实物实验一

1.代码思路

我们将机械臂操纵单物体的过程拆分为七个状态（记作$\{q_i\}(i = start, end, A, B, lift, lay, 0)$），7个阶段。其中各状态分别解释为

- q_0：机械臂**初始**的直立位姿对应的关节角
- q_{start}：机械臂**吸附**物体的位姿对应的关节角
- q_{end}：机械臂**放置**物体的位姿对应的关节角
- $q_{A/B}$：机械臂到达**直线槽A/B侧**的位姿对应的关节角
- q_{lift}, q_{lay}: 机械臂运动过程的**中间点**，q_{lift}表示将物块抬起的关节角，q_{lay}表示将物块放置的关节角

流程 $q_0 \rightarrow q_{start} \rightarrow q_{lift} \rightarrow q_A \Rightarrow q_B \rightarrow q_{lay} \rightarrow q_{end}$ 如此循环（其中$\Rightarrow$是直线运动过程）

各箭头为各状态之间机械臂的移动过程，需要采用不同的路径规划方案
注意：

- 对于A，B两节点的关节角，需要利用世界坐标系下两状态机械臂末端吸盘的位姿坐标，再将其代入我们自己的逆运动学求解器求得。例：

  ```
  qA = iks.solve(np.array([x, y, z, r, p, y]))[:,2]  # [:,2]取某组解
  ```

  此处rpy三个角我们将其规定为`r = np.pi, p = 0, y = -np.pi/2`，以保证其在直线移动的过程中物块不会随意转动。

- 而对于其它关节角，我们利用`Robot`内置函数`syncFeedback()`，手动测出机械臂位于各状态时的关节角并记录。

1.1 直线轨迹规划

直线轨迹规划限制的是世界坐标系下的机械臂的位姿变化。此处我们采用**分段采样法**，先将直线分段，将分段后的每个节点对应的关节角利用逆运动学求解器求得。由于分段数多，各段位姿变化都非常小，因此我们可以直接将各段的始末关节角代入`move()`函数，即可让机械臂完成直线移动。

```
def splinePlanning(startPosition, endPosition):
    iks = IKSolver()
    num_line_points = 1000
    spline_points = []
    spline_kArray = []

    spline_points = [(startPosition + (endPosition - startPosition) * i / num_line_points)
                     for i in range(num_line_points)]

    for i in range(num_line_points):
        spline_kArray.append(
            iks.solve(np.append(spline_points[i], [np.pi, 0, -np.pi/2]))[:, 2])
        contains_nan = np.isnan(spline_kArray[i]).any()
        if contains_nan:
```

```
            spline_kArray[i] = spline_kArray[i-1]
        if spline_kArray[i].any() == np.nan:
            print("res contains nan!")
            return False

    return spline_kArray
```

1.2 非直线轨迹规划

1.2.1 两点间五次多项式轨迹规划

五次多项式

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

有如下矩阵方程

```
timeMatrix = np.matrix([
    [0,             0,              0,          0,          0,      1],
    [time**5,       time**4,        time**3,    time**2,    time,   1],
    [0,             0,              0,          0,          1,      0],
    [5*time**4,     4*time**3,      3*time**2,  2*time,     1,      0],
    [0,             0,              0,          2,          0,      0],
    [20*time**3,    12*time**2,     6*time,     2,          0,      0]
])
invTimeMatrix = np.linalg.inv(timeMatrix)
X = np.matrix([startPosition[i], endPosition[i],        # 第i个关节角
                      startv[i], endv[i], 0, 0]).T      # 填入初末角度，角速度，角加速度
k = np.dot(invTimeMatrix, X)
```

利用矩阵求逆即可得到多项式的系数a_i，接着将时间t代入该多项式即可得到任意时间机械臂的位姿（即各关节角的大小

1.2.1 三点间五次多项式轨迹规划

适用于将 q_{lift}, q_{lay} 作为中间点的轨迹

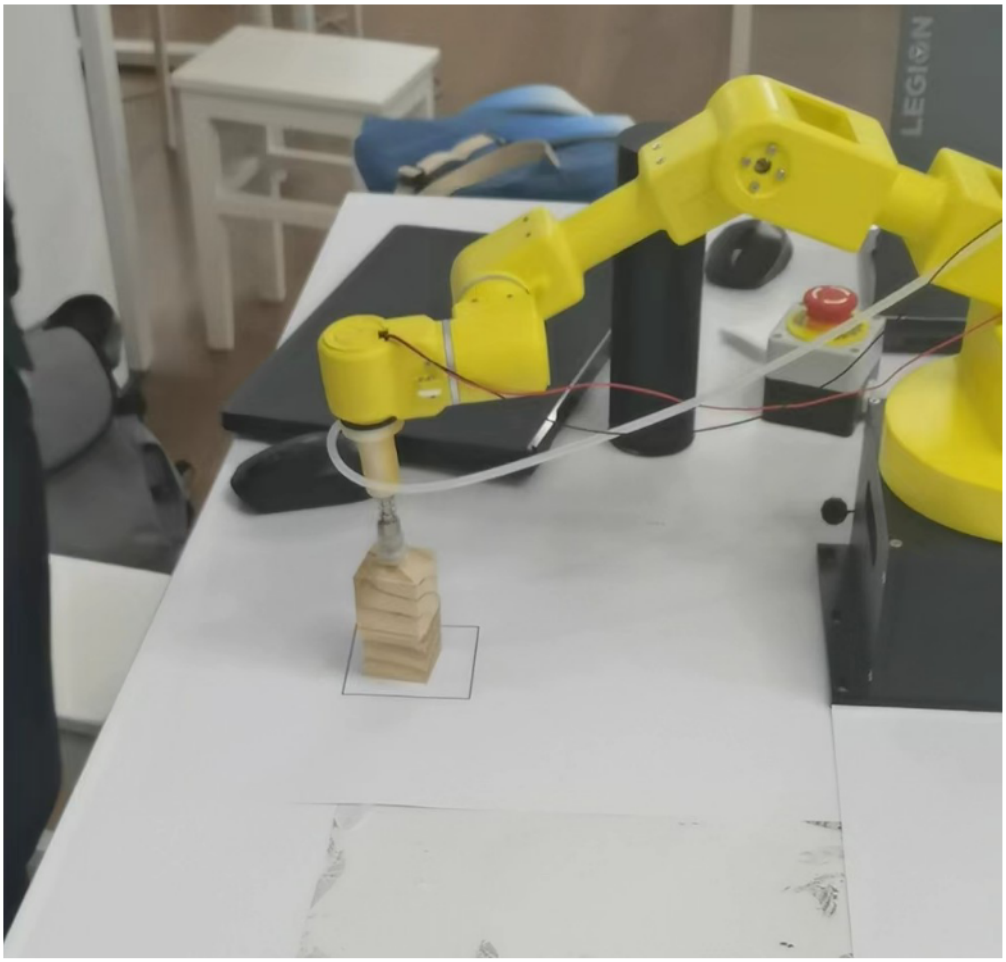
有如下矩阵方程

```
timeMatrix = np.matrix([
    [0,             0,              0,          0,        0,   1],
    [time**5,       time**4,        time**3,    time**2,    time,   1],
    [time1**5,      time1**4,       time1**3,   time1**2,   time1,   1],
    [0,             0,              0,          0,        1,   0],
    [5*time**4,     4*time**3,      3*time**2,      2*time,       1,   0],
    [5*time1**4,    4*time1**3,     3*time1**2,     2*time1,      1,   0],
])
invTimeMatrix = np.linalg.inv(timeMatrix)
X = np.matrix([startPosition[i], middlePosition[i],     # 填入初末角度，中间角度和角速度
                      endPosition[i], 0, midVel, 0]).T
k = np.dot(invTimeMatrix, X)
```

利用该函数能够**使轨迹平滑**，并且**设计lift与lay两个中间状态**可以使机械臂**更好的将物块抬起与放置**，防止出现拖地、碰撞已放置方块的情况

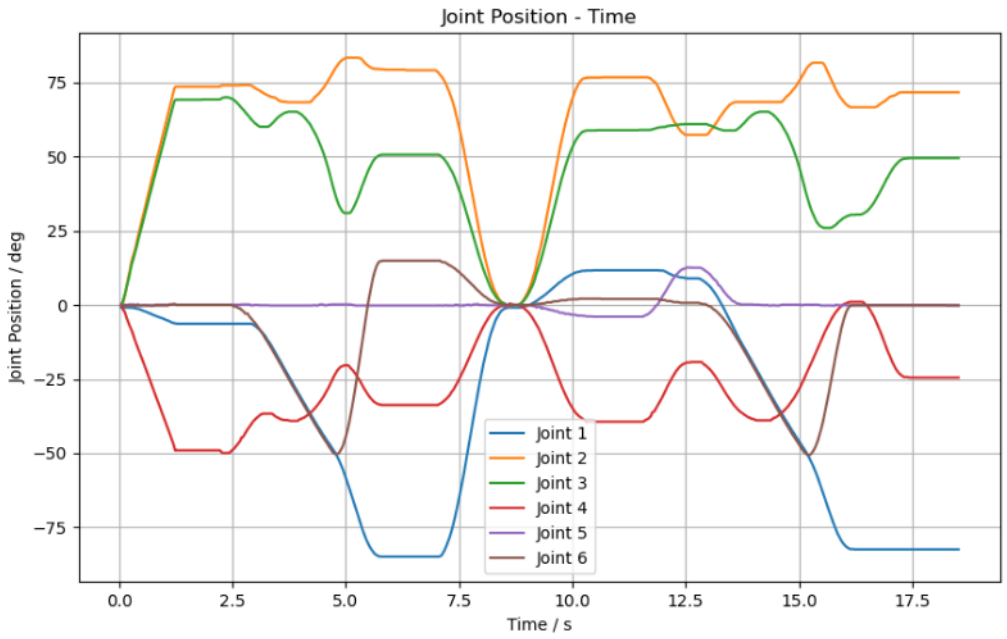
2.仿真结果

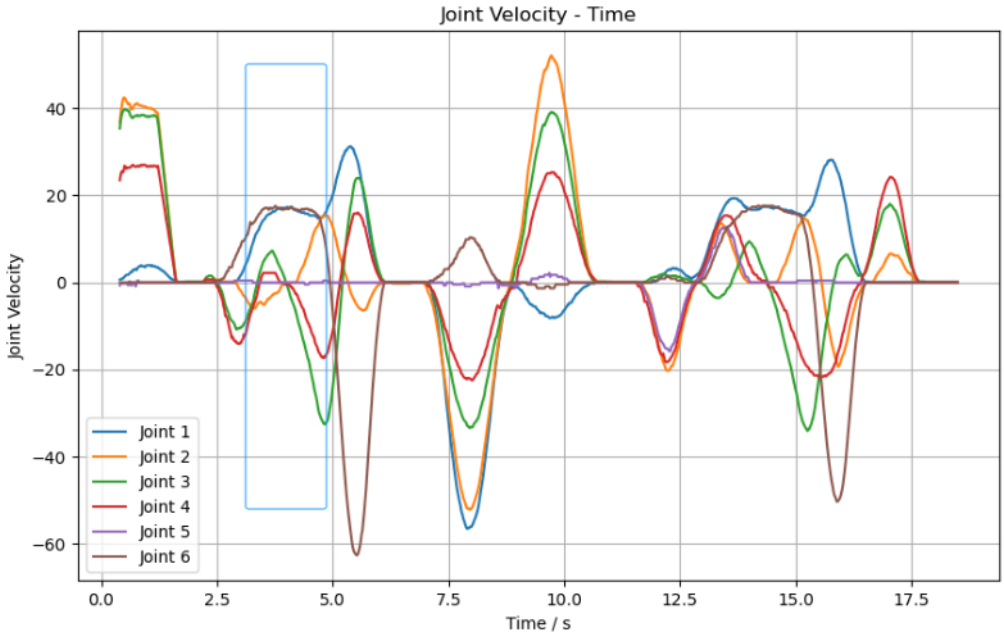
机械臂摆放物块的最终位置如下：



可以看到，两个物块被正确摆在了相应位置。

读取机械臂真实的角度与角速度，保存并可视化





可以看到**各关节角速度曲线都比较平滑**，尤其是**框出区域为直线轨迹规划AB段**，直线轨迹规划前后都**顺滑，没有停顿**，具体实现方式见后文问题解决部分。

3.问题解决

3.1 直线段轨迹规划不平滑

原本的代码对于直线轨迹规划为`rest to rest`，并且没有规定直线行进点的速度，导致在实验时出现机械臂行进卡顿和摇摆的现象。对此，我们采用下述策略

轨迹运动平滑：在进出直线轨迹规划的A\B点**设置对应的结束\初始角速度**使速度平滑，这里的角速度是由差分计算所得。由于直线轨迹规划将运动过程分成了1000段，我们取最初（最末）俩角度，计算$\frac{\Delta\theta}{\Delta t}$，可近似视为A\B处真实的角速度，并代入。实验结果证明这样修改的**效果极好。**

3.2 吸盘上下定位不准确

使用原先的代码操作实物机械臂时，可能由于电机扭矩不够或机械参数误差较大，机械臂在将物块从低处抬升到高处时（如将第二个物块从B点抬升至第一个物块之上）无法抬升到指定位置，总是会存在1-2cm的偏低误差；此外，在吸盘吸起物块之后，由于吸盘位置不准确，会产生物块拖地的现象。因此，我们在原有的五个状态$(start, end, A, B, 0)$基础之上添加了中间状态$lift, lay$。其中,$lift$位于$start$与A之间，使物块先抬升到lift的高度再平移到A，解决了拖地的问题；lay位于B与end之间，让物块先抬升到lay状态的高度再放下，则能有效忽略放置物块时的上下高度误差较大的问题（放下时会被下面的物块挡住）。

所以，添加了两个状态便可较好解决上述问题。

3.3 吸盘水平定位不准确

由于所给定位纸的范围过大，需先确定始末方块的位置在进行调参，以防机械臂吸不上还有初始位置放置偏差引起的最终物块偏移。此处我们的策略是先用铅笔大致描出机械臂抓取物块的合适位置，再将物块放置在所画位置上。这样便能保证每次机械臂抓取物块的位置大致统一。