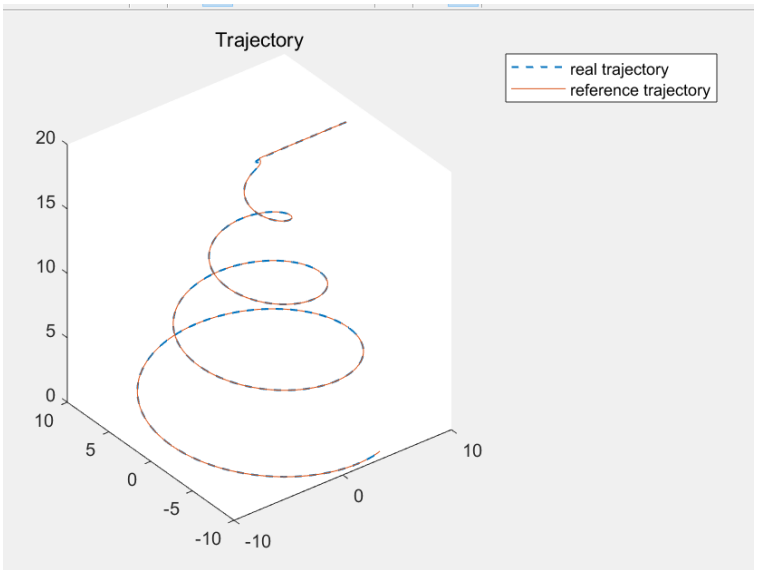
一．代码结构

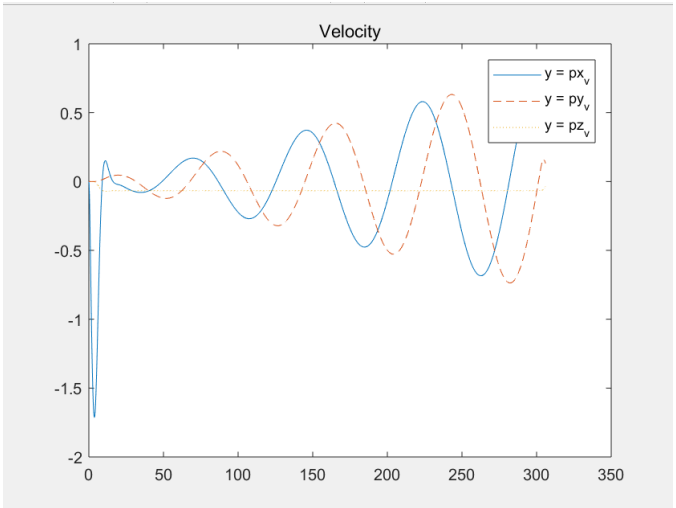
1. function [Px, Py, Pz] = generate_conicalspiral(K, horizontal_n)
   1.此函数生成需要跟踪的轨迹，水平直线部分和下降的螺旋线
   2.K:完成螺旋线需要走过的步数，螺旋线生成函数考虑了 w = 0.08 rad/s
   3.horizontal_n：完成直线需要的步数
2. function [log] = getTrajectory(p_0, up_v, bott_v, up_a, bott_a, up_j, bott_j, Target_p, step_n)
   1. 该函数改编自课中提供的matlab代码
   2. P0为起始点位置，up_,bott_参数为v,a,j的上下界,由于约束实现原因，均为正值, Target_p为存储单一轴上的路径点的数组, step_n运动的步数(优化的循环次数)
   3. 将v,a,j作为软约束加入二次约束
   4. 每一次解优化问题，将生成的20个路径点与要追踪的同时刻的20个路径点的拟合作为目标函数
   5. 从新构造二次优化的结构：
      a) $[TpJ + Bp - Target\_p]^T[TpJ + Bp - Target\_p]$
      $= [TpJ + Bp]^T[TpJ + Bp] - 2 * Target\_p *[TpJ + Bp] + constant$
      $= J^TTp^TTpJ + 2 * Bp^TTpJ - 2 * Target\_p * TpJ + constant$
3. main function
   1. 设置初始化参数和软约束范围
   2. 调用generate_conicalspiral 和 getTrajectory
   3. Plot 结果

二．结果图

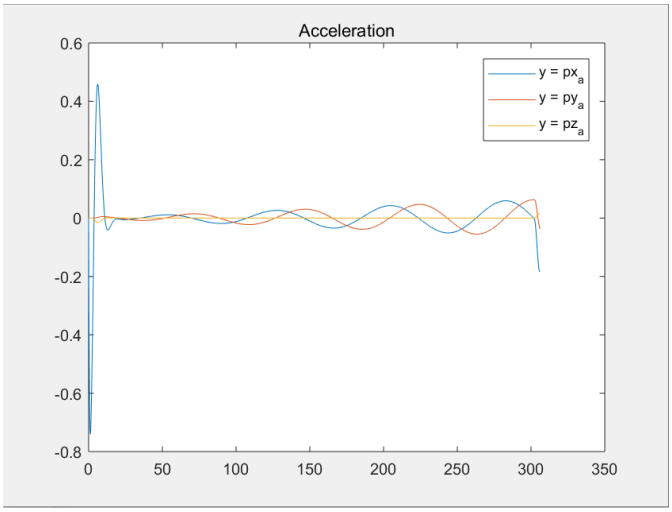
1. 轨迹图：



2. 速度图：



3. 加速度图：



4. jerk图：