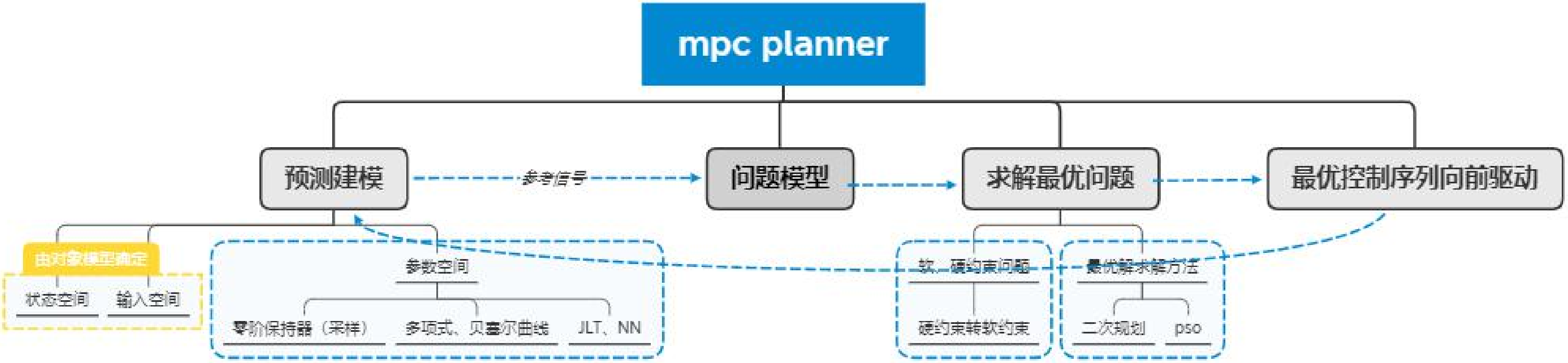
第八章作业分享

主讲人 王亚

纲要

内容整理：

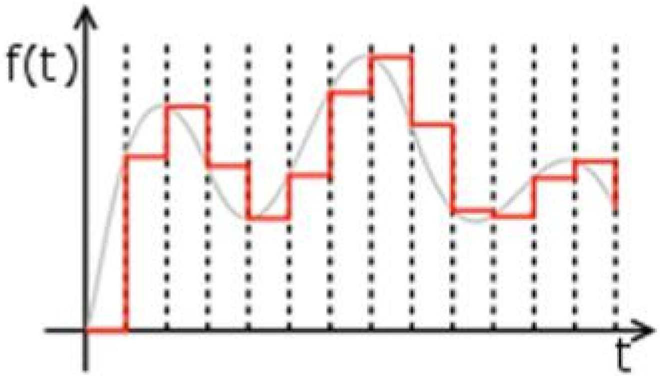


mpc planner
预测建模
参考信号
问题模型
求解最优问题
最优控制序列向前驱动
由对象模型确定
状态空间
输入空间
参数空间
零阶保持器（采样）
多项式、贝塞尔曲线
JLT、NN
软、硬约束问题
硬约束转软约束
最优解求解方法
二次规划
pso

参数空间

目的是将连续问题，变为离散问题，将无限维问题，变为有限维问题。

方法一：零阶保持器



f(t)为最优控制解，零阶保持器是将问题：求解f(t)转换为求解f(kT),k=1,2,…。T是采样时间

参数空间

目的是将连续问题，变为离散问题，将无限维问题，变为有限维问题。

方法二：多项式或贝塞尔曲线

$$u(t) = at^3 + bt^2 + ct + d$$

u(t)为最优控制解，利用多项式将问题：求解f(t)转换为求解a,b,c,d等多项式有限个参数

参数空间

目的是将连续问题，变为离散问题，将无限维问题，变为有限维问题。

方法三：JLT或者神经网络

- BSCP

利用初始状态和目标状态，求解出某种意义下最优的u，即将求解最优u(t)，转换为目标状态空间问题。

MPC框架

1. 建立预测模型

2. 建立问题模型

3. 求解最优问题

4. 最优控制序列向前驱动

```
for t=0.2:0.2:10
    %% Construct the prediction matrix
    [Tp, Tv, Ta, Bp, Bv, Ba] = getPredictionMatrix(K,dt,p_0,v_0,a_0);

    %% Construct the optimization problem
    H = w4*eye(K)+w1*(Tp'*Tp)+w2*(Tv'*Tv)+w3*(Ta'*Ta);
    F = w1*Bp'*Tp+w2*Bv'*Tv+w3*Ba'*Ta;

    %% Solve the optimization problem
    J = quadprog(H,F,[],[]);

    %% Apply the control
    j = J(1);
    p_0 = p_0 + v_0*dt + 0.5*a_0*dt^2 + 1/6*j*dt^3;
    v_0 = v_0 + a_0*dt + 0.5*j*dt^2;
    a_0 = a_0 + j*dt;

    %% Log the states
    log = [log; t p_0 v_0 a_0];
end
```

第一题

- 第一题：三维三阶积分模型，跟踪螺旋线

在第一题中使用了和示例一样的参数空间：零阶保持器，采样周期0.2s，预测时域4s，控制时域0.2s一个周期。

三维三阶积分模型，每一维都和示例形式一样，在代码中，已经给出了每一维的预测模型实现函数，直接调用即可。

第一题

- 第一题：三维三阶积分模型，跟踪螺旋线

```
p_0 = [0 8 20];
v_0 = [0 0 0];
a_0 = [0 0 0];
K=20;
dt=0.2;

P=[];
V=[];
A=[];
```

初始状态

预测时域是4s，零阶保持器的**采样周期**是0.2s，所以在0~4s内，会采样出来20个信号，k=20，用于跟采样信号个数相关的矩阵维数；

第一题

- 第一题：三维三阶积分模型，跟踪螺旋线

```
for t=0.2:0.2:40
    %% Construct the reference signal
    for n = 1:20
        tref = t + n*0.2;
        r=0.25*tref;
        pt(n,1) = r*sin(0.2*tref);
        vt(n,1) = r*cos(0.2*tref);
        at(n,1) = -r*sin(0.2*tref);

        pt(n,2) = r*cos(0.2*tref);
        vt(n,2) = -r*sin(0.2*tref);
        at(n,2) = -r*cos(0.2*tref);

        pt(n,3) = 20 - 0.5*tref;
        vt(n,3) = -0.5;
        at(n,3) = 0;
    end
    %% Do the MPC
```

产生参考信号，也就是要跟踪的螺旋形信号。

注意到：参考信号的产生不是用0.2的采样频率进行采样离散，而是在每个0.2s内由进行了20次采样。

这里我的理解：**和老师课上讲的一样：由于收到速度、加速度等约束限制，mpc求解出的控制量，不一定可以在0.2s内使状态转移到目标状态，因此将目标状态步长变小，以使得0.2s内，状态更可能到目标状态**

第一题

- 第一题：三维三阶积分模型，跟踪螺旋线

```
%% Do the MPC
%% Please follow the example in linear mpc part to fill in the code here to do the tracking
for n = 1:1:size(pt,1)
    j(1) = xy_axis_mpc(K,dt,p_0(1),v_0(1),a_0(1),pt(1,1),vt(1,1),at(1,1));%x
    j(2) = xy_axis_mpc(K,dt,p_0(2),v_0(2),a_0(2),pt(1,2),vt(1,2),at(1,2));%y
    j(3) = z_axis_mpc(K,dt,p_0(3),v_0(3),a_0(3),pt(1,3),vt(1,3),at(1,3));%z
    for i=1:3
        [p_0(i),v_0(i),a_0(i)] = forward(p_0(i),v_0(i),a_0(i),j(i),dt);
    end
end
```

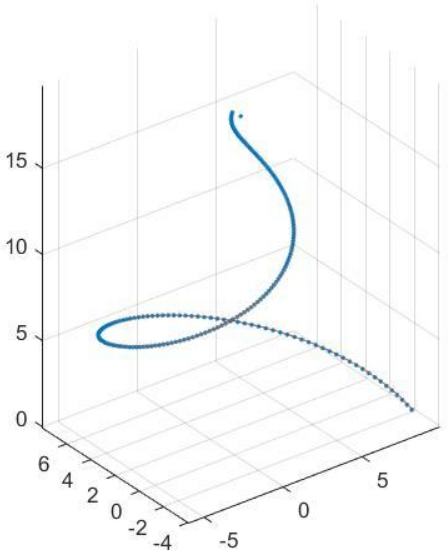
这段代码就是作业要补充的代码。由于Z方向的约束不一样，所以单独写了函数。两个函数，在作业已经完整给出了，与示例代码一样，不再详细叙述。

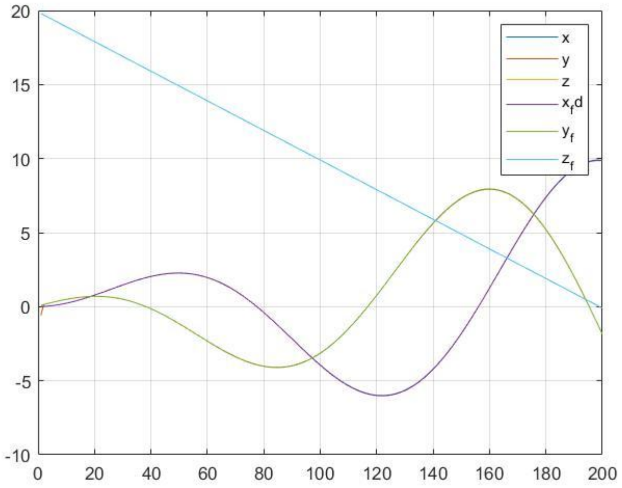
将最优序列第一个值（控制时域是0.2s），给到预测模型，状态向前迭代。

第一题

● 第一题：三维三阶积分模型，跟踪螺旋线

结果：每个0.2s的参数信号进行了20次采样



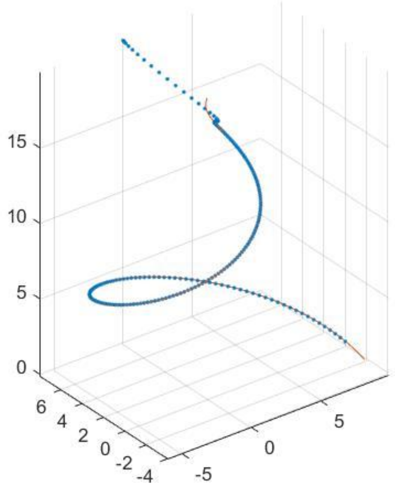


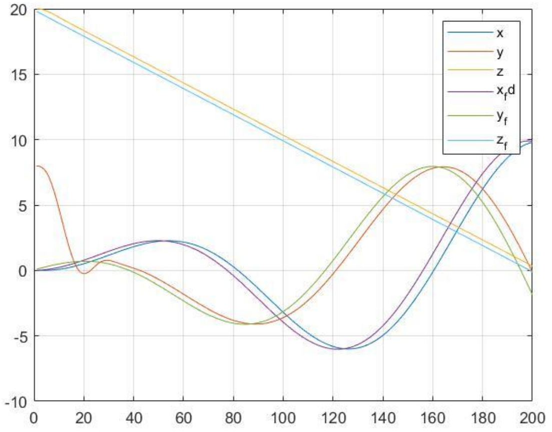
第一题

●第一题：三维三阶积分模型，跟踪螺旋线

结果：每个0.2s的参数信号不进行20次采样

```
%% Construct the reference signal
for n = 1:1 %20
```

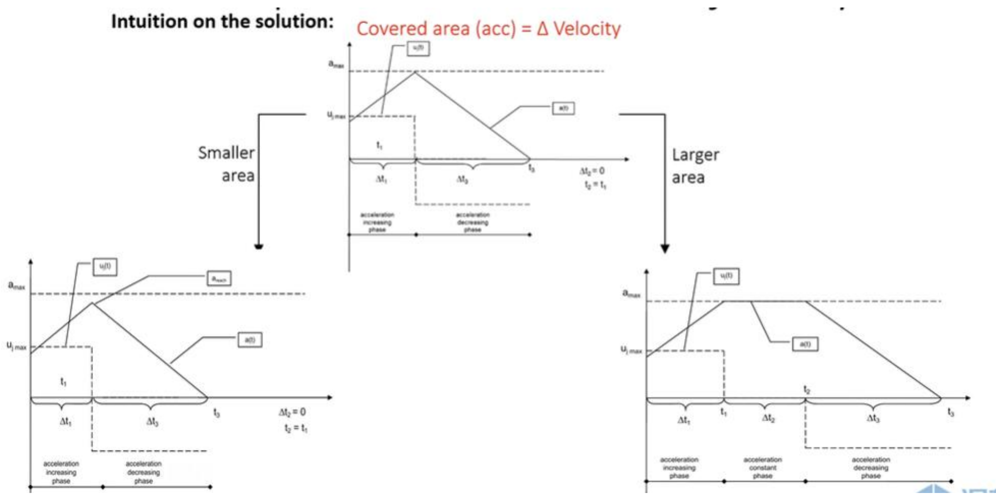




JLT和MPC

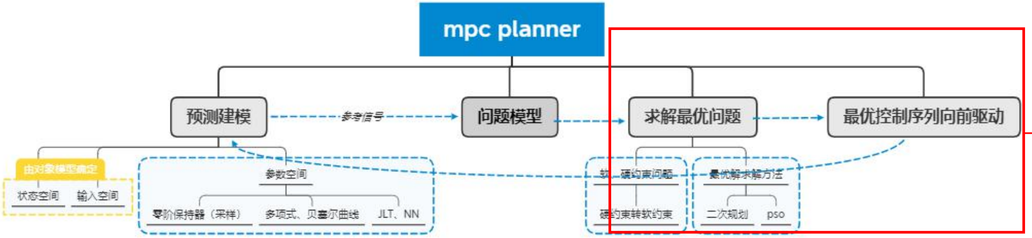
JIT算子解决的是:已知初始状态和目标状态，求出最优控制序列

- 只能求积分模型
- 如果是积分模型，求解很快



JLT和MPC

JIT是怎么和MPC结合的？



JLT代替了零阶保持器模型 MPC过程中，求解最优解和向前驱动得到一段最优轨迹的过程

问题：

1.由于mpc有硬约束，怎么样将得到的一次最优轨迹带入目标函数求的cost，(直接求没办法求，因为不知道是否满足约束)。

2.JLT参数空间是将u(t)连续空间，转换为目标状态空间（一个目标状态对应一个最优序列），怎么样在目标状态空间选离散的目标状态？

JLT和MPC

深蓝学院 shenlanxueyuan.com

JIT是怎么和MPC结合的？

第一个问题：将所有的硬约束转为软约束就可以计算cost值了。

$$\min C_F\left(x(t_f)\right) + \int_{t=t_0}^{t_f} C_R(x,u)\,dt$$

$$\boxed{\dot{x} = f(x,u)}$$

JLT就是解决积分模型，算法本身考虑模型约束，因此不用转软约束

$$\boxed{\begin{array}{c} g(x,u) < 0 \\ h(x,u) = 0 \end{array}}$$

JLT算法计算过程了考虑边界约束和能力约束，因此不用转软约束

$$\boxed{x \notin Obstacle}$$

JLT算法没有考虑障碍物约束，因此需要转为软约束，计算每次JLT得到状态轨迹cost

障碍物硬约束转软约束，课上讲的很清楚，不再赘述

JLT和MPC

JIT是怎么和MPC结合的？

第二个问题：怎么样在目标状态空间选择离散的目标状态。
最直接的想法是：在目标状态空间内均匀采样，但是均匀采样不一定能采样到最优的目标状态。均匀采样升级版本是采样PSO算法，对每个采样点进行迭代，逐渐收敛到最优

PSO

PSO:

描述为:

1.随机撒一些点，其位置记为θ$_i$

2. θ$_i$下一次位置更新的的速度方向（或者叫做步长）δ:

δ = 上一次的速度方向+权重1*全局最优解方向+权重2*当前点历史轨迹中最优的方向

Algorithm 1: Particle Swarm Optimization.

Input: $\mathbf{x}_{\text{ini}}$, $\mathcal{M}$(map)
Output: θ^* (best end state constraint)
$\Theta \leftarrow$ Particle_Initialization();
$c_i^* \leftarrow \infty$, $\theta_i^* \leftarrow \theta_i$, $\delta_i \leftarrow$ rand, $\forall i \in [1, \text{size}(\Theta)]$
for $m = 1$ to MAX_ITERS **do**
  for each $\boldsymbol{\theta}_i \in \Theta$ **do**
    $[\mathbf{x}(t), \mathbf{u}(t)] = \mathcal{S}_{\text{NN}}(\mathbf{x}_{\text{ini}}, \boldsymbol{\theta}_i)$
    $c_i = J(\mathbf{x}(t), \mathbf{u}(t), \mathcal{M})$
    if $c_i < c_i^*$ **then**
      $c_i^* = c_i$
      $\boldsymbol{\theta}_i^* = \boldsymbol{\theta}_i$
  $i^* = \underset{i}{\text{argmin}}(c_i^*)$
  $\boldsymbol{\theta}^* = \boldsymbol{\theta}_{i^*}$
  for each $\boldsymbol{\theta}_i \in \Theta$ **do**
    $\boldsymbol{\delta}_i = \boldsymbol{\delta}_i + k_1 \cdot \text{rand} \cdot (\boldsymbol{\theta}_i^* - \boldsymbol{\theta}_i) + k_2 \cdot \text{rand} \cdot (\boldsymbol{\theta}^* - \boldsymbol{\theta}_i)$
    $\boldsymbol{\theta}_i = \boldsymbol{\theta}_i + \boldsymbol{\delta}_i$

第二题

两维变量的pso算法：
示例中，是一维变量进行pso迭代更新，作业题中，有两个变量需要更新
每一维根据算法单独更新。代码数据结构说明如下：

```
%theta, v_end, v_theta, v_vend, best_theta, best_v, best_cost
P=zeros(N,7);
global_best = zeros(1,3);
global_best(1) = last_theta-theta;
global_best(2) = last_v;
global_best(3) = evaluate(R,omega,p0,last_theta-theta,last_theta-theta,v_ini,last_v);
```

P:如注释所示，第一，二位保持了两个待迭代的变量，三、四维分别是这两个变量下次更新的速度方向，第五、六位分别是两个变量历史最优位置，第7位为全局最优位置

第二题

两维变量的pso算法：
示例中，是一维变量进行pso迭代更新，作业题中，有两个变量需要更新
每一维根据算法单独更新。代码数据结构说明如下：

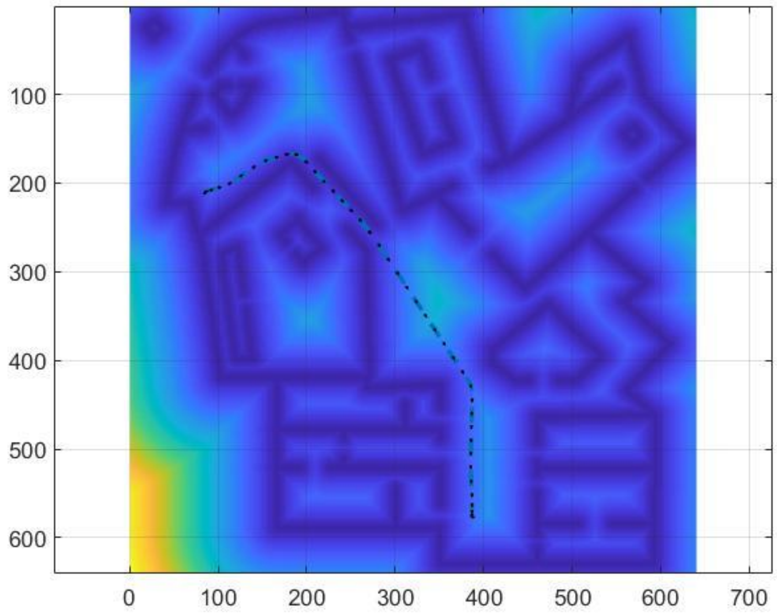
```
if (j~=1)
    %update the particle position for the i'th particle
    %---------------------------------------------------------------
    % To be finished by the student:
    %---------------------------------------------------------------
    P(i,3) = P(i,3)+k1*rand*(P(i,5)-P(i,1)) + k2*rand*(global_best(1) - P(i,1));
    P(i,4) = P(i,4)+k1*rand*(P(i,6)-P(i,2)) + k2*rand*(global_best(2) - P(i,2));
    P(i,1) = P(i,1)+ P(i,3);
    P(i,2) = P(i,2)+ P(i,4);
end
```

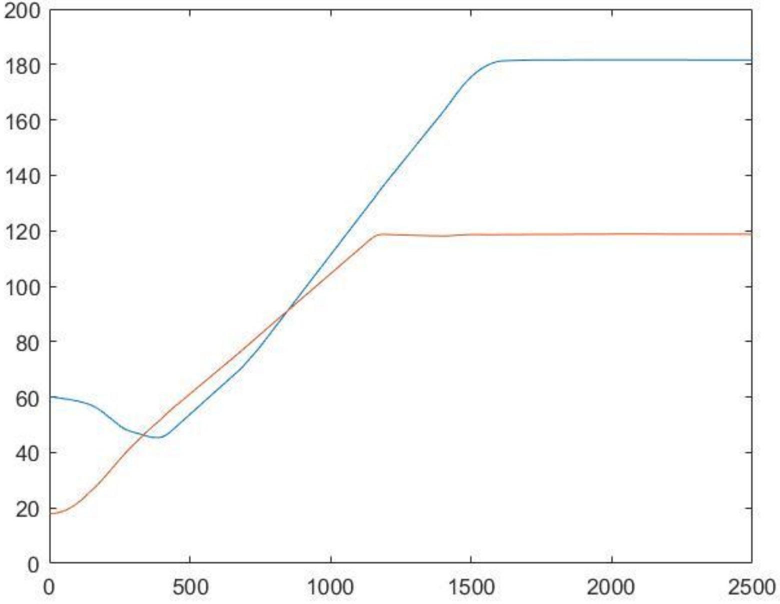
计算各自的速度方向

根据速度方向，更新各自位置

第二题

结果：


100
200
300
400
500
600
0
100
200
300
400
500
600
700


200
180
160
140
120
100
80
60
40
20
0
0
500
1000
1500
2000
2500

神经网络

上述说明了JLT怎么和MPC结合起来使用的，但是JLT并不是通用的求解“从初始状态到目标状态，得到最优的控制序列和最优轨迹”的算法，它只适应于积分模型。

想要得到任意模型的“从初始状态到目标状态，得到最优的控制序列和最优轨迹”解，这个算法工具便是神经网络。

在线问答

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening