# 实验六 自动驾驶建模与仿真

**【实验目的】**

1. 了解Matlab/Simulink软件环境，熟悉Simulink建模步骤；
2. 了解Matlab/Simulink常用功能模块，学会进行串行通信建模；
3. 了解车辆运动控制的基本原理，学会简单的车辆运动控制建模及仿真；
4. 了解自动驾驶建模的基本过程，了解典型ADAS系统模型的应用特点。
5. 了解自动驾驶相关函数，认识自动驾驶函数的功能及用途。

**【实验性质】**

验证性实验。

**【实验要求】**

Matlab2020+软件。

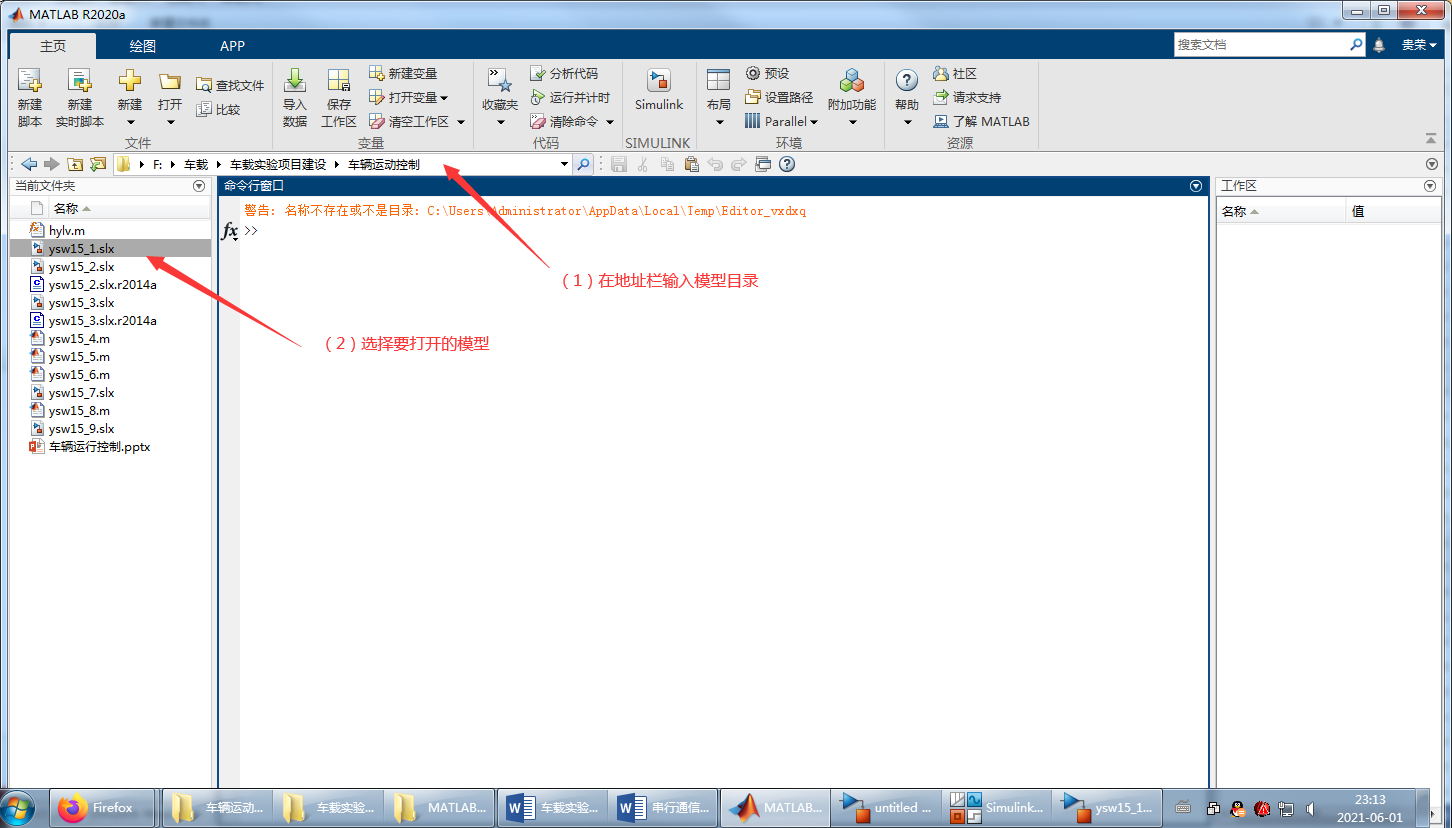
**【实验内容】**

1. 操作Matlab软件，学会进行Simulink图形化建模。
2. 学习Matlab/Simulink常用功能模块，进行简单串行通信建模。
3. 调试车辆运动控制模型。
4. 学习典型ADAS系统，分析其功能模块组成。
5. 调用自动驾驶函数进行自动驾驶场景仿真。

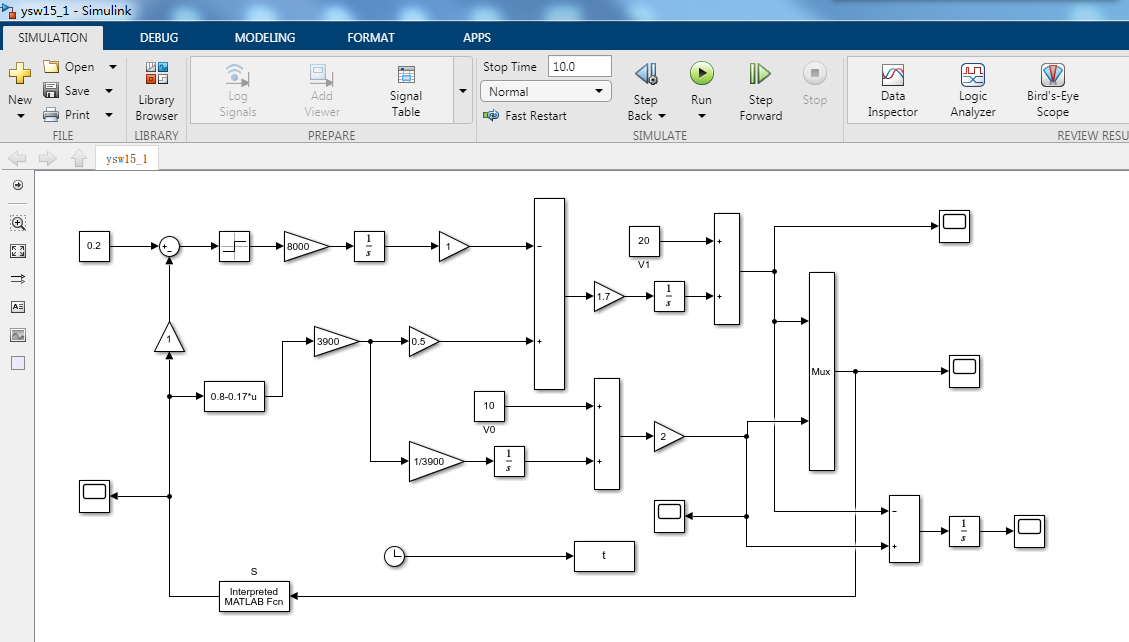
**【实验步骤】**

**1、车辆运动控制建模学习**

（1）按下图打开模型



按上面的操作方式打开的车辆纵向运动控制模型，如下图所示：

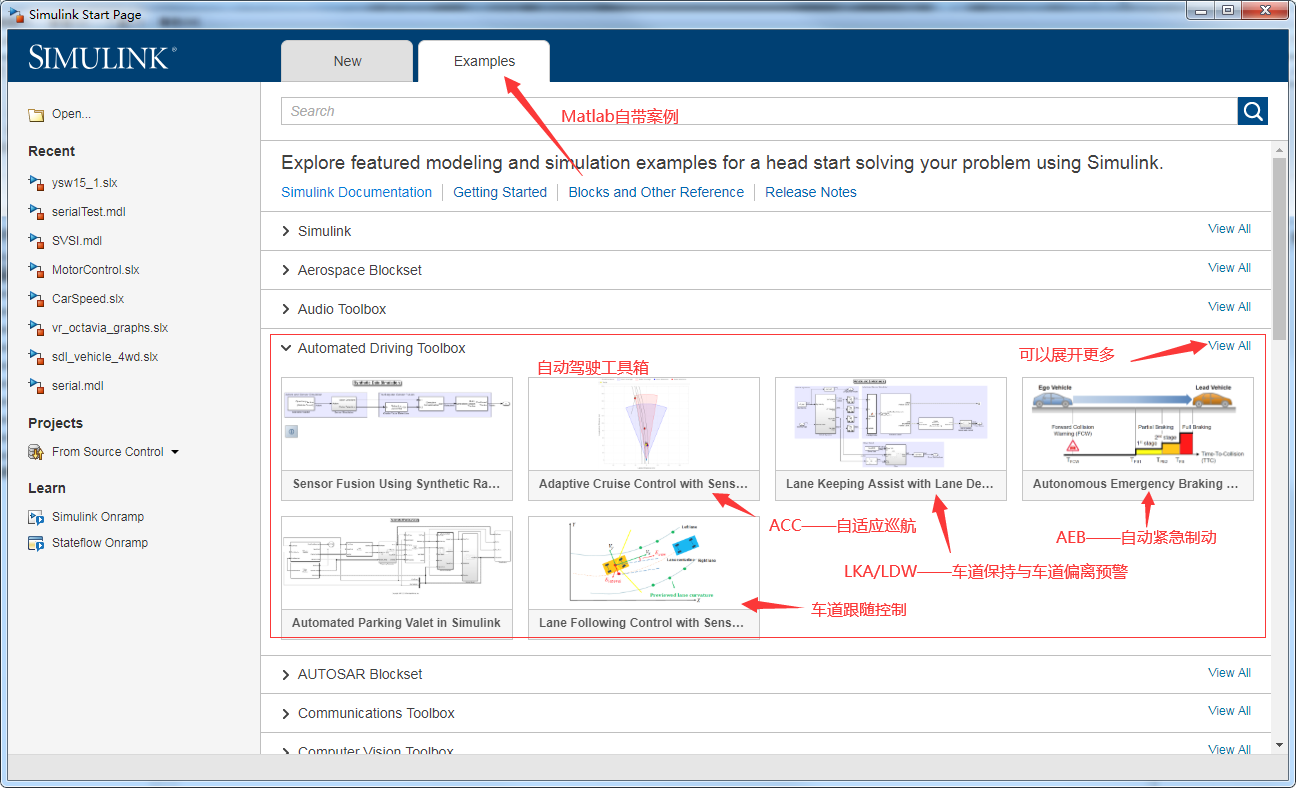




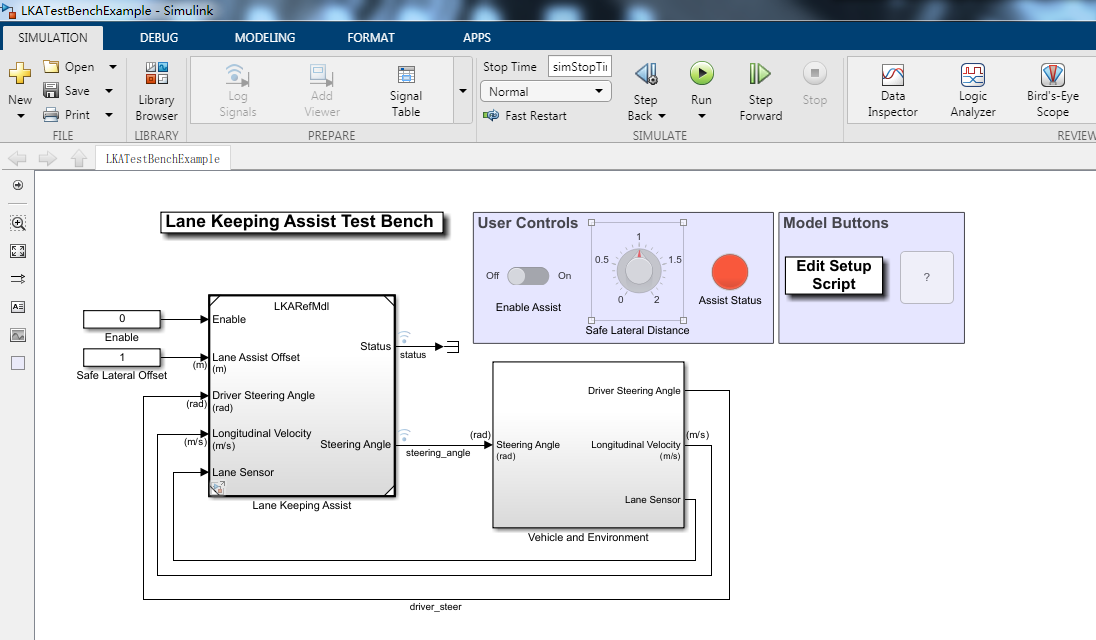
上面模型为汽车单轮制动控制模型，打开Simulink工具箱（即“Library Browser”），在工具箱中查找模型中的图形符号，学习工具箱中模型库的使用方法。

**2、高级驾驶辅助系统（ADAS）建模学习**

　　展开自动驾驶工具箱，可以发现里面有很多Matlab自带的案例，选择其一打开，分析其组成及执行过程。



车道保持辅助系统如下图所示：

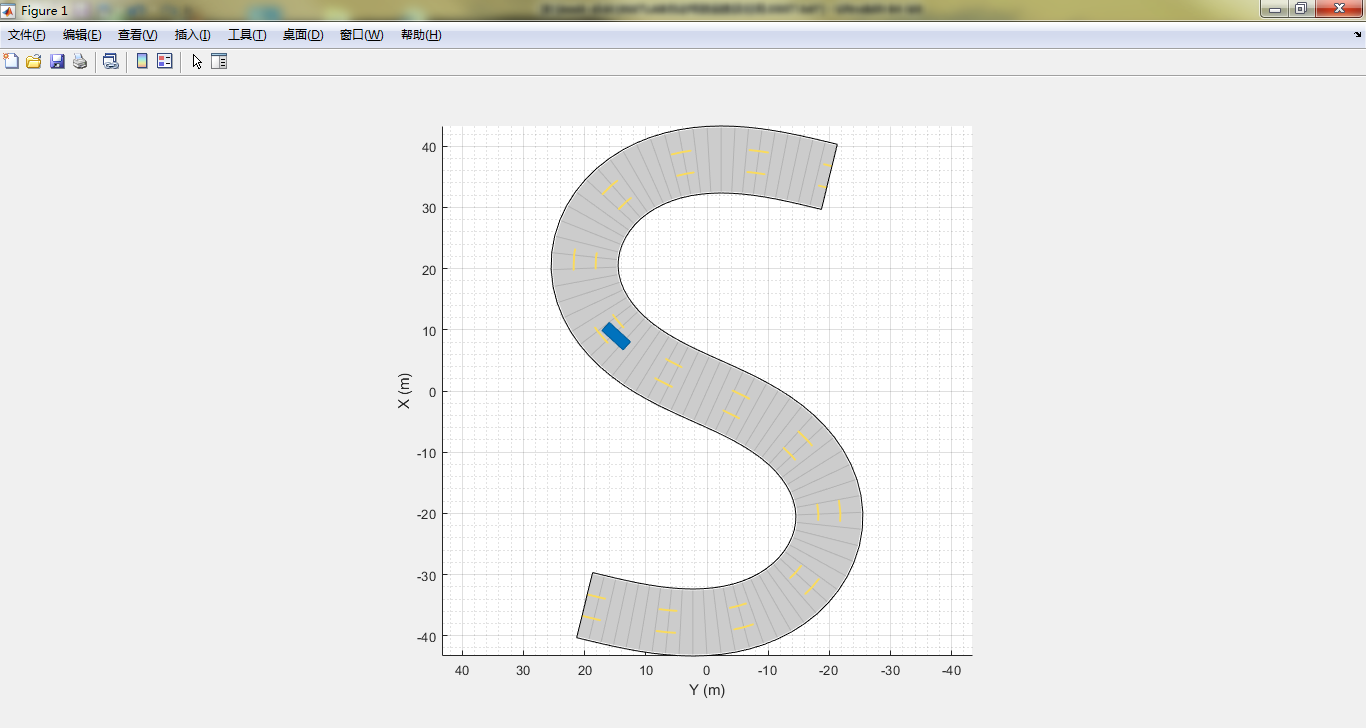


**3、自动驾驶场景练习**

（1）模拟一辆汽车在S形道路上运动

在Matlab命令行窗口输入以下命令（复制—粘贴）

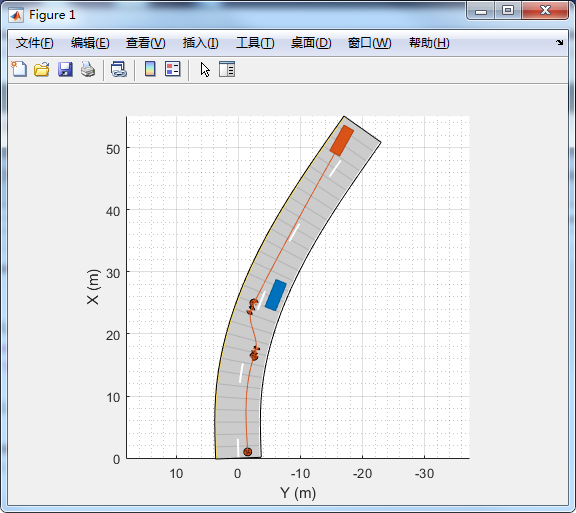
|  |  |
| --- | --- |
| s=drivingScenario;  roadCenters=[-35,20,0;-20,-20,0;0,0,0;20,20,0;35,-20,0];  lm=[laneMarking('Solid','Color','w'); ...  laneMarking('Dashed','Color','y');...  laneMarking('Dashed','Color','y');...  laneMarking('Solid','Color','w')];  ls=lanespec(3,'Marking',lm);  road(s, roadCenters, 'Lanes', ls);  car=vehicle(s, 'ClassID',1,'Position',[-35,20,0]);  waypoints=[-35,20,0;-20,-20,0; 0,0,0;20,20,0; 35,-20,0];  speed=10;  trajectory(car, waypoints, speed);  plot(s);  **while advance(s)**  **lbdry=laneBoundaries(car);**  **end** | %定义驾驶场景  %设置道路中心  %设置车道标线  %设置车道规范  %在驾驶场景中添加道路  %在驾驶场景中添加静止车辆  %在驾驶场景中添加运动车辆  %设置车辆航路点  %设置车速  %创建车辆轨迹  %绘制驾驶场景  %仿真循环开始  %显示车辆边界  %仿真结束 |
|  | 姿态：  ActorID：参与者标识  Position：参与者位置（单位m）  Velocity：速度（单位m/s）  Roll：横滚角（度）  Pitch：俯仰角（度）  Yaw：偏航角（度）  AngularVelocity：角速度（度/s） |



（2）模拟静止车辆和运动车辆在道路上行驶。

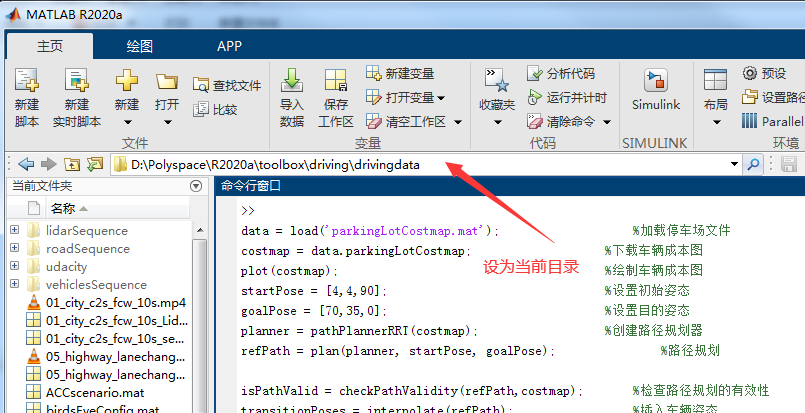
在Matlab命令行窗口输入以下命令（复制—粘贴）

|  |  |
| --- | --- |
| s=drivingScenario;  roadCenters=[0,0; 10,0; 53,-20];  road(s, roadCenters, 'Lanes', lanespec(2));  stationaryCar=vehicle(s, 'Position',[25,-5.5,0], 'Yaw',-22);  passingCar=vehicle(s);  waypoints=[1,-1.5; 16.4,-2.5; 17.4,-2.8; 23.8,-2; 25,-2.5; 50,-16];  speed=5;  **trajectory(passingCar, waypoints, speed);**  plot(s, 'Waypoints', 'on');  rec=record(s);  rec(1).ActorPoses(2);  rec(end).ActorPoses(2);  ans  **%while advance(s)**  **% pause(0.01);**  **%end**  **%poses=actorPoses(s);** | %定义驾驶场景  %设置道路中心  %在驾驶场景中添加道路  %在驾驶场景中添加静止车辆  %在驾驶场景中添加运动车辆  %设置车辆航路点  %设置车速  %创建车辆轨迹  %绘制驾驶场景  %运行场景并记录  %显示车辆模拟开始姿态  %显示车辆模拟结束姿态  %姿态结构  %仿真循环开始  %停顿时间  %仿真结束  %获取交通参与者姿态 |
|  | 姿态：  ActorID：参与者标识  Position：参与者位置（单位m）  Velocity：速度（单位m/s）  Roll：横滚角（度）  Pitch：俯仰角（度）  Yaw：偏航角（度）  AngularVelocity：角速度（度/s） |



1. 停车场停车路径规划（基于碰撞的路径规划）

在Matlab安装目录中搜索 parkingLotCostmap.mat 文件所在路径，并将该目录在Matlab地址栏中打开（设为当前路径）

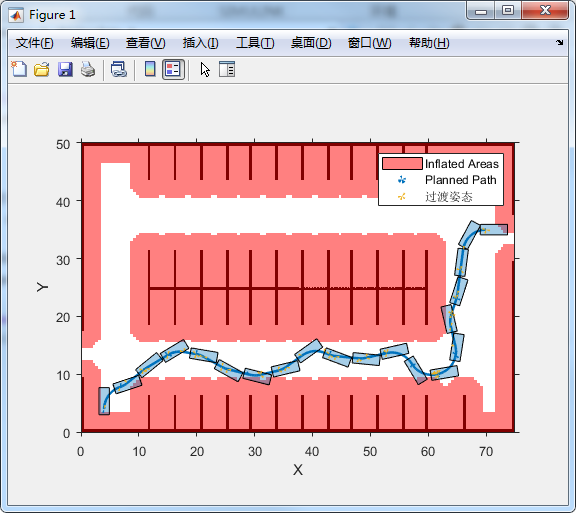


|  |
| --- |
| data = load('**parkingLotCostmap.mat**'); %加载停车场文件  costmap = data.parkingLotCostmap; %下载车辆成本图  figure; %设置图形窗口  plot(costmap); %绘制车辆成本图  vehicleDims = vehicleDimensions(4.5, 1.7); %设置车辆尺寸  numCircles = 3; %圆圈数  ccConfig = inflationCollisionChecker(vehicleDims, numCircles); %检测碰撞  costmap.CollisionChecker = ccConfig; %重新配置成本图  figure; %设置图形窗口  plot(costmap); %绘制车辆成本图  startPose = [11, 10, 0]; %设置初始姿态  goalPose = [31.5, 17, 90]; %设置目的姿态  planner = pathPlannerRRT(costmap); %创建路径规划器  refPath = plan(planner, startPose, goalPose); %路径规划  hold on; %保存图形  plot(refPath); %绘制规划路径 |
|  |

1. 驶出停车场路径规划（验证路径规划）

在Matlab安装目录中搜索 parkingLotCostmap.mat 文件所在路径，并将该目录在Matlab地址栏中打开（设为当前路径）。

|  |
| --- |
| data = load('**parkingLotCostmap.mat**'); %加载停车场文件  costmap = data.parkingLotCostmap; %下载车辆成本图  plot(costmap); %绘制车辆成本图  startPose = [4,4,90]; %设置初始姿态  goalPose = [70,35,0]; %设置目的姿态  planner = pathPlannerRRT(costmap); %创建路径规划器  refPath = plan(planner, startPose, goalPose); %路径规划  isPathValid = checkPathValidity(refPath,costmap); %检查路径规划的有效性  transitionPoses = interpolate(refPath); %插入车辆姿态  hold on; %保存图形  plot(refPath,'DisplayName','Planned Path'); %绘制路径规划  scatter(transitionPoses(:,1),transitionPoses(:,2),[],'filled','DisplayName','过渡姿态'); %绘制过渡姿态 |



**【扩展思考】**

1、描述MATLAB自动驾驶工具箱中的例程LKA/LDW、ACC、AEB、LFC运行调试时会输出哪些信号（任选其一）。