理，并广泛应用于各种应用中，包括自动模式识别。

Rost-Sander 数据集包含的蛋白质结构涵盖了相对广泛的域类型、组成和长度范围。文件 RostSanderDataset.mat 包含了该数据集的一个子集，其中对每个蛋白质序列报告了每个残基的结构分配。

本实验中我们使用前馈神经网络和深度学习来进行蛋白质二级结构预测。通过构建一个神经网络，根据训练阶段中观察到的结构模式，学习给定蛋白质中每个残基的结构状态（螺旋、片段或卷曲）。

2，实验内容

(1) 定义网络架构：定义一个具有一个输入层、一个隐藏层和一个输出层的神经网络
(2) 创建神经网络：使用上面定义的输入和目标矩阵创建一个模式识别神经网络。
(3) 训练神经网络：模式识别网络使用默认的缩放共轭梯度算法进行训练
(4) 网络反应分析：通过考虑训练网络的输出并将其与预期结果(目标)进行比较来检查混淆矩阵
(5) 改进神经网络：增加训练向量的数量，提高预测精度
(6) 评估网络性能：通过计算预测质量指数来详细评估结构预测

3，实验代码/流程细节（截图或代码文本插入，或另附）

实验代码：

```
load RostSanderDataset.mat
N = numel(allSeq);
id = allSeq(7).Header % 给定蛋白质序列的注释
seq = int2aa(allSeq(7).Sequence) % 蛋白质序列
str = allSeq(7).Structure % 结构分配
```

为了确保结果的可重现性，我们将全局随机生成器重置为加载文件中的保存状态，如下所示：

```
rng(savedState);
```

(1) 定义网络架构

定义一个具有一个输入层、一个隐藏层和一个输出层的神经网络。输入层在每个输入氨基酸序列中编码一个滑动窗口，并对窗口中中心残基的结构状态进行预测。

根据给定残基位置的二级结构与预测点两侧的 8 个残基之间的统计相关性选择大小为 17 的窗口。每个窗口位置使用大小为 20 的二进制数组进行编码，每种氨基酸类型都有一个元素。在每组 20 个输入中，与给定位置的氨基酸类型相对应的元素被设置为 1，而所有其他输入都被设置为 0。因此，输入层由 R = 17x20 个输入单元组成，即 17 组 20 个输入 h。

在下面的代码中，我们首先通过创建一个 Hankel 矩阵来确定每个蛋白质序列对应于大小为 W 的滑动窗口的所有可能的子序列，其中第 i 列表示从原始序列#中第 i 个位置开始的子序列，对于窗口中的每个位置，我们创建一个大小为 20 的数组，如果给定位置上的残数具有等于 j 的数字表示，我们将第 j 个元素设置为 1。

```
%% 定义网络架构
```

```
W = 17; % 滑动窗口大小

% === 输入的二值化
for i = 1:N
    seq = double(allSeq(i).Sequence);   % 当前序列
    win = hankel(seq(1:W),seq(W:end));  % 所有可能的滑动窗口
    myP = zeros(20*W,size(win,2));      % 对于当前序列的输入矩阵
    for k = 1:size(win, 2)
        index = 20*(0:W-1)' + win(:,k); % 每个位置 k 的输入数组
        myP(index,k) = 1;
    end
    allSeq(i).P = myP;
end
```

神经网络的输出层由三个单元组成，每个单元对应一个考虑的结构状态(或类)，它们使用二进制方案进行编码。为了创建神经网络的目标矩阵，我们首先从数据中获得与滑动窗口对应的所有可能子序列的结构赋值。然后考虑每个窗口的中心位置，并使用以下二进制编码转换相应的结构分配：100 表示线圈状，010 表示薄片状，001 表示螺旋状。

```
%%

cr = ceil(W/2); % 中心残馀位置

% === 目标二值化
for i = 1:N
    str = double(allSeq(i).Structure); % 当前结构分配
    win = hankel(str(1:W),str(W:end)); % 所有可能的滑动窗口
    myT = false(3,size(win,2));
    myT(1,:) = win(cr,:) == double('C');
    myT(2,:) = win(cr,:) == double('E');
    myT(3,:) = win(cr,:) == double('H');
    allSeq(i).T = myT;
end
```

通过以下的等效代码，以一种更简洁的方式对上述两步中描述的输入矩阵和目标矩阵进行二值化：

```
%%

% === 输入和目标的简明二值化
for i = 1:N
    seq = double(allSeq(i).Sequence);
    win = hankel(seq(1:W),seq(W:end)); % 并发输入(滑动窗口)

    % === 输入矩阵的二值化
    allSeq(i).P = kron(win,ones(20,1)) == kron(ones(size(win)),(1:20)');
```

```
    % === 目标矩阵的二值化
    allSeq(i).T = allSeq(i).Structure(repmat((W+1)/2:end-(W-1)/2,3,1)) == ...
        repmat(('CEH')',1,length(allSeq(i).Structure)-W+1);
end
```

一旦我们为每个序列定义了输入矩阵和目标矩阵，我们就创建了一个输入矩阵 P 和目标矩阵 T，它们代表了输入到网络中的所有序列的编码。

```
% === 构造输入矩阵和目标矩阵
P = double([allSeq.P]); % 输入矩阵
T = double([allSeq.T]); % 目标矩阵
```

(2) 创建神经网络

二级结构预测问题是一个模式识别问题，其中网络被训练来识别当给定滑动窗口中观察到特定残基时最可能出现的中心残基的结构状态。我们使用上面定义的输入和目标矩阵创建一个模式识别神经网络，并指定大小为 3 的隐藏层。

```
hsize = 3;
net = patternnet(hsize);
net.layers{1} %隐藏层
net.layers{2} %输出层
```

(3) 训练神经网络

模式识别网络使用默认的缩放共轭梯度算法进行训练，但也可以使用其他算法(请参阅深度学习工具箱文档以获取可用函数列表)。在每个训练周期，训练序列通过上面定义的滑动窗口呈现给网络，每次一个残差。每个隐藏单元通过使用传递函数 logsig 对从输入层接收的信号进行变换，产生介于或接近 0 或 1 之间的输出信号，模拟神经元的放电。调整权重，使每个单元的观察输出与目标矩阵指定的期望输出之间的误差最小化。

```
% === 使用 log sigmoid 作为传递函数
net.layers{1}.transferFcn = 'logsig';

% === 训练网络
[net,tr] = train(net,P,T);
```

在训练过程中，训练工具窗口打开并显示进度。显示了训练细节，如算法、性能标准、考虑的错误类型等。

使用函数视图 view(net)生成神经网络的图形视图。

在神经网络训练过程中可能出现数据过拟合，即网络倾向于记住训练示例，而不学习如何推广到新的情况。提高泛化的默认方法被称为提前停止，它包括将可用的训练数据集分为三个子集：(i)训练集，用于计算梯度和更新网络权重和偏差；(ii)验证集，在训练过程中

对其误差进行监控，因为当数据过拟合时，它往往会增加；(iii)测试集，其误差可用于评估数据集划分的质量。

在使用函数 train 时，默认情况下，数据是随机划分的，60%的样本分配给训练集，20%分配给验证集，20%分配给测试集，但可以通过指定属性 net 应用其他类型的划分。divideFnc(默认的除数)。三个子集中残基的结构组成具有可比性，从以下内容可以看出：

```
[i,j] = find(T(:,tr.trainInd));
Ctrain = sum(i == 1)/length(i);
Etrain = sum(i == 2)/length(i);
Htrain = sum(i == 3)/length(i);

[i,j] = find(T(:,tr.valInd));
Cval = sum(i == 1)/length(i);
Eval = sum(i == 2)/length(i);
Hval = sum(i == 3)/length(i);

[i,j] = find(T(:,tr.testInd));
Ctest = sum(i == 1)/length(i);
Etest = sum(i == 2)/length(i);
Htest = sum(i == 3)/length(i);

figure()
pie([Ctrain; Etrain; Htrain]);
title('训练数据集的结构赋值');
legend('C', 'E', 'H')

figure()
pie([Cval; Eval; Hval]);
title('验证数据集中的结构赋值');
legend('C', 'E', 'H')

figure()
pie([Ctest; Etest; Htest]);
title('测试数据集的结构分配');
legend('C', 'E', 'H')
```

函数 plotperform 显示训练迭代通过时的训练、验证和测试错误的趋势。

```
figure()
plotperform(tr)
```

当验证误差增加到指定的迭代次数(6)或达到允许的最大迭代次数(1000)时，训练过程停止。

(4) 网络反应分析

为了分析网络响应，通过考虑训练网络的输出并将其与预期结果（目标）进行比较来检查混淆矩阵。

对角线单元格显示了每个结构类正确分类的残基位置的数量。非对角线单元格显示了错误分类的残基位置的数量（例如，螺旋位置被预测为螺旋位置）。对角线单元格对应于正确分类的观察结果。每个单元格中都显示了观测次数和观测总数的百分比。该图最右边的一列显示了预测属于正确和错误分类的每个类别的所有示例的百分比。这些指标通常分别被称为精度（或正预测值）和错误发现率。该图底部的行显示了属于每个类别的所有示例中正确分类和错误分类的百分比。这些指标通常分别称为召回率（或真阳性率）和假阴性率。该图右下角的单元格显示了总体精度。

我们还可以考虑接受者工作特征（ROC）曲线，即真阳性率（敏感性）与假阳性率（1-specificity）的图。

(5) 改进神经网络以获得更准确的结果

因为此时神经网络较简单，为了提高预测精度，我们通过增加训练向量的数量。增加用于训练的序列数量需要一个更大的蛋白质结构数据库，并适当分布卷曲、螺旋和片状元素。

增加输入值的数量。增加窗口大小或添加更多相关信息（如氨基酸的生化特性）是有效的选择。

我们可以在创建模式识别网络时指定更多的隐藏层或者增加隐藏层的大小：

```
hsize = [3 4 2];
net3 = patternnet(hsize);

hsize = 20;
net20 = patternnet(hsize);
```

通过设置 net20，将网络初始权值分配到-0.1 ~ 0.1 范围内的随机值。属性如下：

```
%权重设置
net20.IW{1} = -.1 + (.1 + .1) .* rand(size(net20.IW{1}));
net20.LW{2} = -.1 + (.1 + .1) .* rand(size(net20.LW{2}));
```

一般来说，较大的网络(包含 20 个或更多隐藏单元)在蛋白质训练集上获得更好的准确度，但在预测准确度上较差。由于一个 20 个隐藏单元的网络包含近 7000 个权重和偏差，因此网络通常能够很好地拟合训练集，但失去了泛化能力。在密集训练和预测精度之间的妥协是神经网络的基本限制之一。

```
net20 = train(net20,P,T);
O20 = sim(net20,P);
```

```
numWeightsAndBiases = length(getx(net20))
```

(6) 评估网络性能

通过计算预测质量指数来详细评估结构预测，该指数表明对特定状态的预测有多好，以及是否发生了过度预测或低估。我们定义状态 S（S = {C, E, H}）的指数 pcObs(S) 为状态 S 正确预测的残数除以状态 S 观察到的残数。同样，我们定义状态 S 的指数 pcPred(S) 为状态 S 正确预测的残数除以状态 S 预测的残数。

```
%% 评估网络性能

[i,j] = find(compet(O));
[u,v] = find(T);

% === 当观察到给定状态时，计算正确预测的分数
pcObs(1) = sum(i == 1 & u == 1)/sum (u == 1); % 状态 C
pcObs(2) = sum(i == 2 & u == 2)/sum (u == 2); % 状态 E
pcObs(3) = sum(i == 3 & u == 3)/sum (u == 3); % 状态 H

% === 当预测给定状态时，计算正确预测的分数
pcPred(1) = sum(i == 1 & u == 1)/sum (i == 1); % 状态 C
pcPred(2) = sum(i == 2 & u == 2)/sum (i == 2); % 状态 E
pcPred(3) = sum(i == 3 & u == 3)/sum (i == 3); % 状态 H

% === 比较预测质量指标
figure()
bar([pcObs' pcPred'] * 100);
ylabel('正确预测位置 (%)');
ax = gca;
ax.XTickLabel = {'C';'E';'H'};
legend({'观察值','预测值'});
```

实验结果：

神经网络训练 (2023-06-16 18:03:41)

网络图

训练结果

训练结束: 已达到最小梯度

训练进度

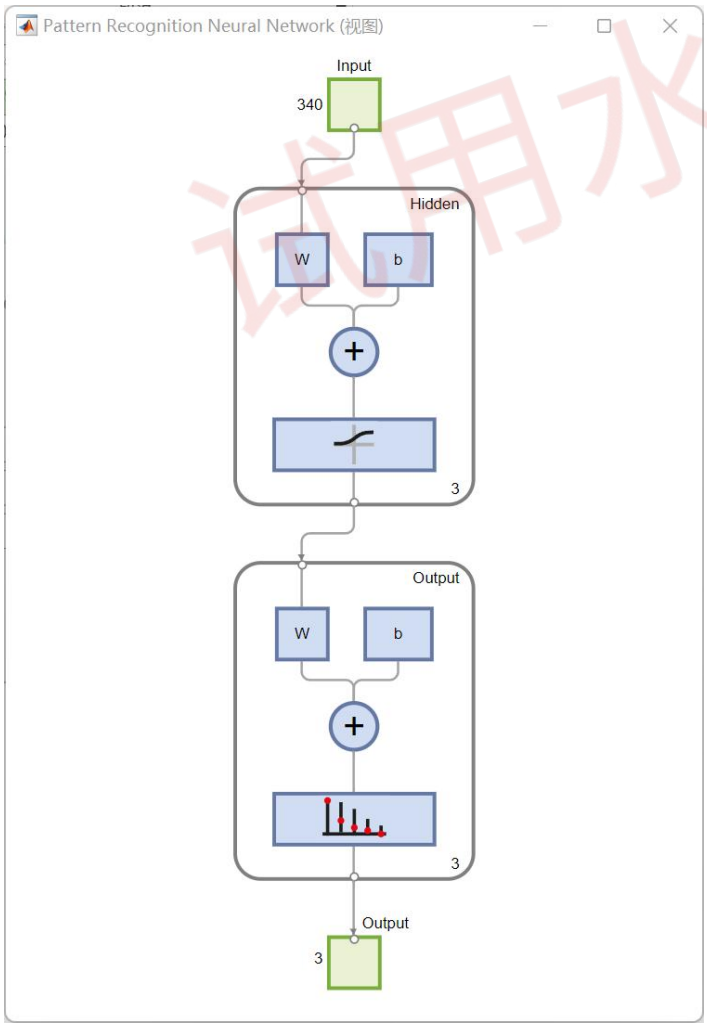
单位	初始值	停止值	目标值
轮	0	8	1000
历时	-	00:00:04	-
性能	0.552	0.349	0
梯度	0.396	8.05e-07	1e-06
验证检查	0	2	6

训练算法

数据划分: 随机 dividerand
训练: 量化共轭梯度 trainscg
性能: 交叉熵 crossentropy
计算: MEX

训练图

性能 | 训练状态
误差直方图 | 混淆
受试者工作特征



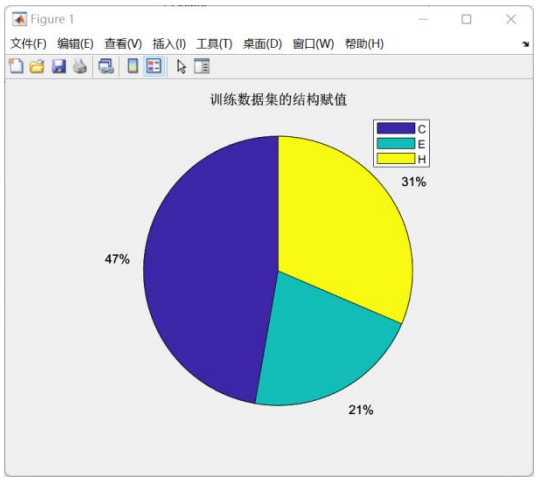


Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
训练数据集的结构赋值
C
E
H
31%
47%
21%

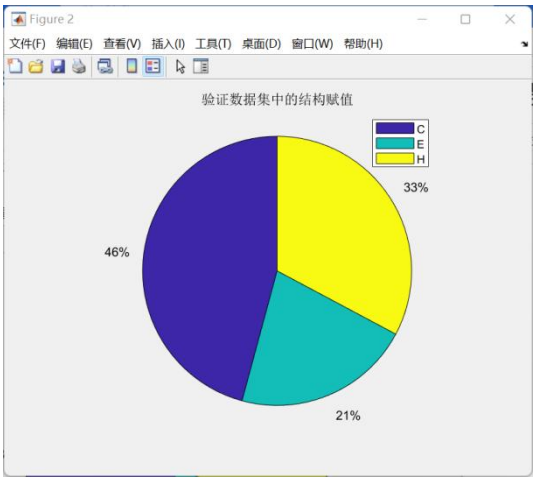


Figure 2
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
验证数据集中的结构赋值
C
E
H
33%
46%
21%

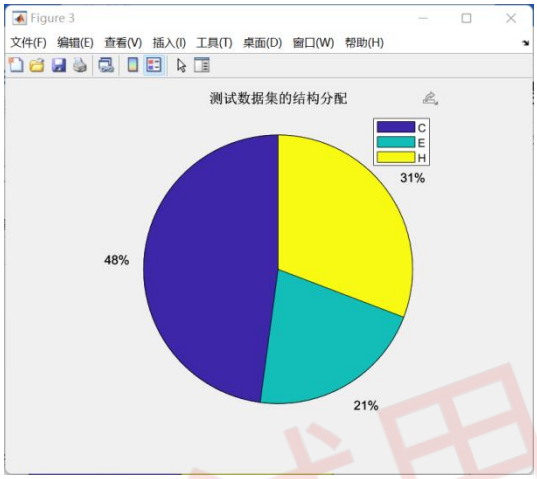
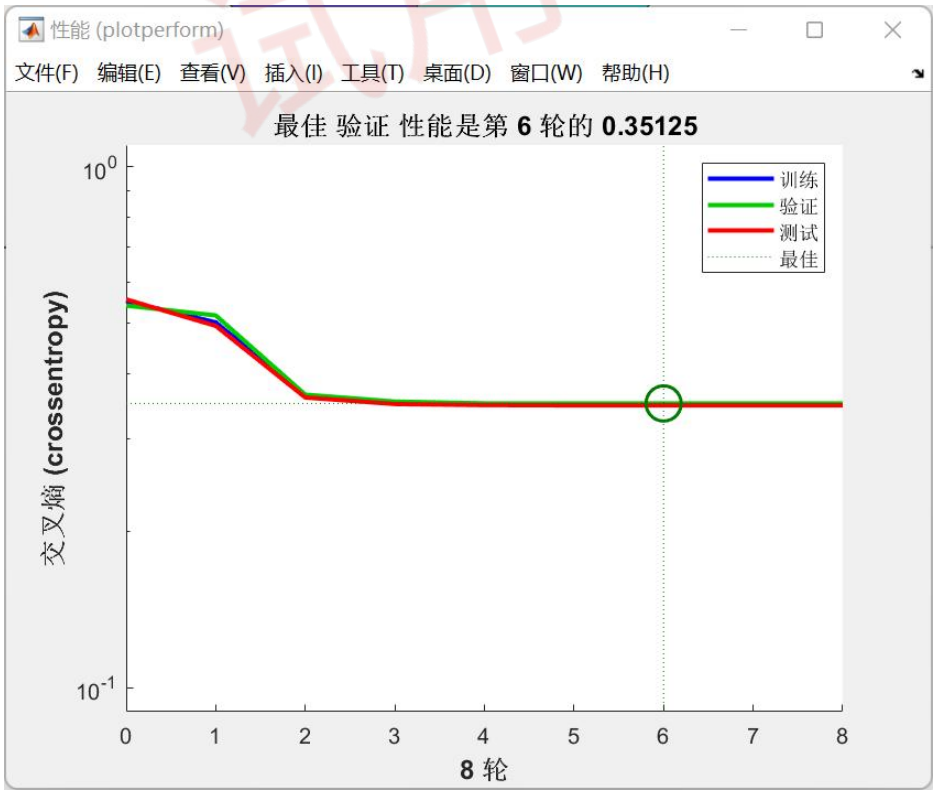
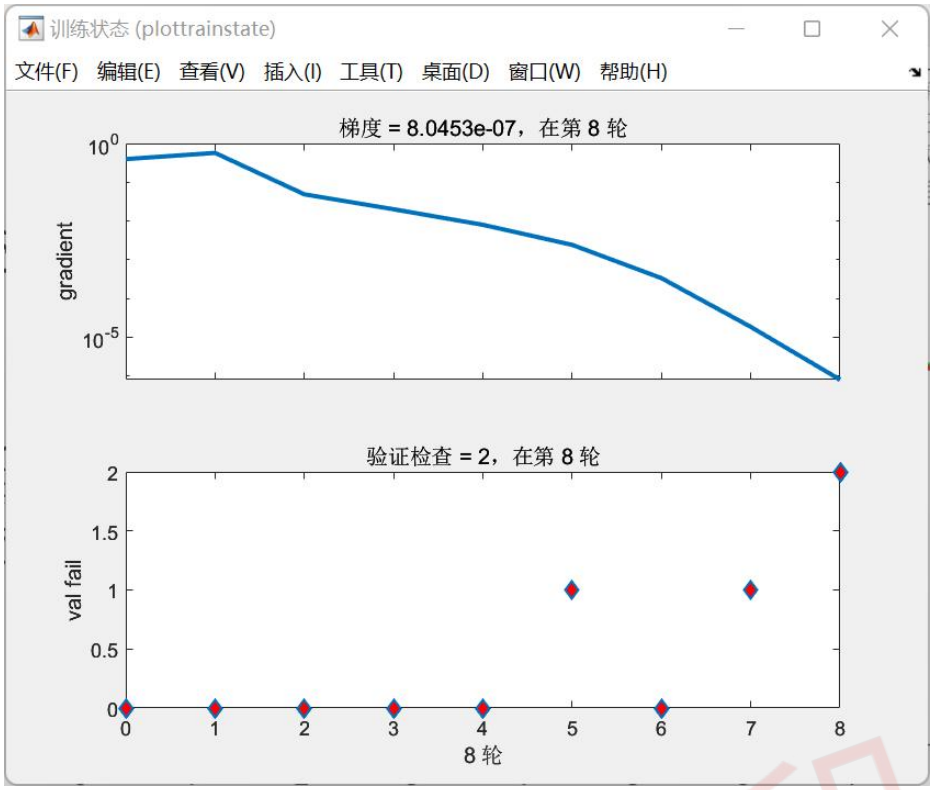
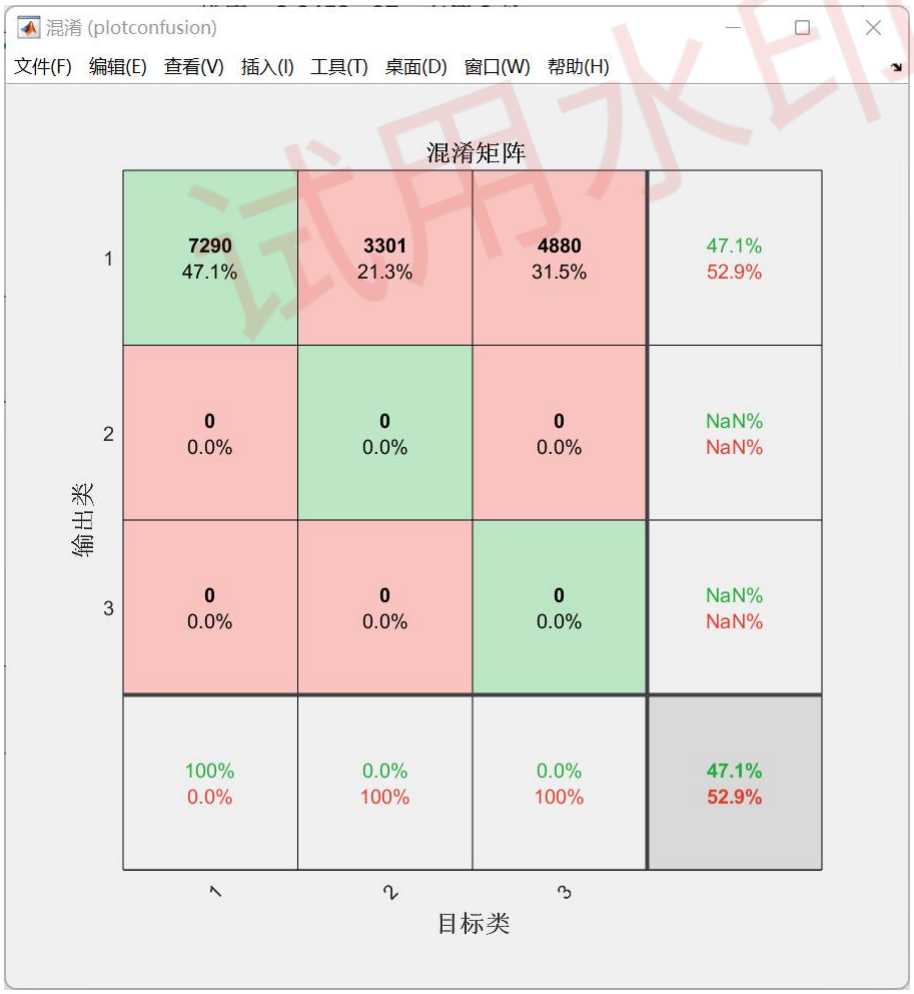
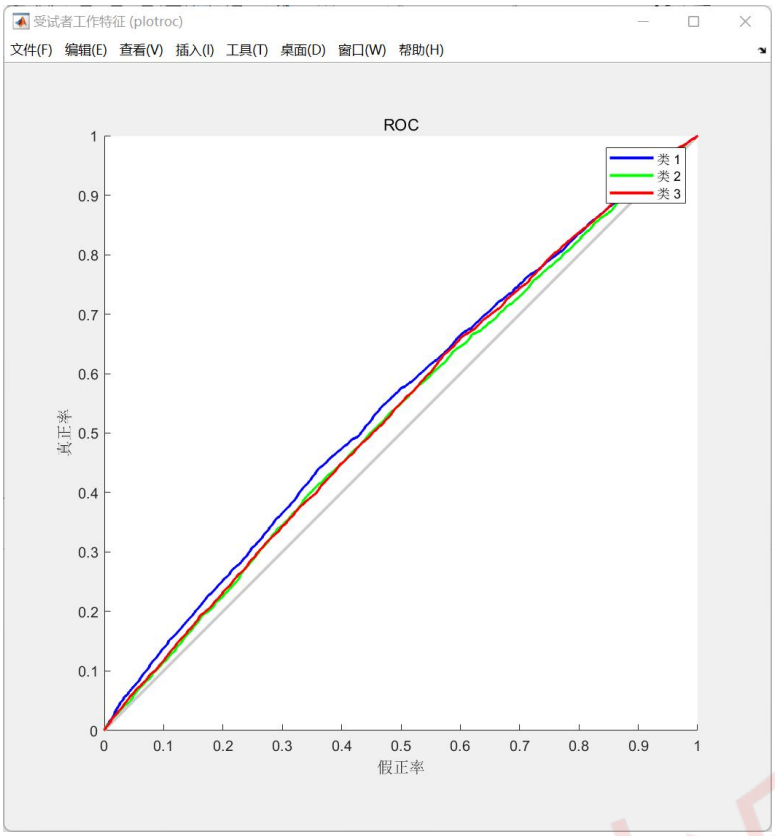


Figure 3
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
测试数据集的结构分配
C
E
H
31%
48%
21%


性能 (plotperform)
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
最佳 验证 性能是第 6 轮的 0.35125
训练
验证
测试
最佳
交叉熵 (crossentropy)
8 轮


训练状态 (plottrainstate)
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
梯度 = 8.0453e-07，在第 8 轮
gradient
验证检查 = 2，在第 8 轮
val fail
8 轮


混淆 (plotconfusion)
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
混淆矩阵
输出类
目标类
7290
47.1%
3301
21.3%
4880
31.5%
47.1%
52.9%
0
0.0%
0
0.0%
0
0.0%
NaN%
NaN%
0
0.0%
0
0.0%
0
0.0%
NaN%
NaN%
100%
0.0%
0.0%
100%
0.0%
100%
47.1%
52.9%


受试者工作特征 (plotroc)
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
ROC
类 1
类 2
类 3
真正率
假正率
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


神经网络训练 (2023-06-16 18:03:49)
网络图
训练结果
训练结束: 满足验证条件
训练进度
单位 初始值 停止值 目标值
轮 0 60 1000
历时 - 00:00:03 -
性能 0.499 0.277 0
梯度 2.33 0.0369 1e-06
验证检查 0 6 6
训练算法
数据划分: 随机 dividerand
训练: 量化共轭梯度 trainscg
性能: 交叉熵 crossentropy
计算: MEX
训练图
性能
训练状态
误差直方图
混淆
受试者工作特征

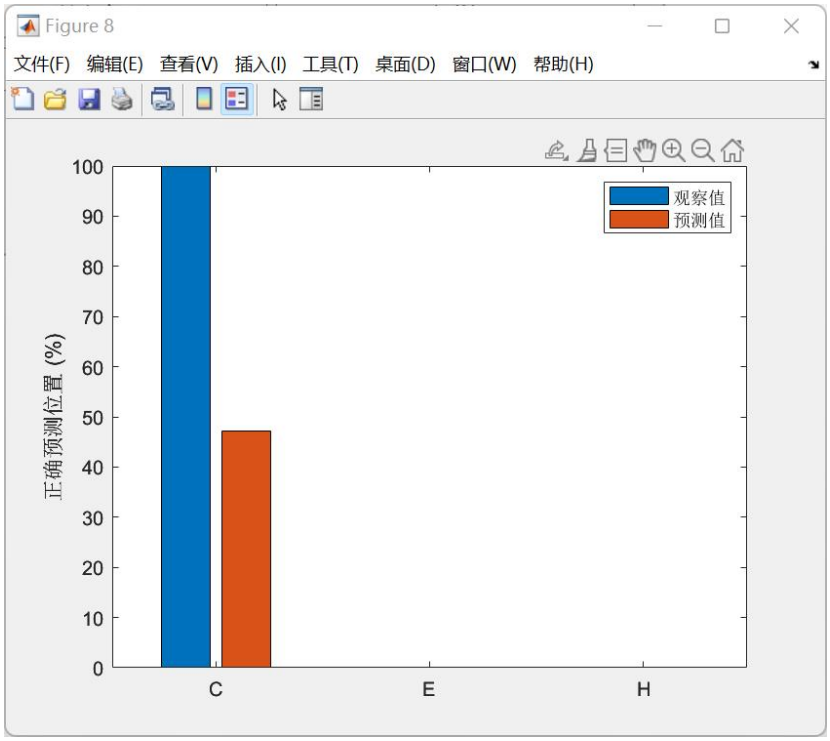


Figure 8
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
正确预测位置 (%)
100
90
80
70
60
50
40
30
20
10
0
C
E
H
观察值
预测值

实验二：生物系统建模分析——内分泌系统血糖调节

1，实验目的

Ⅰ型糖尿病（T1DM）是由人体自身免疫反应引起的糖代谢疾病。I 型糖尿病患者的免疫系统会攻击产生胰岛素的胰岛 B 细胞，使之无法正常分泌胰岛素，从而引起血糖升高。我国是世界上 T1DM 发病率较低的国家，但由于人口基数大，加之多数患者血糖控制不达标，血糖波动大，各种急慢性并发症发生率较高，T1DM 仍是一个十分严峻的挑战。

Ⅰ型糖尿病(T1D) 患者的血糖自动调节是长久以来人造胰腺项目的最终目标。该研究的核心是控制算法，该算法可闭合胰岛素泵和连续葡萄糖测量(CGM)传感器之间的回路。比例积分微分与模型预测控制是目前人工胰腺研究的主要内容。

本实验旨在通过 matlab 软件进行血糖调节模型的仿真研究，建立了胰岛素调控下的血糖模型和基于 Simulink 设计的人工胰腺模型。

2，实验内容

模型①是在使用皮下葡萄糖测量对 I 型糖尿病患者血糖进行模型预测控制中发现的模型，反映了血糖浓度与胰岛素含量之间的变化关系。

模型②是基于 Simulink 设计的人工胰腺模型，通过定期检测葡萄糖，利用反馈调节判断胰岛素注射情况。

3，实验代码/流程细节

①Blood Glucose Model for Insulin Control（胰岛素调控下的血糖模型）

Matlab 代码：

blood_glucose.m:

```
% Blood Glucose Model

function xdot = blood_glucose(t,x)

global u A

% Input (1)
% Insulin infusion rate (mU/min)
U = 3;

% States (3)
% Plasma Glucose Conc. (mmol/L)
G = x(1,1);
% Plasma Insulin Conc. (mU/L) in remote compartment
X = x(2,1);
% Plasma Insulin Conc. (mU/L)
I = x(3,1);
```

```
% Disturbances (1):
% Meal glucose disturbance (mmol/L-min)
% Disturbance from the large meal
D = 3 * exp(-0.05 * t);

% Parameters
% Basal values of glucose and insulin conc.
G_basal = 4.5; % mmol/L
X_basal = 15; % mU/L
I_basal = 15; % mU/L
% For a type-I diabetic
P1 = 0.028735; % min-1
P2 = 0.028344; % min-1
P3 = 5.035e-5; % mU/L
V1 = 12; % L
n = 5/54; % min

Gdot = -P1 * (G - G_basal) - (X - X_basal) * G + D;
Xdot = -P2 * (X - X_basal) + P3 * (I - I_basal);
Idot = -n * I + U / V1;

% Vector to return
xdot = [Gdot; Xdot; Idot];
```

step.m:

```
% Created for the blood glucose step response
global u A


% Steady State Initial Conditions for the States
% Basal values of glucose and insulin conc.
G_ss = 4.5; % mmol/L
X_ss = 15; % mU/L
I_ss = 15; % mU/L

x_ss = [G_ss;I_ss;X_ss];

% Steady State Initial Condition for the Control
u_ss = 16.667; % mU/min

% Steady State for the Disturbance
d_ss = 0; % mmol/L-min

% Final Time (min)
tf = 400;
```

```
[t,x] = ode15s('blood_glucose',[0 tf],x_ss);

% Separate out the state values
G = x(:,1);
X = x(:,2);
I = x(:,3);

% Plot the results
figure(1);
plot(t,G);
legend('Glucose');
xlabel('Time (min)');
ylabel('mmol/L');

figure(2);
plot(t,I,t,X);
legend('Plasma Insulin','Remote Compartment Insulin');
xlabel('Time (min)');
ylabel('mU/L');
```

运行“step.m”文件后得到的仿真结果：

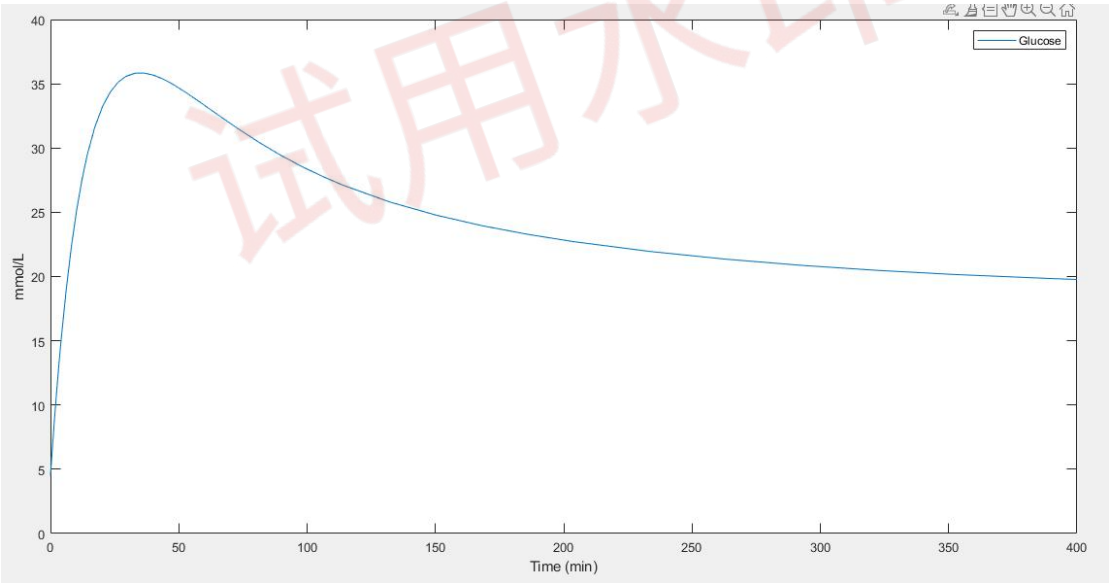


图 1 血糖浓度随时间的变化过程模拟

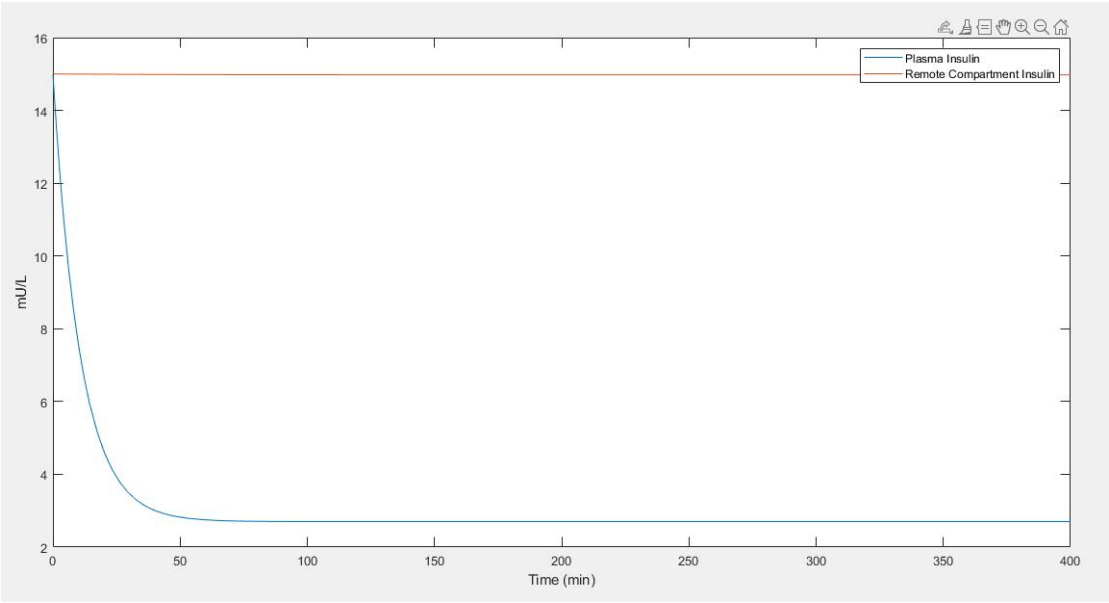


图 2 血液中胰岛素浓度和远程隔室中胰岛素浓度随时间变化模拟

分析：当血糖含量升高时，血液中的胰岛素会比远程隔室中的胰岛素优先发挥作用，使血糖含量下降，当血糖含量平稳时，血液中的胰岛素含量保持相对稳定。

②Blood Glucose Regulation in a Type-I Diabetic in Simulink（Simulink 中 I 型糖尿病患者的血糖调节）

实验过程：

打开文件“diabetes_3pid.slx”，可以看到 simulink 模型，如图 3：

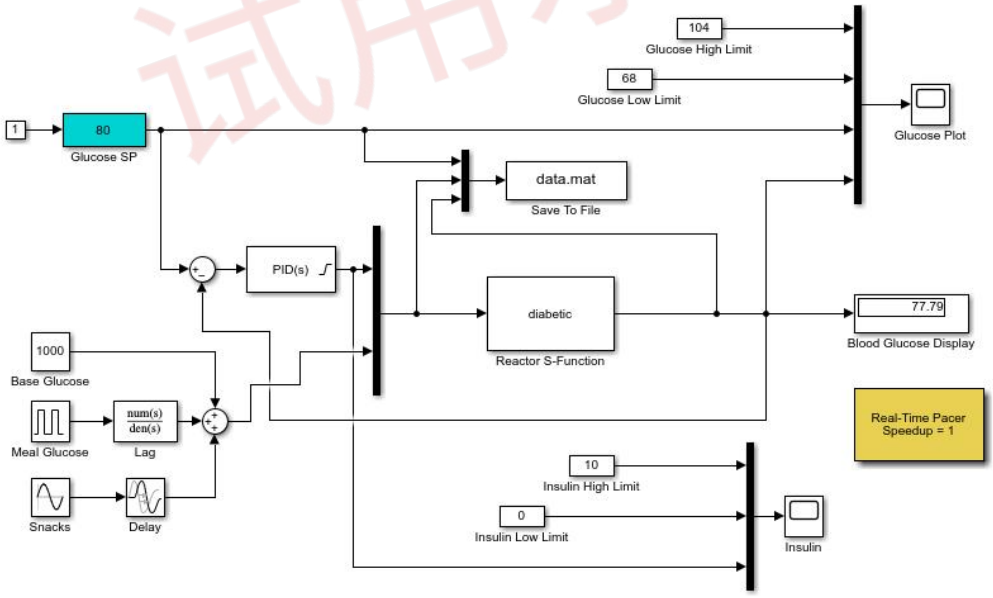


图 3 simulink 模型

运行该模型，得到如下结果：

图 4 在人工胰腺作用下血糖浓度的变化模拟

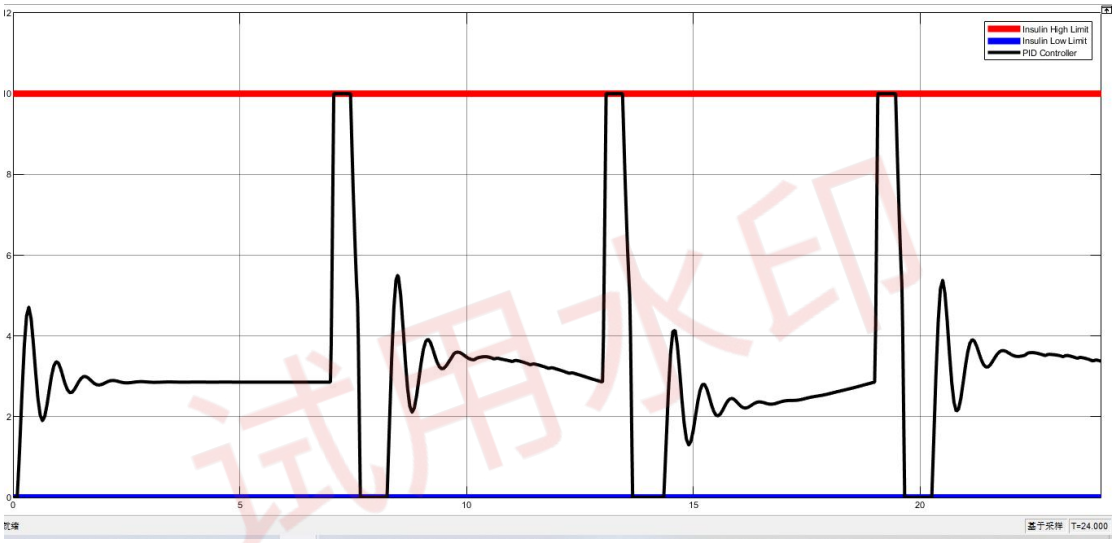


图 5 胰岛素注射速率

或运行 MATLAB 代码文件“plot_results.m”得到如下结果：

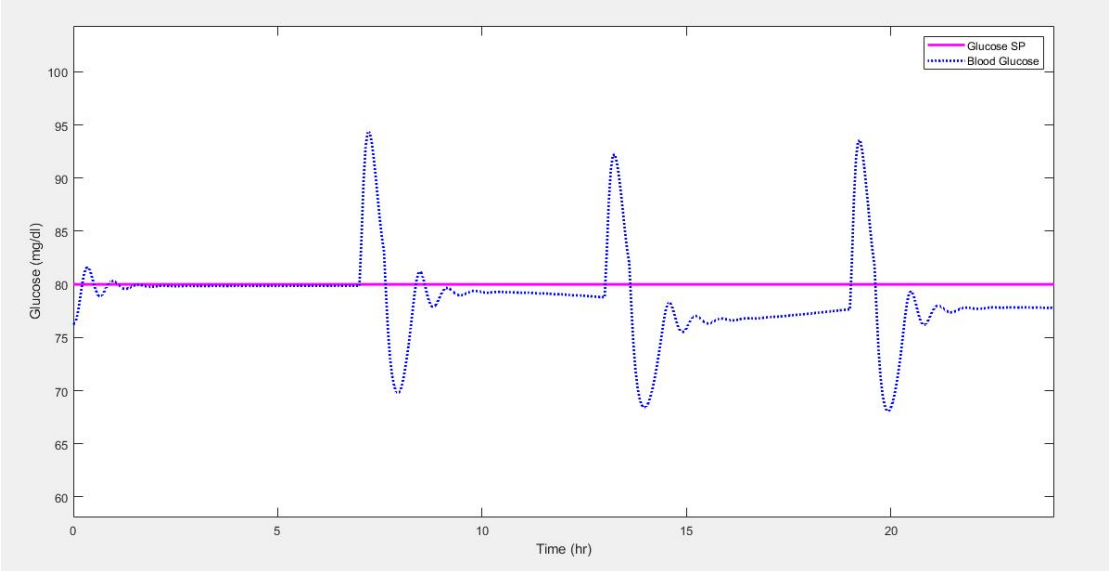


图 6 在人工胰腺作用下血糖浓度的变化模拟

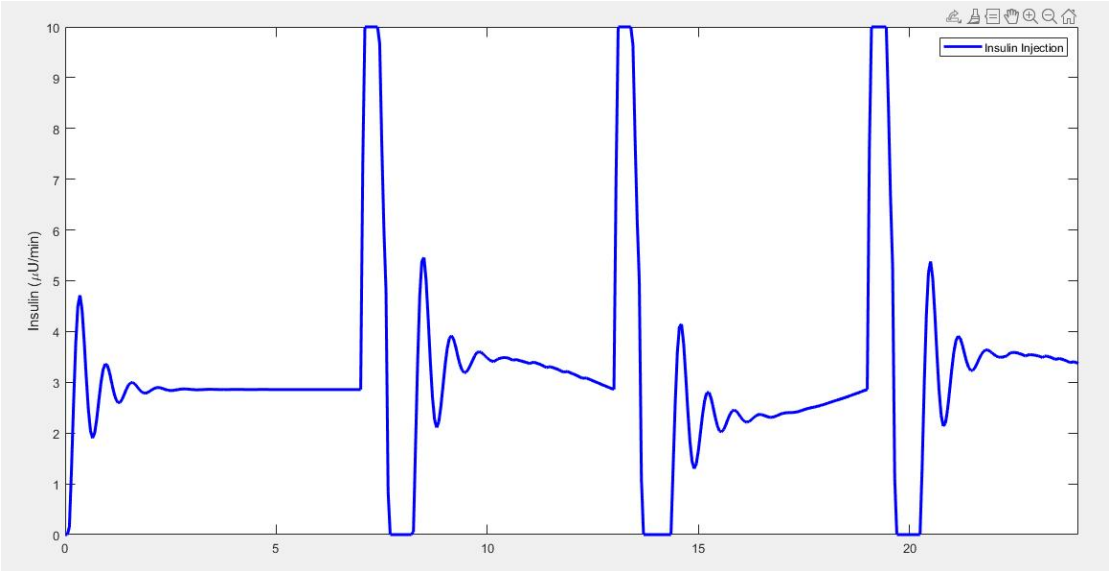


图 7 胰岛素注射速率

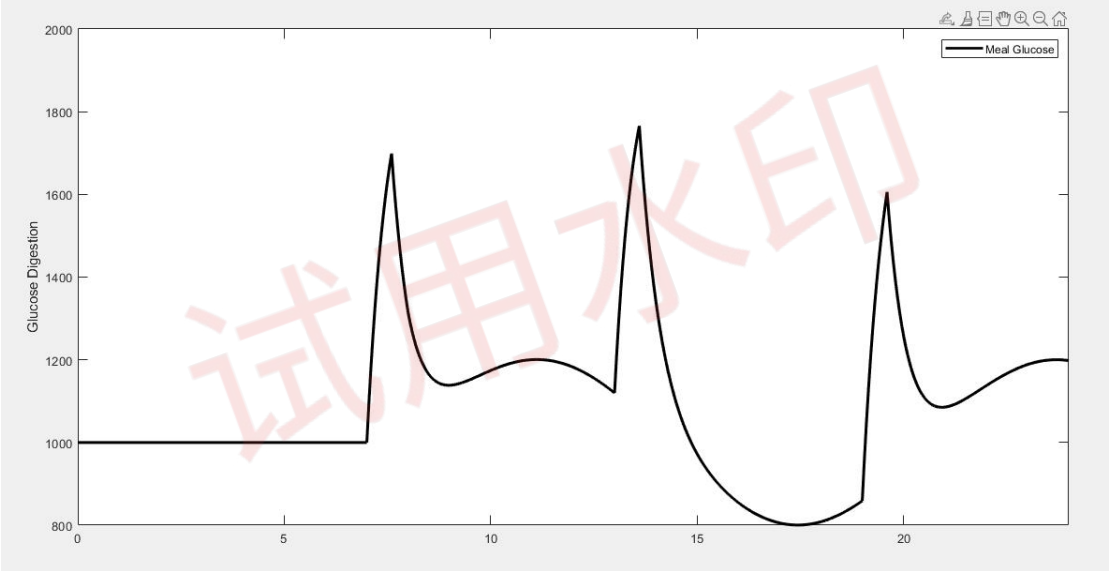


图 8 葡萄糖消化量与时间关系

分析：人工胰腺通过反馈调节注射胰岛素，使Ⅰ型糖尿病病人进食后的血糖浓度可以保持在一个比较健康的水平。

实验三：生物系统建模分析——呼吸系统建模 Simulation of Respiratory Mechanics on Simulink with GUI

1，实验目的

呼吸运动的基本意义是使肺内气体与外界气体交流，有效地提供机体代谢所需的氧，排出体内产生的二氧化碳。通过呼吸运动原理，可用人工方法控制胸廓有节律的扩大和缩小，以帮助呼吸运动减弱或暂时停止呼吸的患者维持肺的通气功能。建立准确的呼吸系统力学模型，能够为呼吸机通气控制研究提供有效的模拟负载平台。

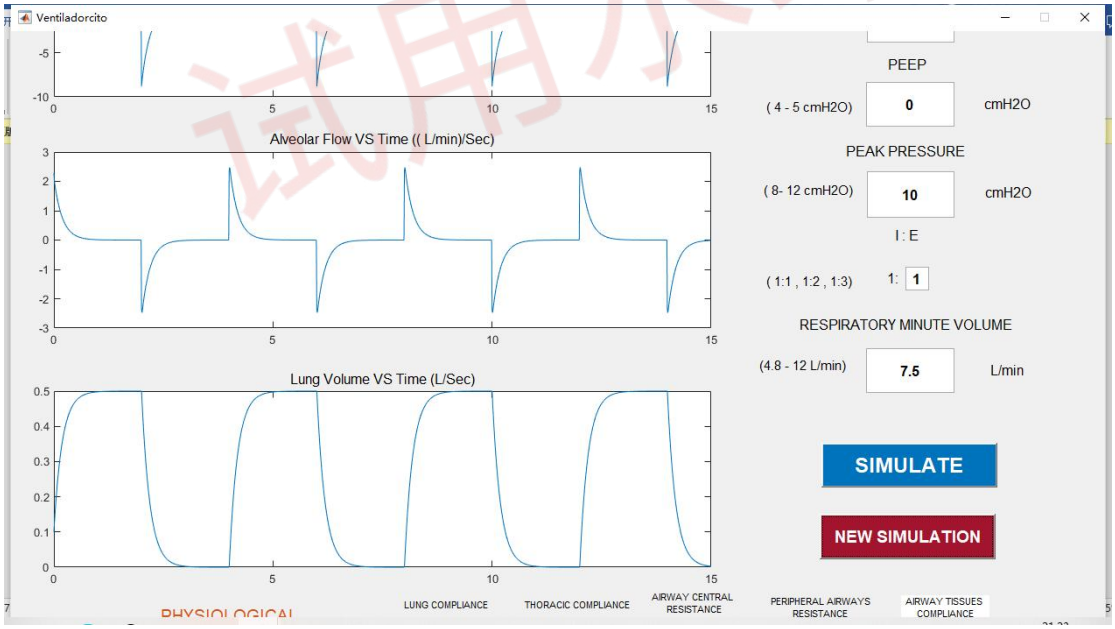
本实验基于 Simulink 设计了一个人类呼吸力学模型，该模型考虑了生理变量，使我们能够了解生物系统的行为并研究它在呼吸病理学面前的行为。此外，它还演示了如何通过改变不同的呼吸变量来改善呼吸系统健康。

2，实验内容

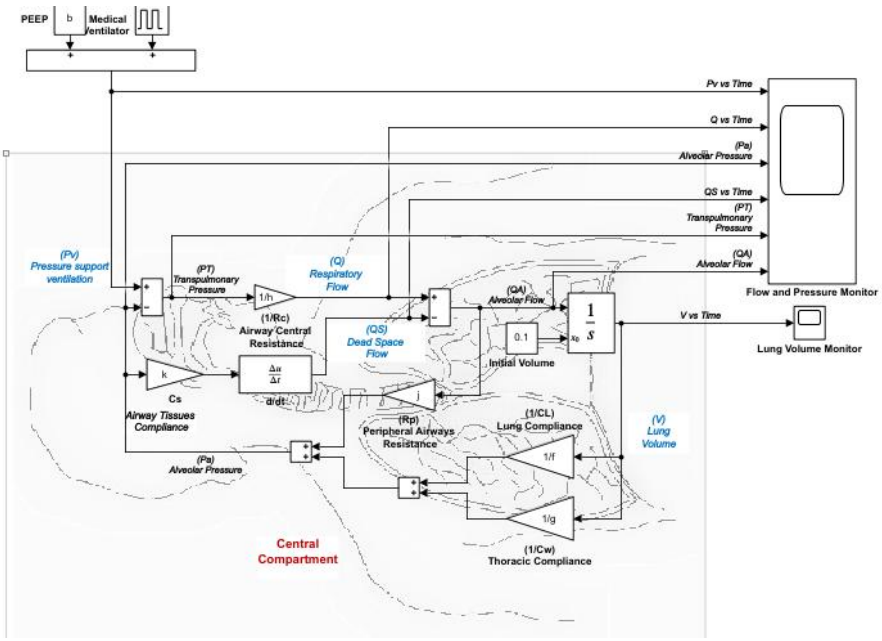
该模型在 Simulink 上实现，并通过使用 MATLAB 指南开发的图形用户界面（GUI）进行操作。

3，实验代码/流程细节

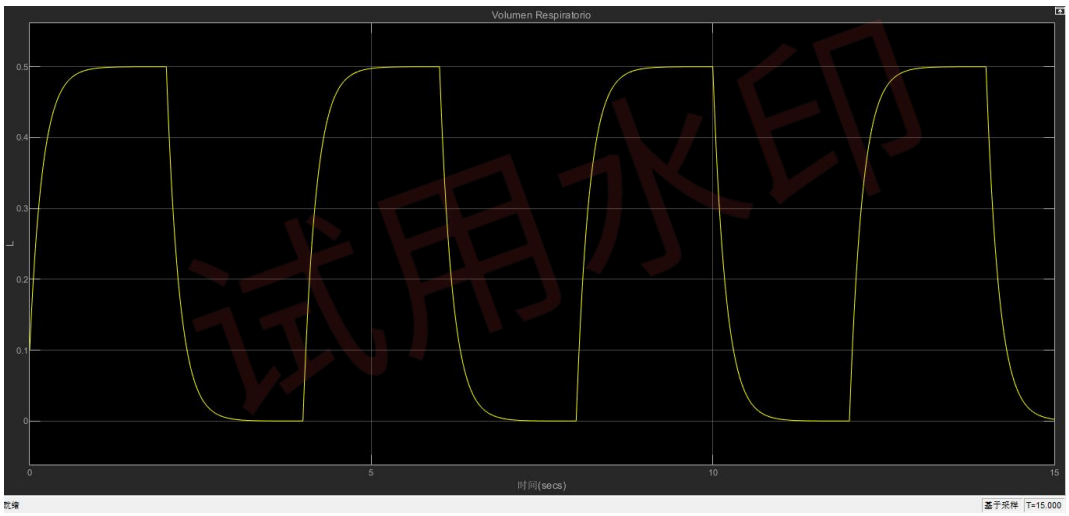
在 MATLAB 中，先打开文件夹中的“Ventiladorcito.m”文件，在得到的如下图形界面中点击“simulate”

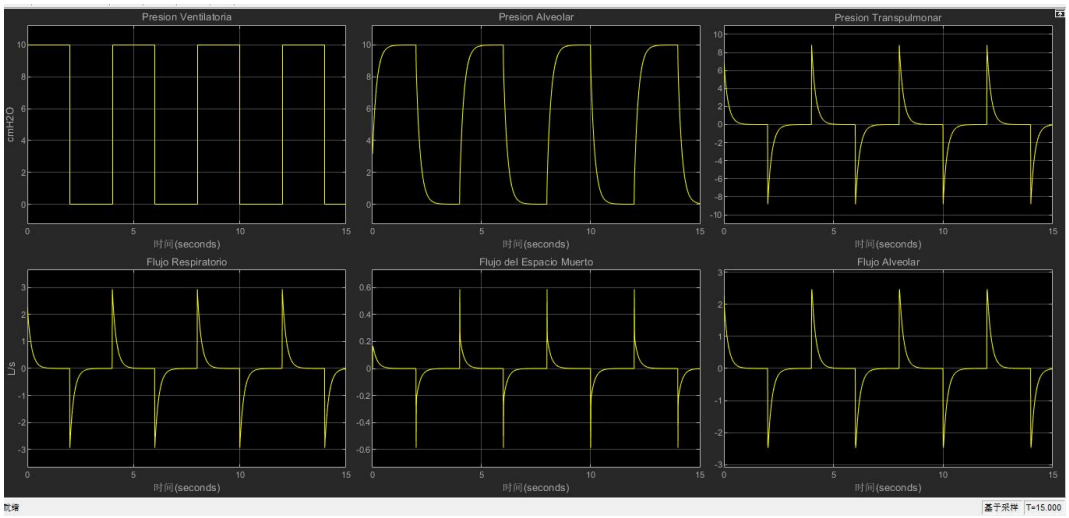


打开“Modelo_Respiratorio.slx”，可以看到 simulink 界面：



运行该 simulink 模型，得到如下结果：





分析：该系统演示了如何通过改变不同的呼吸变量来改善呼吸系统健康。