《系统仿真与 matlab》综合试题

题 目：药物在体内的分布与排除

编 号：4

姓 名 李星毅

班 级 自实 1701

学 号 U201712072

联系方式 13714231278

成 绩

目录

系统建模与仿真研究报告

李星毅 自实 1701 U201712072

2019.12.24

一. 问题模型介绍

药物在体内的分布和排除

药物进入机体后，在随血液输送到各器官和组织的过程中，不断地被吸收、分布、代谢，最终排出体外。药物在血液中的浓度（$\mu g / mv$）称血药浓度。血液浓度的大小直接影响到药物的疗效，浓度太低不能达到预期的治疗效果，浓度太高又可能导致中毒、副作用太强或造成浪费。因此研究药物在体内吸收、分布和排除的动态过程，对于新药研制时剂量的确定、给药方案设计等药理学和临床医学的发展具有重要的指导意义和实用价值。

为了研究目的，将一个机体划分成若干个房室，每个房室是机体的一部分，比如中心室和周边室。在一个房室内药物呈均匀分布，而在不同的房室之间按一定规律进行转移。如果要求的精度不是太高的情况下，可以只考虑一室模型。

模型假设

1. 药物进入机体后，全部进入中心室（血液较丰富的心、肺、肾等器官和组织），中心室的容积V 在给药过程中保持不变；
2. 药物从中心室排出体外，与排除的数量相比，药物的吸收可以忽略；
3. 药物排除的速率与中心室的血药浓度$c(\mathrm{t})$ 成正比，比例系数为k；
4. 给药速率为 $f(t)$，中心室药量为$x(t)$。

仿真要求 根据上述假设进行系统建模与仿真，系统输入为$x(0)$ 和 $f(\mathrm{t})$，比例系数k，中心室容积V。系统输出为t时刻的中心室血药浓度$c(\mathrm{t})$。要求有输入、输出界面及仿真过程。

二. 试题建模过程

1. 题目理解

根据平时所学的数学知识与课上所讲的建模方法，很容易理解题目的要求。即建立一个仅含中心室的一室模型，以时间为线，查看不同参数下，中心室的血药浓度的变化情况。步骤应该如下，首先根据自己建立的模型求解出一组能使系统稳定的系数。再用 Matlab 工具的一系列画图函数以及动画效果，实际模拟其数量的变化情况，在考虑实际情况的因素下，看仿真结果是否与实际情况相符，若相符则模型建立基本正确，若不成功就需要再改进模型。

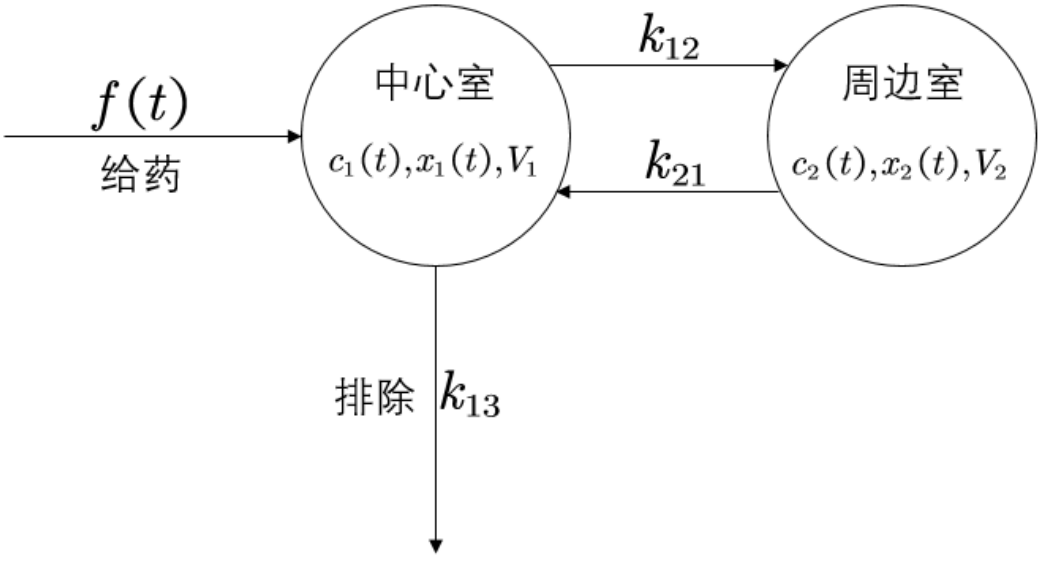
除此之外，通过查阅资料，为了更加精确地仿真，我将一个机体划分成两个房室，即将机体分为血液较丰富的中心室（包括心、肺、肾等器官）和血液较贫乏的周边室（如四肢、肌肉组织等）。两室模型的建立和求解方法可以推广到多室模型。

2. 做题目前的准备工作

在上课的时候，王小平老师已经讲过了一些实际模型的建立方法以及求解过程。王老师也花了一部分时间，给我们讲解了 Matlab 的基本用法以及 Matlab 的 GUI 工具。自己在上课的时候课下也基本都一一试过这些操作。再加上原先已经对于 Matlab 有过一定的了解，所以对于此题目入门还是很快的。

3. 模型建立

模型抽象 如下图所示。



规定参数含义 如下表所示。

仿真时间	T
药物种类	$medicine\_type$
中心室容积	V_1
周边室容积	V_2
中心室向周边室的药物转移速率	k_12
周边室向中心室的药物转移速率	k_21
药物从中心室向体外排除的速率系数	k_13
给药速率	f_t
给药时间	f_t_time
中心室初始药量	$x1(0)$
周边室初始药量	$x2(0)$
中心室血药浓度	$c1$
周边室血药浓度	$c2$

数学推导 根据假设条件和抽象模型可以写出两个房室满足的微分方程组。于是有

$$\frac{dx_1(t)}{dt} = -k_{12}c_1(t) - k_{13}c_1(t) + k_{21}c_2(t) + f(t)$$

$$\frac{dx_2(t)}{dt} = k_{12}c_1(t) - k_{21}c_2(t)$$

其中，药量$x_i(t)$与血药浓度$c_i(t)$、房室容积V_i之间显然满足关系式

$$V_i \times c_i(t) = x_i(t),\ i = 1, 2$$

将其带入上面的微分方程式子可得

$$\frac{dx_1(t)}{dt} = -\frac{k_{12}}{V_1}x_1(t) - \frac{k_{12}}{V_1}x_1(t) + \frac{k_{12}}{V_2}x_2(t) + f(t)$$

$$\frac{dx_2(t)}{dt} = \frac{k_{12}}{V_1}x_1(t) - \frac{k_{21}}{V_2}x_2(t)$$

或者写成关于浓度的微分方程组

$$\frac{dc_1(t)}{dt} = -\frac{k_{12}}{V_1}c_1(t) - \frac{k_{12}}{V_1}c_1(t) + \frac{k_{12}}{V_1}c_2(t) + \frac{1}{V_1}f(t)$$

$$\frac{dc_2(t)}{dt} = \frac{k_{12}}{V_2}c_1(t) - \frac{k_{21}}{V_2}c_2(t)$$

求解方式 既可以使用常微分方程组的的数学理论求出理论解，也可以采用数值分析中的迭代法求解微分方程的数值解，这里采用的是龙格库塔法求解数值解。

给药方式 考虑两种常见的给药方式。

1. 快速静脉注射

这种注射可简化为在$t=0$的瞬间将剂量D_0的药物输入中心室，血药浓度立即上升为$\frac{D_0}{V_1}$，于是有

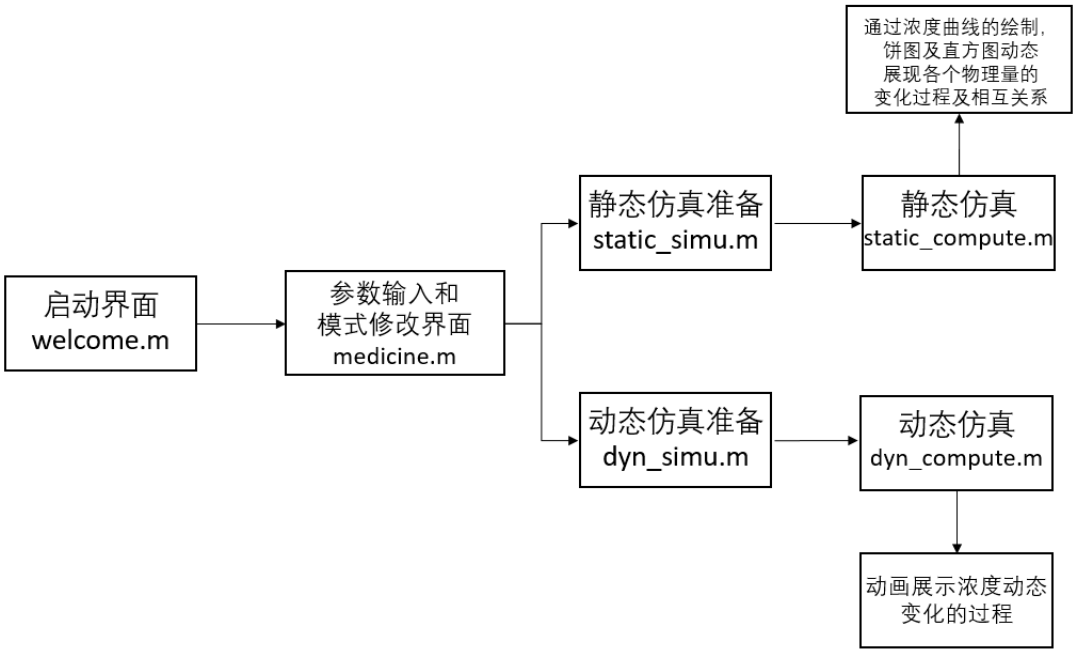
$$f(t)=0, c_1(0)=\frac{D_0}{V_1}, c_2(0)=0$$

2. 恒速静脉滴注

当静脉滴注的速率为常数k_0时，于是有

$$f(t)=k_0, c_1(0)=0, c_2(0)=0$$

三． 系统仿真流程



四． 系统仿真关键点

1. Matlab 仿真界面的设计

通过 GUIDE 工具进行界面的设计，会自行产生界面和 m 文件，同时生成回调函数。同时也可以用 uicontrol 辅助添加各种界面操作工具。同时对所有位置参数进行归一化处理，适应各种环境。然后使用 set 函数进行界面参数的读取和处理。在实际设计时要注意由 Matlab 指令运行延迟产生的 bug，比如界面的关闭。通过建立的模块的 callback 函数，实现仿真的流程。

2. 微分方程组处理

建立微分方程组，采用 Matlab 里的 ode45 函数求取数值解。所编写的 Matlab 代码如下。

```
% Procedure of performing ode45
if (f_t_time == 0 || f_t_time >= T)
   tspan = linspace(0, T, T);
   x_0 = [x1_0, x2_0, 0];
   [t, y] = ode45('odefunc', tspan, x_0);
else
   tspan1 = linspace(0, f_t_time, f_t_time);
   tspan2 = linspace(f_t_time, T, T-f_t_time);
   x_0 = [x1_0/V_1, x2_0/V_2, 0];

   [t1, y1] = ode45('odefunc', tspan1, x_0);

   x_f_t_time = [y1(end, 1), y1(end, 2), y1(end, 3)];
   [t2, y2] = ode45('odefunc', tspan2, x_f_t_time);

   t = [t1; t2];
   y = [y1; y2];
end
```

3. 仿真参数的输入

需要输入的仿真参数包括仿真时间T、药物种类$medicine_type$、中心室容积V_1、周边室容积V_2、中心室向周边室的药物转移速率k_12、周边室向中心室的药物转移速率k_21、药物从中心室向体外排除的速率系数k_13、给药速率f_t、给药时间f_t_time、中心室初始药量$x1(0)$、周边室初始药量$x2(0)$。我使用 Matlab 的 GUI 的可编辑文本实现了这一功能 ，将每一个可编辑文本中的 tag 选项变为我们想要的参数，通过使用键盘输入合适的数字，就可以将此数字赋给 tag 的对应的参数，这样就可以通过键盘输入完成参数的输入。

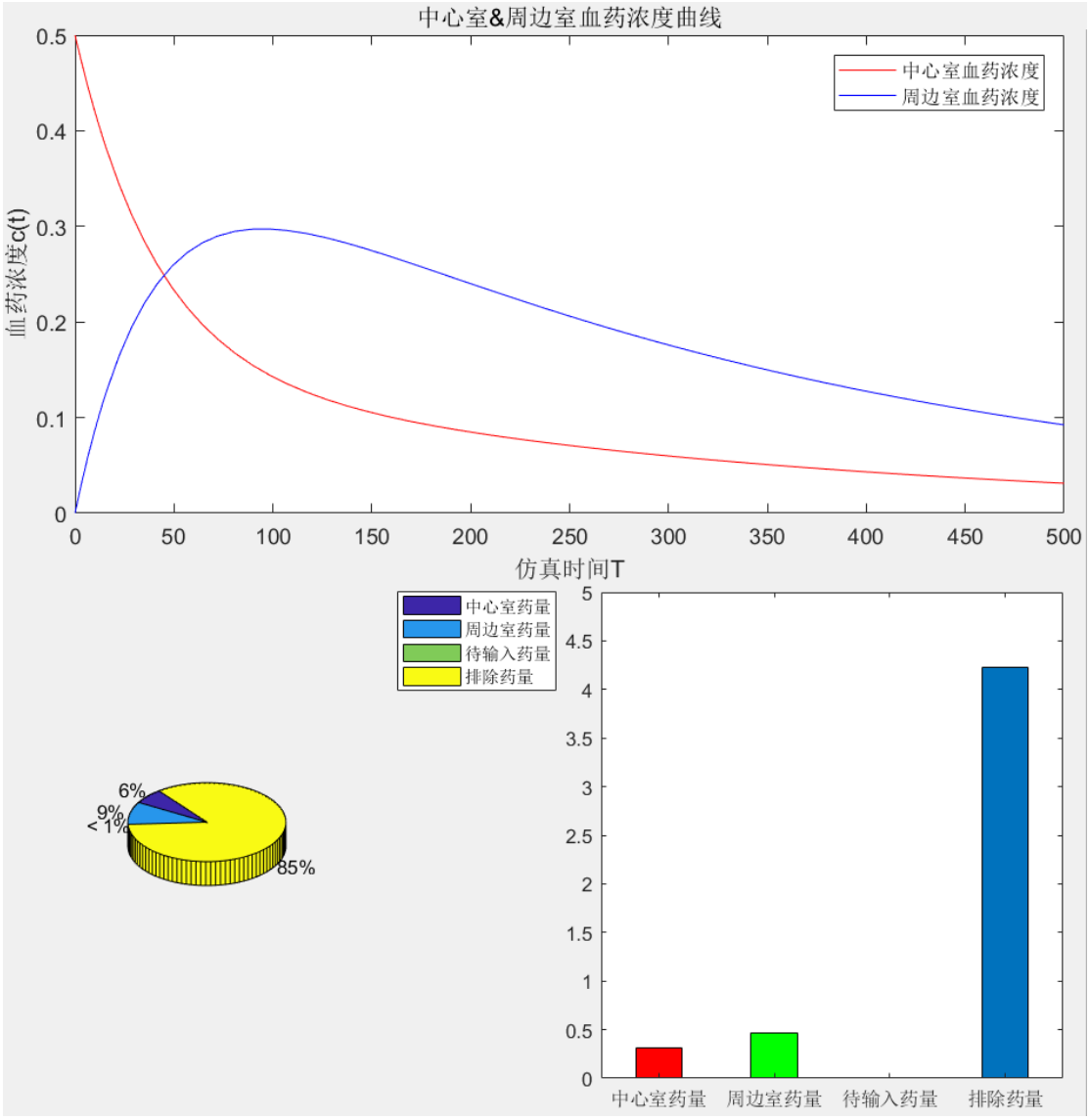
4. 仿真模式的选择

仿真模式有两种选择，一是静态仿真，二是动态仿真。两者的切换我使用的是 Matlab GUI 中的按钮完成的这一功能。当点击“静态仿真”按钮时，触发静态仿真的 callback 函数，读取输入参数，进入静态仿真；当点击“动态仿真”按钮时，触发动态仿真的 callback 函数，读取输入参数，进入动态仿真。

5. 静态仿真

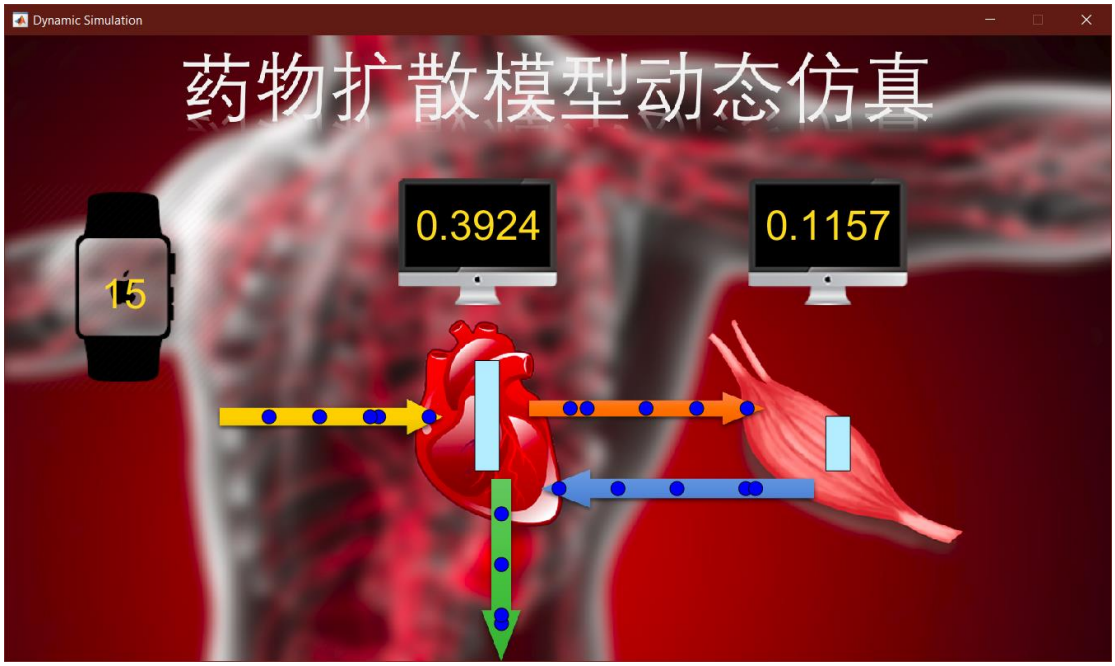
通过浓度曲线的绘制、饼图和直方图展现血药浓度随时间的变化以及相互之

间的大小关系。



6. 动态仿真

通过药物运动和浓度的数字表示来形象地展示浓度的变化过程。



五． 程序运行指南

1. 关于文件及内容

该药物扩散仿真系统共 7 个 m 文件：welcome.m、odefunc.m、medicine.m、static_simu.m、static_compute.m、dyn_simu.m、dyn_compute.m。并有 1 个 fig 文件：medicine.fig。还有 3 张图片 start2.jpg、welcome.png、dyn_simu_bg2.png。

welcome.m 文件：系统仿真入口，运行后进入欢迎界面，其中也有介绍界面以及作者简介界面。
odefunc.m 文件：微分方程函数 ode45 的辅助函数，用于输入微分方程组和做一些判断、准备工作
medicine.m 文件：仿真系统的主界面，包括系统参数设置界面，用于各种参数输入、仿真模式选择、以及选择仿真的对象等；
static_simu.m 文件：为静态仿真做准备，读取用户输入的参数。
static_compute.m 文件：用于计算并仿真中心室和周边室浓度的变化，数据分析以及动态曲线、饼图和直方图绘制。
dyn_simu.m 文件：为动态仿真做准备，读取用户输入的参数。
dyn_compute.m 文件：用于动态显示中心室和周边室浓度的变化情况，来实时仿真中心室和周边室浓度变化以及药物扩散速度。

2. 系统简介

在命令行窗口输入 welcome 开始运行程序，进入欢迎界面，在欢迎界面可以看到作者信息。右下角有一个按钮，点击后自动进入仿真界面。
进入仿真主界面中，有仿真参数的输入以及仿真对象以及仿真模式等的选择，设置好所有的参数后（参数一定要合理否则会报错，而且不能为空），点选不同的按钮来选择静态仿真或者动态仿真。

六． 系统模型的亮点以及新颖的地方

1. 题目要求中没有考虑周边室，但是我考虑了，建立了一个应用更为广泛，更为精确的两室模型。
2. 题目没有要求动态展现浓度的变化，但是我为了更好的展现药物扩散、排除的过程，精心设计了动态画曲线和饼图、直方图的动态变化这一过程。
3. 题目中没有要求动态仿真，我建立了一个动态仿真的模型，可以实时反映浓度的动态变化，以及药物扩散速度。
4. 可以自己定义给药方式以及给药速度，满足了用户好奇心。
5. 这个模型的考虑还是比较全面的，例如参数的正确性在仿真之前都已经通过了验证，已经成为合适的参数后，才能进行仿真，否则系统会报错而且会弹出对话框以提醒。
6. 多界面形式，不是单一界面，所以要考虑不同界面转换的情况，更要考虑参数传递的正确性、准确性以及稳定性。
7. 所有代码以及界面全部自己独立完成。

七． 程序运行实例分析

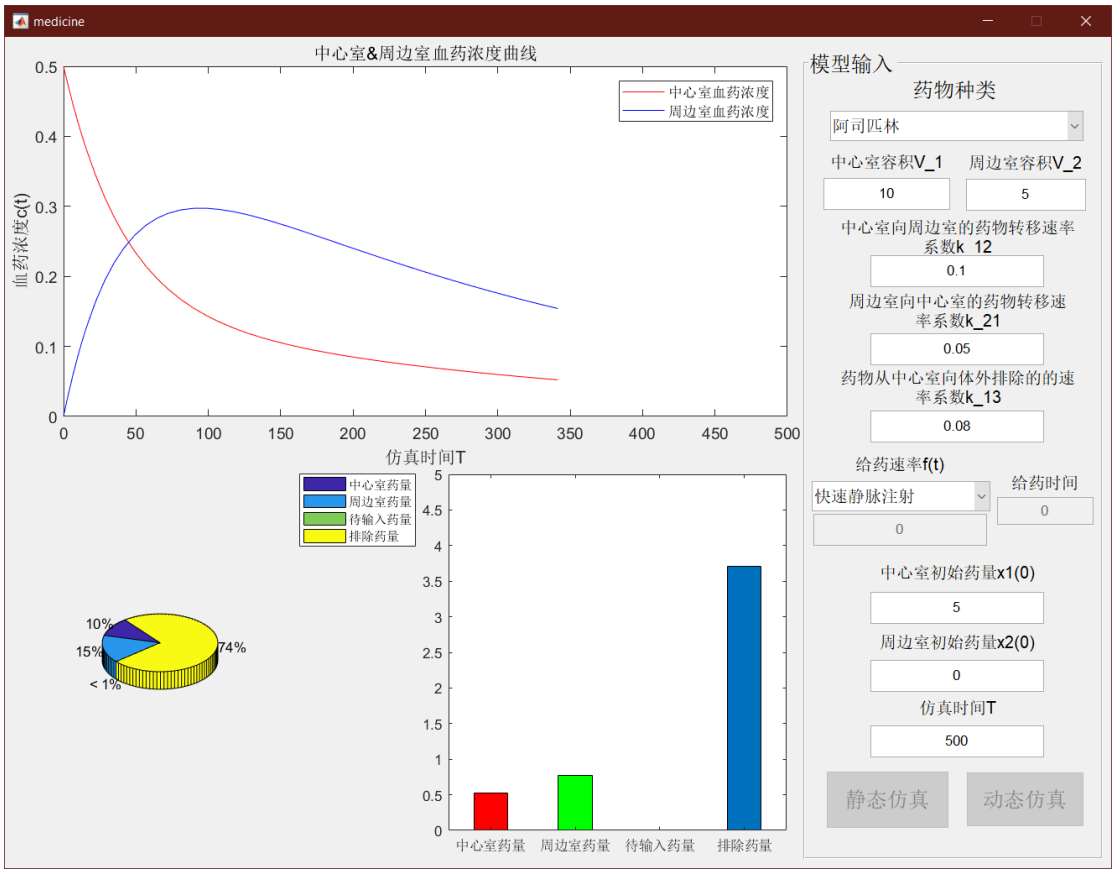
下面介绍仿真程序运行过程，并截取仿真过程中的部分图片。
在命令行窗口输入 welcome 开始运行程序，进入欢迎界面。

点击右下角箭头按钮，进入参数设定界面，并自动关闭欢迎界面。

输入合理的参数，如参数不符合要求，则会警告（举一个不符合的例子，这样的约束不止一个）

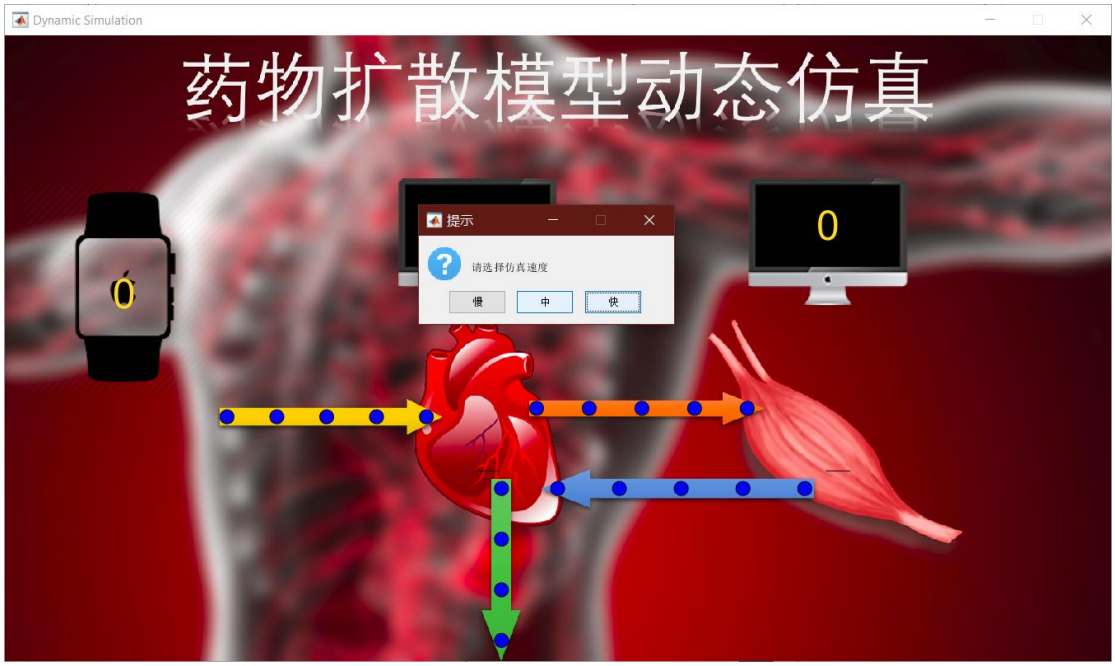
输入合理的仿真时间，系统设定的仿真时间为 50-10000，小于 50 和大于 10000 都会出现报错信息

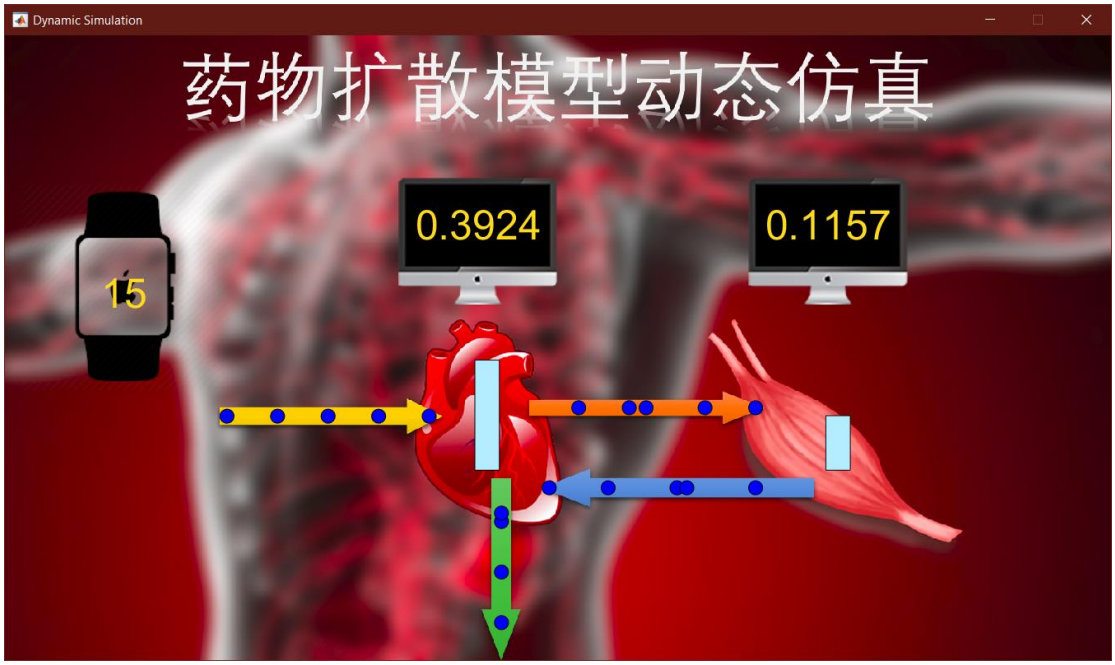
在正确的参数输入的情况下（假设为默认参数），点击静态仿真按钮。



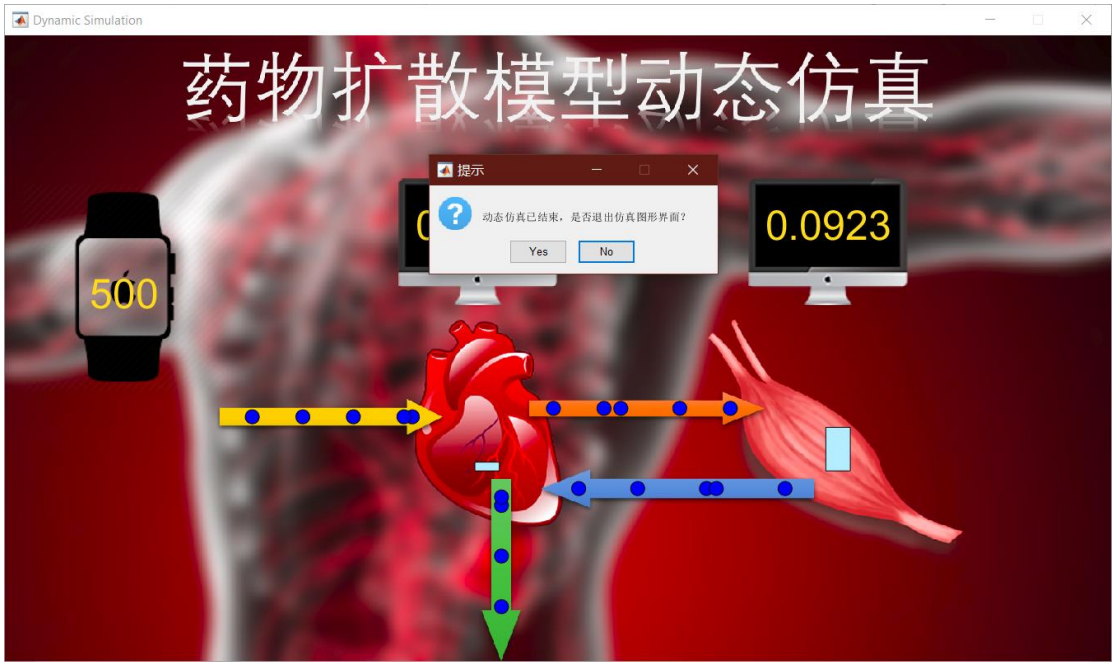
在正确的参数输入的情况下（假设为默认参数），点击动态仿真按钮，会弹出选

择仿真速度的对话框。

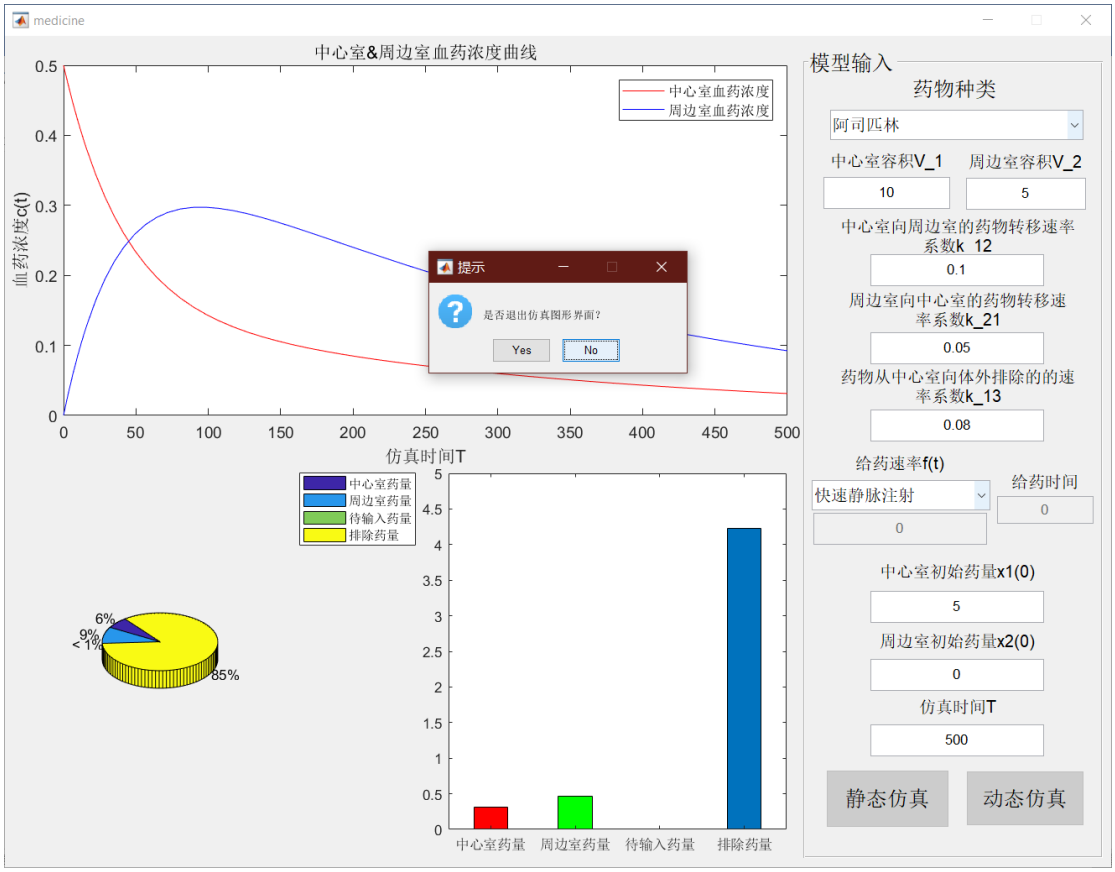




动态仿真结束时，会弹出对话框提示仿真结束。



关闭仿真系统时，会弹出对话框确认是否真的关闭系统。



八. 心得体会

由于之前已经对于这个题目有过一定的了解，而且上课的时候也跟着老师学习了系统建模以及 Matlab 的知识，所以这个题目做起来还是比较快的。但是其中的计算稳定性是花了我很多很多的时间，为了找到一组十分合理的参数，我也查阅了相关的书籍，在网上也找到了很多的资料，通过努力自己也成功的解决了问题。

这个课程的本质是让我们使用 Matlab 来解决一个实际问题，所以建立模型的稳定性、正确性与 Matlab 的仿真界面同等重要。作为一个毫无艺术细胞的人，自己在界面设计以及系统仿真流程这方面也是下了很大的功夫。尽量做到了结构清晰简洁大方，不刻意追求花里胡哨的外表。将模型和 Matlab 很好的融合在了一起。

自己在开始学习这门课程时，更加深入的学写了 Matlab。做完这个课程设计以后，我对于使用 Matlab 进行系统的仿真也有了更深一层的理解。

我觉得使用 Matlab 制作一个仿真系统是一个既简单又复杂的事。简单在于 Matlab 强大的功能，如此多的库函数已经方便的图形化界面设计。而难是因为首先要进行模型的建立，从一个理想化的模型，到越来越接近实际的一个真实的模型，我还没有达到能够完全抽象一个模型的能力。

在进行这个课程设计的过程中，我对于 Matlab 图形化界面的设计和分析问题解决问题的能力已经有了一定的进步。也已经可以使用 Matlab 的 GUI 功能搭建基于图形化界面的小系统。

但是这并不是终点，Matlab 还有更多可以学习的地方等着我去学习，对于系统模型的建立以及建立后的仿真，自己也应该再下功夫去深究。自己也应该带着学习 Matlab 的态度以及做本课程设计的态度去进行接下来的工作与学习。最后还是应该感谢帮助我完成课程设计的老师和学长、学姐和同学们。

九. 参考书籍及资料

1.《系统建模与仿真》 齐欢 王小平 编著 清华大学出版社
2.《MATLAB 程序设计与应用（第二版）》 刘卫国 编著 高等教育出版社
3.《MATLAB 实用教程》 苏金明 阮沈勇 编著 电子工业出版社

十. 代码附录

1. welcome.m

```
% Caution: please switch the encoding to GB2312
%
% Created by Xingyi Li on Dec. 24, 2019.
% Copyright (c) 2019 Huazhong University of Science and Technology.
% All rights reserved.
```

```
% Clear everything before run the medicine simulation system
clear;
close all;
clc;

% Create a background figure
start_background = figure('Units', 'normalized', ...
                                    'NumberTitle','off', ...
                                    'Resize', 'off', ...
                                    'Position', [0.15, 0.15, 0.661, 0.7], ...
                                    'Name', 'Welcome', ...
                                    'Menu', 'None', ...
                                    'Color', 'Black',...
                                    'CloseRequestFcn', @my_closereq);

% Within the background figure, generate an axis, then place an image
ha = axes('Parent', start_background, 'Units', 'normalized',...
            'Position', [0, 0, 1, 1]);
img = imread('welcome.png');
imshow(img, 'Parent', ha);

% Create a start-up button
next = false;
start = imread('start2.jpg');
start_button = uicontrol('Style','pushbutton',...
                                    'Units', 'normalized', ...
                                    'Callback', 'delete(start_background);next = true;', ...
                                    'Position',[0.94 0 0.06 0.09], ...
                                    'FontSize', 25, ...
                                    'ForegroundColor', 'r' ,...
                                    'CData', start);

% Wait for user's response
while 1
    if ~ishandle(start_background)
        if next == true
            clear;
            medicine;
            return;
        else
            clear;
            return;
        end
    end
```

```
    pause(0.05);
end


function my_closereq(src, callbackdata)
    %MY_CLOSEREQ offers a self-defined close request to display a question dialog box
    %
    % Created by Xingyi Li on Dec. 24, 2019.
    % Copyright (c) 2019 Huazhong University of Science and Technology.
    % All rights reserved.

    selection = questdlg('是否退出仿真图形界面？', ...
                         '提示', ...
                         'Yes','No','No');
    switch selection
        case 'Yes'
            delete(gcf);
        case 'No'
            return
    end
end
```

2. medicine.m

```
% Caution: please switch the encoding to GB2312

function varargout = medicine(varargin)
% MEDICINE MATLAB code for medicine.fig
%      MEDICINE, by itself, creates a new MEDICINE or raises the existing
%      singleton*.
%
%      H = MEDICINE returns the handle to a new MEDICINE or the handle to
%      the existing singleton*.
%
%      MEDICINE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MEDICINE.M with the given input arguments.
%
%      MEDICINE('Property','Value',...) creates a new MEDICINE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before medicine_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to medicine_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
```

```
%           instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help medicine

% Last Modified by GUIDE v2.5 24-Dec-2019 11:31:19

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',             mfilename, ...
                           'gui_Singleton',    gui_Singleton, ...
                           'gui_OpeningFcn', @medicine_OpeningFcn, ...
                           'gui_OutputFcn',    @medicine_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
% End initialization code - DO NOT EDIT


% --- Executes just before medicine is made visible.
function medicine_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject       handle to figure
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)
% varargin      command line arguments to medicine (see VARARGIN)

% Choose default command line output for medicine
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes medicine wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = medicine_OutputFcn(hObject, eventdata, handles)
% varargout   cell array for returning output args (see VARARGOUT);
% hObject     handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function V_1_Callback(hObject, eventdata, handles)
% hObject     handle to V_1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of V_1 as text
%        str2double(get(hObject,'String')) returns contents of V_1 as a double


% --- Executes during object creation, after setting all properties.
function V_1_CreateFcn(hObject, eventdata, handles)
% hObject     handle to V_1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function V_2_Callback(hObject, eventdata, handles)
% hObject     handle to V_2 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of V_2 as text
```

```
%        str2double(get(hObject,'String')) returns contents of V_2 as a double


% --- Executes during object creation, after setting all properties.
function V_2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to V_2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function k_12_Callback(hObject, eventdata, handles)
% hObject    handle to k_12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of k_12 as text
%        str2double(get(hObject,'String')) returns contents of k_12 as a double


% --- Executes during object creation, after setting all properties.
function k_12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to k_12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function k_21_Callback(hObject, eventdata, handles)
% hObject    handle to k_21 (see GCBO)
```

```
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of k_21 as text
%        str2double(get(hObject,'String')) returns contents of k_21 as a double


% --- Executes during object creation, after setting all properties.
function k_21_CreateFcn(hObject, eventdata, handles)
% hObject       handle to k_21 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function x2_0_Callback(hObject, eventdata, handles)
% hObject       handle to x2_0 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of x2_0 as text
%        str2double(get(hObject,'String')) returns contents of x2_0 as a double


% --- Executes during object creation, after setting all properties.
function x2_0_CreateFcn(hObject, eventdata, handles)
% hObject       handle to x2_0 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function k_13_Callback(hObject, eventdata, handles)
% hObject    handle to k_13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of k_13 as text
%        str2double(get(hObject,'String')) returns contents of k_13 as a double


% --- Executes during object creation, after setting all properties.
function k_13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to k_13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function x1_0_Callback(hObject, eventdata, handles)
% hObject    handle to x1_0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of x1_0 as text
%        str2double(get(hObject,'String')) returns contents of x1_0 as a double


% --- Executes during object creation, after setting all properties.
function x1_0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to x1_0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
```

```
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in f_t_type.
function f_t_type_Callback(hObject, eventdata, handles)
% hObject    handle to f_t_type (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns f_t_type contents as cell array
%        contents{get(hObject,'Value')} returns selected item from f_t_type
switch get(hObject,'Value')
    case 1
        set(handles.f_t, 'Enable', 'off', 'String', '0');
        set(handles.f_t_time, 'Enable', 'off', 'String', '0');
        set(handles.x1_0, 'String', '5');
    case 2
        set(handles.f_t, 'Enable', 'on', 'String', '0.05');
        set(handles.f_t_time, 'Enable', 'on', 'String', '50');
        set(handles.x1_0, 'String', '0');
    case 3
        set(handles.f_t, 'Enable', 'on', 'String', '1/(t+3)');
        set(handles.f_t_time, 'Enable', 'on', 'String', '50');
        set(handles.x1_0, 'String', '0');
end


% --- Executes during object creation, after setting all properties.
function f_t_type_CreateFcn(hObject, eventdata, handles)
% hObject    handle to f_t_type (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in static_simu.
function static_simu_Callback(hObject, eventdata, handles)
% hObject    handle to static_simu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
static_simu(handles);


function f_t_Callback(hObject, eventdata, handles)
% hObject    handle to f_t (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of f_t as text
%        str2double(get(hObject,'String')) returns contents of f_t as a double


% --- Executes during object creation, after setting all properties.
function f_t_CreateFcn(hObject, eventdata, handles)
% hObject    handle to f_t (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in medicine_type_other.
function medicine_type_Callback(hObject, eventdata, handles)
% hObject    handle to medicine_type (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns medicine_type contents as cell array
%        contents{get(hObject,'Value')} returns selected item from medicine_type


% --- Executes during object creation, after setting all properties.
function medicine_type_CreateFcn(hObject, eventdata, handles)
% hObject    handle to medicine_type (see GCBO)
```

```
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function medicine_type_text_CreateFcn(hObject, eventdata, handles)
% hObject      handle to medicine_type_text (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called


function simu_time_Callback(hObject, eventdata, handles)
% hObject      handle to simu_time (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of simu_time as text
%        str2double(get(hObject,'String')) returns contents of simu_time as a double


% --- Executes during object creation, after setting all properties.
function simu_time_CreateFcn(hObject, eventdata, handles)
% hObject      handle to simu_time (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function curves_diagram_CreateFcn(hObject, eventdata, handles)
% hObject      handle to curves_diagram (see GCBO)
```

```
% eventdata    reserved - to be defined] in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate curves_diagram


% --- Executes on button press in dyn_simu.
function dyn_simu_Callback(hObject, eventdata, handles)
% hObject       handle to dyn_simu (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)
dyn_simu(handles);


function f_t_time_Callback(hObject, eventdata, handles)
% hObject       handle to f_t_time (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of f_t_time as text
%              str2double(get(hObject,'String')) returns contents of f_t_time as a double


% --- Executes during object creation, after setting all properties.
function f_t_time_CreateFcn(hObject, eventdata, handles)
% hObject       handle to f_t_time (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if                  ispc                &&                  isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function pie_diagram_CreateFcn(hObject, eventdata, handles)
% hObject       handle to pie_diagram (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles       empty - handles not +--created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate pie_diagram
```

```
% --- Executes during object creation, after setting all properties.
function hist_diagram_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hist_diagram (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate hist_diagram


% --- Executes during object creation, after setting all properties.
function static_simu_CreateFcn(hObject, eventdata, handles)
% hObject    handle to static_simu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: delete(hObject) closes the figure
selection = questdlg('是否退出仿真图形界面？', ...
                                    '提示', ...
                                    'Yes','No','No');
switch selection
    case 'Yes'
        delete(hObject);
    case 'No'
        return
end
```

3. odefunc.m

```
% Caution: please switch the encoding to GB2312

function dx = odefunc(t, x)
    %ODEFUNC prepares for ode45.
    %
    % Created by Xingyi Li on Dec. 24, 2019.
    % Copyright (c) 2019 Huazhong University of Science and Technology.
    % All rights reserved.
```

```
    % Some global varibles, whose meanings can be easily guessed literally
    global k_12 k_13 k_21 V_1 V_2 f_t f_t_time;

    % Type in the ordinary differential equations
    dx = zeros(3, 1);
    try
        if (t <= f_t_time)
            dx(1) = -(k_12 + k_13)/V_1*x(1) + k_21/V_2*x(2) + f_t;
        else
            dx(1) = -(k_12 + k_13)/V_1*x(1) + k_21/V_2*x(2);
        end
    catch
        if (t <= f_t_time)
            dx(1) = -(k_12 + k_13)/V_1*x(1) + k_21/V_2*x(2) + f_t(t);
        else
            dx(1) = -(k_12 + k_13)/V_1*x(1) + k_21/V_2*x(2);
        end
    end
    dx(2) = k_12/V_1*x(1) - k_21/V_2*x(2);
    dx(3) = k_13/V_1*x(1);
end
```

4. static_simu.m

```
% Caution: please switch the encoding to GB2312

function static_simu(handles)
    %
    %STATIC_SIMU performs preliminary steps for static simulation.
    %
    % Created by Xingyi Li on Dec. 24, 2019.
    % Copyright (c) 2019 Huazhong University of Science and Technology.
    % All rights reserved.

    % Some global varibles, whose meanings can be easily guessed literally
    global k_12 k_13 k_21 V_1 V_2 f_t x1_0 x2_0 f_t_type T medicine_type f_t_time;

    % The following codes are trying to acquire corresponding values from GUI panel
    k_12 = str2double(get(handles.k_12, 'String'));
    k_13 = str2double(get(handles.k_13, 'String'));
    k_21 = str2double(get(handles.k_21, 'String'));
    V_1 = str2double(get(handles.V_1, 'String'));
    V_2 = str2double(get(handles.V_2, 'String'));
    x1_0 = str2double(get(handles.x1_0, 'String'));
    x2_0 = str2double(get(handles.x2_0, 'String'));
```

```
    f_t_type = get(handles.f_t_type, 'Value');
    T = str2double(get(handles.simu_time, 'String'));
    medicine_type = get(handles.medicine_type, 'Value');
    f_t_time = str2double(get(handles.f_t_time, 'String'));

    % Respond according to the f(t) type
    switch f_t_type
        % If constant function
        case {1, 2}
            % In these cases, both are actually constant
            f_t = str2double(get(handles.f_t, 'String'));
            if (isnan(f_t))
                msgbox('参数不能为空', 'Error','error');
            elseif (f_t < 0)
                msgbox('常数函数不能为负数', 'Error','error');
            end
        % If non-constant function
        case 3
            % Catch the function that user inputs
            s = get(handles.f_t, 'String');
            s = strrep(s, '*', '.*');
            s = strrep(s, '/', './');
            s = strrep(s, '^', '.^');
            f_t = str2func(['@(t)', s]);
            if ~isa(f_t, 'function_handle')
                msgbox('函数输入有误（自变量需关于 t）', 'Error','error');
            end
    end

    % Assert the validity of the input parameters
    if (isnan(x1_0) || isnan(x2_0) || isnan(k_12) || isnan(k_21) || isnan(k_13) || isnan(V_1) ||
isnan(V_2) || isnan(T) || isnan(f_t_time))
        msgbox('参数不能为空', 'Error','error');
    elseif (T > 10000 || T < 50)
        msgbox('仿真时间请设定在 50-10000 之间', 'Error','error');
    elseif (x1_0 < 0 || x2_0 < 0 || k_12 < 0 || k_21 < 0 || k_13 < 0 || V_1 < 0 || V_2 < 0)
        msgbox('参数不可为负数', 'Error','error');
    else
        static_compute(handles);
    end
end
```

5. static_compute.m

```
% Caution: please switch the encoding to GB2312
```

```
function static_compute(handles)
    %STATIC_COMPUTE performs the main procedure of static simulation.
    %
    % Created by Xingyi Li on Dec. 24, 2019.
    % Copyright (c) 2019 Huazhong University of Science and Technology.
    % All rights reserved.

    % Some global varibles, whose meanings can be easily guessed literally
    global x1_0 x2_0 T V_1 V_2 f_t f_t_time;

    % Procedure of performing ode45
    if (f_t_time == 0 || f_t_time >= T)
        tspan = [0, T];
        x_0 = [x1_0, x2_0, 0];
        [t, y] = ode45('odefunc', tspan, x_0);

    else
        tspan1 = [0, f_t_time];
        tspan2 = [f_t_time, T];
        x_0 = [x1_0/V_1, x2_0/V_2, 0];

        [t1, y1] = ode45('odefunc', tspan1, x_0);

        x_f_t_time = [y1(end, 1), y1(end, 2), y1(end, 3)];
        [t2, y2] = ode45('odefunc', tspan2, x_f_t_time);

        t = [t1; t2];
        y = [y1; y2];

    end

    % Calculate the total medicine volume
    if isa(f_t, 'function_handle')
        total_med_vol = integral(f_t, 0, f_t_time) + x1_0 + x2_0;
    else
        total_med_vol = f_t*f_t_time + x1_0 + x2_0;
    end

    % Disable the static simulation button as well as the dynamic simulation button
    set(handles.static_simu, 'Enable', 'off');
    set(handles.dyn_simu, 'Enable', 'off');

    xlabel(handles.curves_diagram, '仿真时间 T');
```

```
    ylabel(handles.curves_diagram, '血药浓度 c(t)');
    title(handles.curves_diagram, '中心室&周边室血药浓度曲线');

    % Number of frames
    n_frames = size(t, 1);
    for k = 1:n_frames
        % Compute the input medicine volume
        if k < f_t_time
            if isa(f_t, 'function_handle')
                input_med_vol = integral(f_t, k, f_t_time);
            else
                input_med_vol = f_t*(f_t_time-k);
            end
        else
            input_med_vol = 0;
        end

        % Plot the concentration curves
        hold on;
        cla(handles.curves_diagram);
        plot(handles.curves_diagram, t(1:k), y(1:k, 1)/V_1, 'Color', 'r');
        plot(handles.curves_diagram, t(1:k), y(1:k, 2)/V_2, 'Color', 'b');
        axis(handles.curves_diagram, [0, max(t), 0, max(max([y(:, 1)/V_1, y(:, 2)/V_2]))]);
        legend(handles.curves_diagram, '中心室血药浓度', '周边室血药浓度');

        % Plot the pie diagram
        cla(handles.pie_diagram);
        pie3(handles.pie_diagram, [max(y(k, 1), 1e-6), max(y(k, 2), 1e-6), ...
            max(input_med_vol, 1e-6), ...
            max(y(k, 3), 1e-6)]);
        legend(handles.pie_diagram, '中心室药量', '周边室药量', '待输入药量', '排除药量
');

        % Plot the histogram
        cla(handles.hist_diagram);
        b = bar(handles.hist_diagram, ...
            categorical({'中心室药量', '周边室药量', '待输入药量', '排除药量'}), ...
            [max(y(k, 1), 1e-6), max(y(k, 2), 1e-6), ...
            max(input_med_vol, 1e-6), ...
            max(y(k, 3), 1e-6)],...
            0.4);
        b.FaceColor = 'flat';
        b.CData(1, :) = [1, 0, 0];
        b.CData(2, :) = [0, 1, 0];
```

```
            b.CData(3, :) = [0, 0, 1];
            ymax = max(max(y));
            set(handles.hist_diagram, 'YLim', [0, max([ymax, ...
                total_med_vol])]);

            pause(0.001);
        end
        % Enable the static simulation button as well as the dynamic simulation button
        set(handles.static_simu, 'Enable', 'on');
        set(handles.dyn_simu, 'Enable', 'on');
end
```

6. dyn_simu.m

```
% Caution: please switch the encoding to GB2312

function dyn_simu(handles)
        %DYN_SIMU Performs preliminary steps for dynamic simulation.
        %
        % Created by Xingyi Li on Dec. 24, 2019.
        % Copyright (c) 2019 Huazhong University of Science and Technology.
        % All rights reserved.

        % Some global varibles, whose meanings can be easily guessed literally
        global k_12 k_13 k_21 V_1 V_2 f_t x1_0 x2_0 f_t_type T medicine_type f_t_time;

        % The following codes are trying to acquire corresponding values from GUI panel
        k_12 = str2double(get(handles.k_12, 'String'));
        k_13 = str2double(get(handles.k_13, 'String'));
        k_21 = str2double(get(handles.k_21, 'String'));
        V_1 = str2double(get(handles.V_1, 'String'));
        V_2 = str2double(get(handles.V_2, 'String'));
        x1_0 = str2double(get(handles.x1_0, 'String'));
        x2_0 = str2double(get(handles.x2_0, 'String'));
        f_t_type = get(handles.f_t_type, 'Value');
        T = str2double(get(handles.simu_time, 'String'));
        medicine_type = get(handles.medicine_type, 'Value');
        f_t_time = str2double(get(handles.f_t_time, 'String'));

        % Respond according to the f(t) type
        switch f_t_type
            % If constant function
            case {1, 2}
                % In these cases, both are actually constant
                f_t = str2double(get(handles.f_t, 'String'));
```

```
            if (isnan(f_t))
                msgbox('参数不能为空', 'Error','error');
            elseif (f_t < 0)
                msgbox('常数函数不能为负数', 'Error','error');
            end
        % If non-constant function
        case 3
            % Catch the function that user inputs
            s = get(handles.f_t, 'String');
            s = strrep(s, '*', '.*');
            s = strrep(s, '/', './');
            s = strrep(s, '^', '.^');
            f_t = str2func(['@(t)', s]);
            if ~isa(f_t, 'function_handle')
                msgbox('函数输入有误（自变量需关于 t）', 'Error','error');
            end
    end

    % Assert the validity of the input parameters
    if (isnan(x1_0) || isnan(x2_0) || isnan(k_12) || isnan(k_21) || isnan(k_13) || isnan(V_1) ||
isnan(V_2) || isnan(T) || isnan(f_t_time))
        msgbox('参数不能为空', 'Error','error');
    elseif (T > 1000 || T < 10)
        msgbox('仿真时间请设定在 10-1000 之间', 'Error','error');
    elseif (x1_0 < 0 || x2_0 < 0 || k_12 < 0 || k_21 < 0 || k_13 < 0 || V_1 < 0 || V_2 < 0)
        msgbox('参数不可为负数', 'Error','error');
    else
        dyn_compute(handles);
    end
end

7. dyn_compute.m
% Caution: please switch the encoding to GB2312

function dyn_compute(handles)
    %DYN_COMPUTE performs the main procedure of dynamic simulation.
    %
    % Created by Xingyi Li on Dec. 24, 2019.
    % Copyright (c) 2019 Huazhong University of Science and Technology.
    % All rights reserved.

    % Some global varibles, whose meanings can be easily guessed literally
    global x1_0 x2_0 T V_1 V_2 f_t_time k_12 medicine_type;
```

```
% Disable the static simulation button as well as the dynamic simulation button
set(handles.static_simu, 'Enable', 'off');
set(handles.dyn_simu, 'Enable', 'off');

% Create a background figure
dyn_simu_background = figure('Units', 'normalized', ...
                                        'NumberTitle','off', ...
                                        'Resize', 'off', ...
                                        'Position', [0.15, 0.15, 0.7, 0.7032], ...
                                        'Name', 'Dynamic Simulation', ...
                                        'Menu', 'None', ...
                                        'Color', [0.38, 0, 0.0235],...
                                        'CloseRequestFcn', @my_closereq);

% Within the background figure, generate an axis, then place an image
ha = axes('Parent', dyn_simu_background, 'Units', 'normalized',...
                'Position', [0, 0, 1, 1]);
img = imread('dyn_simu_bg2.png');
imshow(img);

% Below are the process of constructing some text objects, with the aim of displaying
% the concentrations of those medicine in both center room and the nearby room
center_concentration_display = text('Parent', ha, 'Units', 'Normalized',...
                                                'String', '0', 'LineWidth' , 1,    ...
                                                'Position', [0.428 0.7],    ...
                                                'Color', [1 0.86 0.109], 'FontSize', 30,...
                                                'HorizontalAlignment', 'center');

nearby_concentration_display = text('Parent', ha, 'Units', 'Normalized',...
                                                'String', '0', 'LineWidth' , 1,    ...
                                                'Position', [0.744 0.7],    ...
                                                'Color', [1 0.86 0.109], 'FontSize', 30,...
                                                'HorizontalAlignment', 'center');

% Displaying current simulation time
simu_time_display = text('Parent', ha, 'Units', 'Normalized',...
                                                'String', '0', 'LineWidth' , 1,    ...
                                                'Position', [0.108 0.59],    ...
                                                'Color', [1 0.86 0.109], 'FontSize', 30,...
                                                'HorizontalAlignment', 'center');
% Two concentration bars embodying the changes process of concentrations vividly
center_concentration_bar = rectangle('Position', [670 620 35 0], ...
                                                 'FaceColor', [0.70588, 0.9294, 1]);
nearby_concentration_bar = rectangle('Position', [1170 620 35 0], ...
```

```
                                        'FaceColor', [0.70588, 0.9294, 1]);


% Assert and catch the medicine type, which could be represented by
% color of the medicine
switch medicine_type
    case 1
        % Blue
        [medicine_12, medicine_21, ...
            medicine_13, medicine_31] = create_medicine([0 0 1]);
    case 2
        % Red
        [medicine_12, medicine_21, ...
            medicine_13, medicine_31] = create_medicine([1 0 0]);
    case 3
        % Green
        [medicine_12, medicine_21, ...
            medicine_13, medicine_31] = create_medicine([0 1 0]);
    case 4
        % Black
        [medicine_12, medicine_21, ...
            medicine_13, medicine_31] = create_medicine([0 0 0]);
    case 5
        % White
        [medicine_12, medicine_21, ...
            medicine_13, medicine_31] = create_medicine([1 1 1]);
end

% Below are temporary variables, which are utilized to
% store the position info of the medicine
now_12 = zeros(5, 4);
for i = 1:5
    now_12(i, :) = get(medicine_12(i), 'Position');
end

now_21 = zeros(5, 4);
for i = 1:5
    now_21(i, :) = get(medicine_21(i), 'Position');
end

now_13 = zeros(4, 4);
for i = 1:4
    now_13(i, :) = get(medicine_13(i), 'Position');
end
```

```
now_31 = zeros(5, 4);
for i = 1:5
    now_31(i, :) = get(medicine_31(i), 'Position');
end


% Procedure of performing ode45
if (f_t_time == 0 || f_t_time >= T)
    tspan = linspace(0, T, T);
    x_0 = [x1_0, x2_0, 0];
    [t, y] = ode45('odefunc', tspan, x_0);
else
    tspan1 = linspace(0, f_t_time, f_t_time);
    tspan2 = linspace(f_t_time, T, T-f_t_time);
    x_0 = [x1_0/V_1, x2_0/V_2, 0];

    [t1, y1] = ode45('odefunc', tspan1, x_0);

    x_f_t_time = [y1(end, 1), y1(end, 2), y1(end, 3)];
    [t2, y2] = ode45('odefunc', tspan2, x_f_t_time);

    t = [t1; t2];
    y = [y1; y2];
end

% Request for simulation speed
speed_choice = questdlg('请选择仿真速度', ...
                        '提示', ...
                        '慢', '中', '快', '快');
switch speed_choice
    case '慢'
        simu_speed = 1;
    case '中'
        simu_speed = 2;
    case '快'
        simu_speed = 5;
end

% Number of frames
n_frames = size(t, 1);

% Main loop
for k = 1:n_frames
```

```
% Calculate the motion speed of medicine, which could be affected by
% the concentration and the coeffients k_ij
v = max(round(k_12*y(k, 1)/V_1*250), 2);

for j = 1:v
    if ~ishandle(dyn_simu_background)
        set(handles.static_simu, 'Enable', 'on');
        set(handles.dyn_simu, 'Enable', 'on');
        clear;
        return;
    end

    % Alter the position property respectively
    for i = 1:5
        if now_12(i, 1) > 1050
            now_12(i, 1) = 748;
        else
            now_12(i, 1) = now_12(i, 1) + 36/3;
        end
        set(medicine_12(i), 'Position', now_12(i, :));

        if now_21(i, 1) < 780
            now_21(i, 1) = 1130;
        else
            now_21(i, 1) = now_21(i, 1) - 42/3;
        end
        set(medicine_21(i), 'Position', now_21(i, :));

        if i <= 4
            if now_13(i, 2) > 850
                now_13(i, 2) = 635;
            else
                now_13(i, 2) = now_13(i, 2) + 36/3;
            end
            set(medicine_13(i), 'Position', now_13(i, :));
        end

        if now_31(i, 1) > 590
            now_31(i, 1) = 307;
        else
            now_31(i, 1) = now_31(i, 1) + 36/3;
        end
        set(medicine_31(i), 'Position', now_31(i, :));
    end
```

```
                % Display info
                if mod(j, v) == 0
                    % Display simulation time
                    set(simu_time_display, 'String', num2str(k));
                    % Display the concentration of the center room
                    set(center_concentration_display, 'String', num2str(roundn(y(k, 1)/V_1, -
4)));
                    % Display the concentration of the nearby room
                    set(nearby_concentration_display, 'String', num2str(roundn(y(k, 2)/V_2, -
4)));

                    set(center_concentration_bar, 'Position', [670 620-200*y(k, 1)/max(y(:, 1))
35 200*y(k, 1)/max(y(:, 1))]);
                    set(nearby_concentration_bar, 'Position', [1170 620-200*y(k, 2)/max(y(:, 2))
35 200*y(k, 2)/max(y(:, 2))]);
                end

                pause(0.5/(v*simu_speed));
            end
        end
        % Enable the static simulation button as well as the dynamic simulation button
        set(handles.static_simu, 'Enable', 'on');
        set(handles.dyn_simu, 'Enable', 'on');

        selection = questdlg('动态仿真已结束，是否退出仿真图形界面？', ...
                                '提示', ...
                                'Yes','No','No');
        switch selection
            case 'Yes'
                delete(gcf);
            case 'No'
                return
        end
end

function [medicine_12, medicine_21, medicine_13, medicine_31] =
create_medicine(FaceColor)
        %CREATE_MEDICINE helps perform the generation of medicine objects.
        %
        % Created by Xingyi Li on Dec. 24, 2019.
        % Copyright (c) 2019 Huazhong University of Science and Technology.
        % All rights reserved.
```

```
    % For medicine_12: [748 521 20 20] & [1050 521 20 20], delta = 75
    medicine_12 = zeros(5, 1);
    for i = 1:5
        medicine_12(i) = rectangle('Position', [748 + 75*(i-1) 521 20 20], ...
                                   'Curvature', [1 1], ...
                                   'FaceColor', FaceColor);
    end
    % For medicine_21: [780 635 20 20] & [1130 635 20 20], delta = 88
    medicine_21 = zeros(5, 1);
    for i = 1:5
        medicine_21(i) = rectangle('Position', [1130 - 88*(i-1) 635 20 20], ...
                                   'Curvature', [1 1], ...
                                   'FaceColor', FaceColor);
    end
    % For medicine_13: [698 635 20 20] & [698 850 20 20], delta = 72
    medicine_13 = zeros(4, 1);
    for i = 1:4
        medicine_13(i) = rectangle('Position', [698 635 + 72*(i-1) 20 20], ...
                                   'Curvature', [1 1], ...
                                   'FaceColor', FaceColor);
    end
    % For medicine_31: [307 533 20 20] & [590 533 20 20], delta = 71
    medicine_31 = zeros(5, 1);
    for i = 1:5
        medicine_31(i) = rectangle('Position', [307 + 71*(i-1) 533 20 20], ...
                                   'Curvature', [1 1], ...
                                   'FaceColor', FaceColor);
    end
end

function my_closereq(src, callbackdata)
    %MY_CLOSEREQ offers a self-defined close request to display a question dialog box
    %
    % Created by Xingyi Li on Dec. 24, 2019.
    % Copyright (c) 2019 Huazhong University of Science and Technology.
    % All rights reserved.

    selection = questdlg('是否退出仿真图形界面？', ...
                         '提示', ...
                         'Yes','No','No');
    switch selection
        case 'Yes'
            delete(gcf);
        case 'No'
```

```
                return
        end
end
```