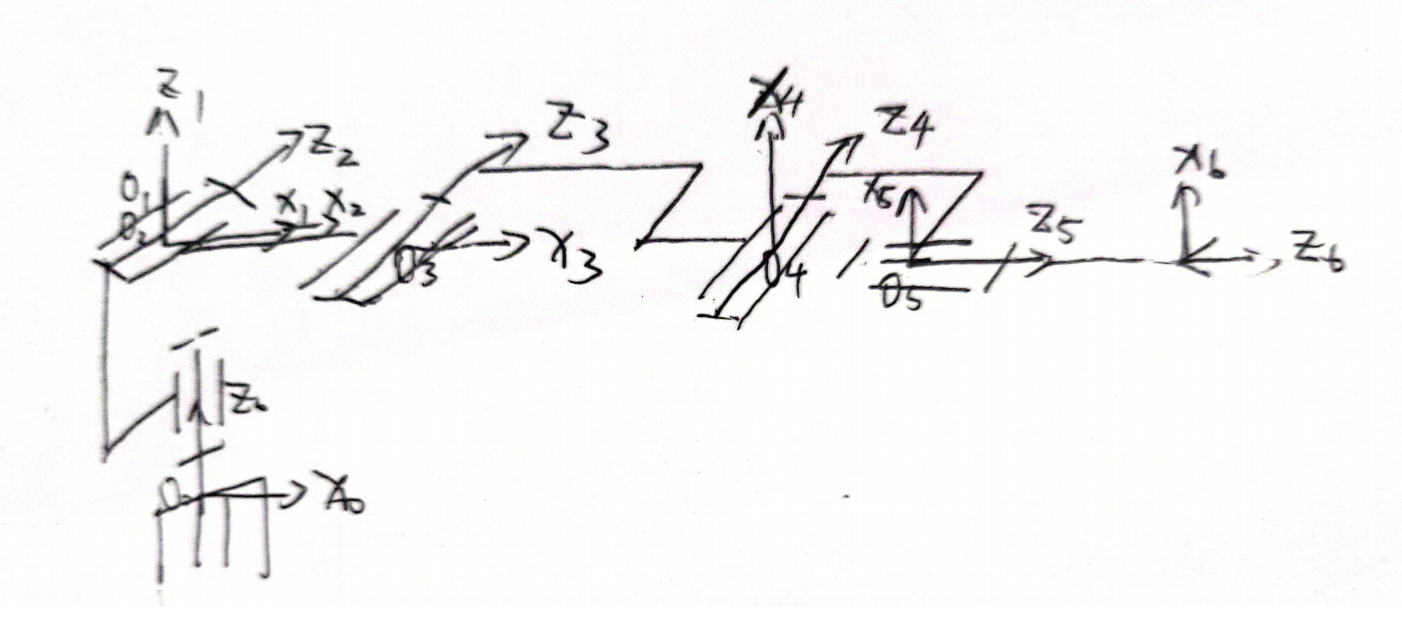


**1.机械臂结构分析**

机械臂作为移动机器人的重要执行机构，通常由多个关节、连杆和末端执行器组成。其设计需兼顾工作空间范围、灵活性、负载能力以及控制精度。



1 机构简图

**基座**作为机械臂的基础支撑，固定于底盘或工作平台，为整个系统提供稳定性。

**关节**通过电机驱动以实现旋转或线性运动，是机械臂灵活性的关键所在。

关节之间通过**连杆**连接，连杆长度及其布置方式直接影响机械臂的运动范围和操作性能。

**末端执行器**是夹爪，用于执行抓取等各种精细操作。这些部件协同工作，共同确保机械臂在不同应用场景中实现高效、精准的任务执行。

#### 2.机械臂正运动学模型

正运动学用于描述机械臂末端执行器的位置和姿态（位姿）相对于基座坐标系的关系。通过关节变量（如角度或位移）和连杆参数，使用齐次变换矩阵表示末端的位姿。

**2.1连杆参数定义（标准DH参数法）：**

连杆长度（a）：相邻关节轴线的距离。

连杆扭角（alpha）：相邻关节轴线的夹角。

连杆偏移（d）：关节轴线沿其方向的偏移量。

关节变量（theta）：关节旋转角度或线性位移。

以下是按照**标准DH参数法绘制的D-H参数表**

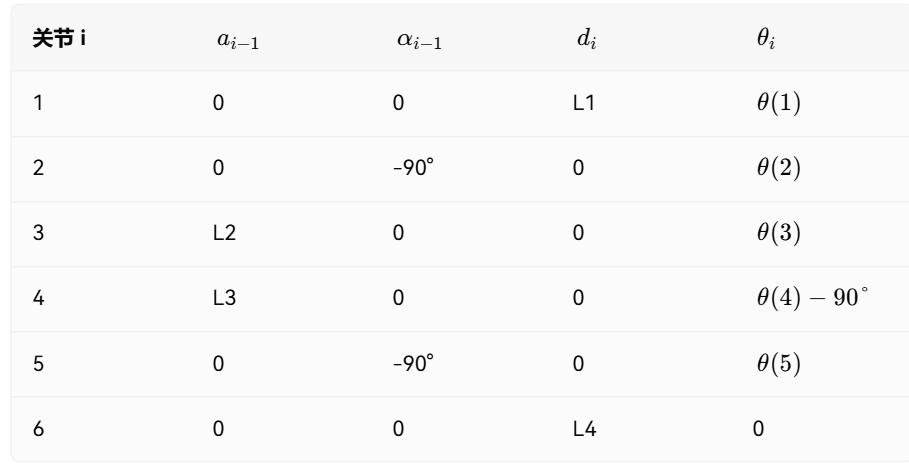


图2 D-H参数表

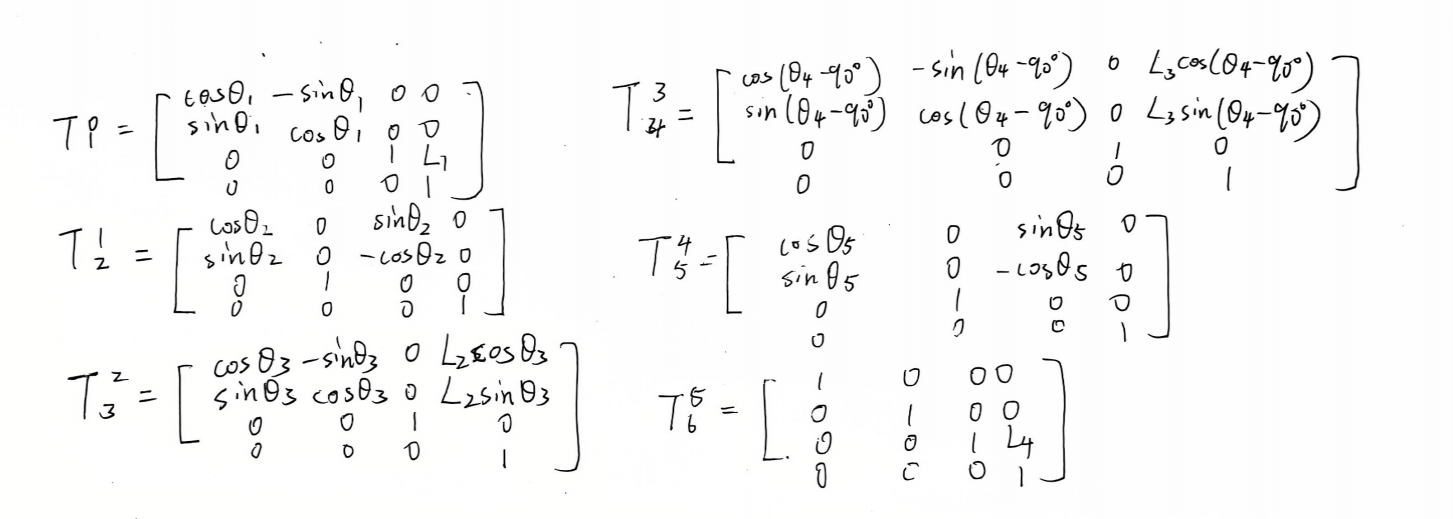
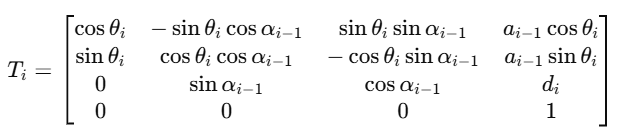
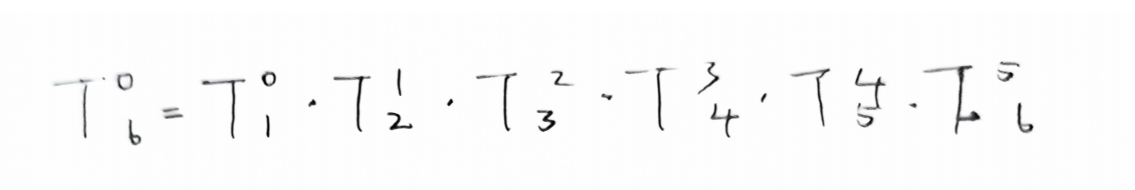
**2.2建立齐次变换矩阵：**

图2 变换矩阵

**2.3递推求解末端位姿：**  
通过逐层相乘各关节的变换矩阵，得到末端执行器的位姿：

图3 位姿求解

#### 3.机械臂逆运动学求解

逆运动学用于计算机械臂实现目标位姿所需的关节变量。逆运动学问题通常比正运动学复杂，因为存在多个解甚至无解的情况。

**逆运动学求解方法：**

**解析法：**  
通过几何推导和代数计算得出关节变量。适用于关节数较少、结构规则的机械臂（如3R或6R）。

优点：求解速度快，适合实时控制。

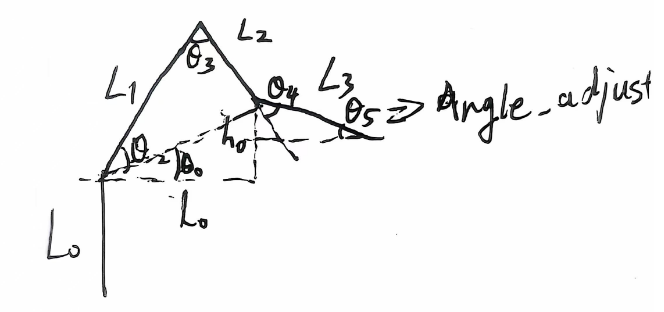
缺点：模型复杂性较高时难以推广。

**数值法：**  
基于迭代优化方法（如牛顿-拉夫逊法），通过目标位姿和初始关节配置逐步逼近解。

优点：适合任意复杂结构的机械臂。

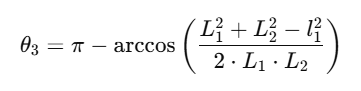
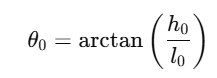
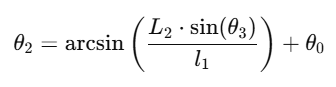
缺点：计算量大，可能收敛到局部最优解。

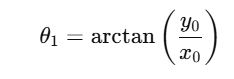
本课程所用机器人虽然有五个自由度，但是末端的舵机基本维持不动，所以约等于只有四自由度。由于自由度较低，所以我选择使用解析法求解，当解析法难以求解时，用数值法枚举。此方法可以高效地形成c++代码在单片机上运行。

图4 几何示意图

假设末端的目标坐标是(x0,y0,z0)。由于云台高度L0不变，我们首先用z0 = z0 - L0将高度缩短。

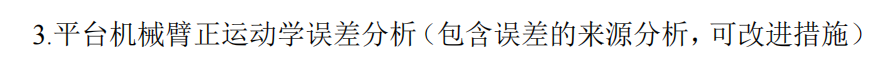
确定末端连杆和水平面的夹角theta5(ANGLE\_ADJUST)，由于第三个连杆的长度L3确定，第三个关节相对于原点的高度h0和距离l0也可以通过计算得知。

计算出上面两个变量后，机械臂从4个自由度退化为3个自由度，更易解。

其中theta1是云台转动角度

如果结果脱离几何条件约束，则判定为误解，然后会进一步计算下一个theta5的情况。

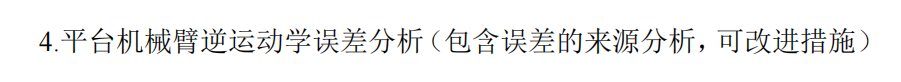




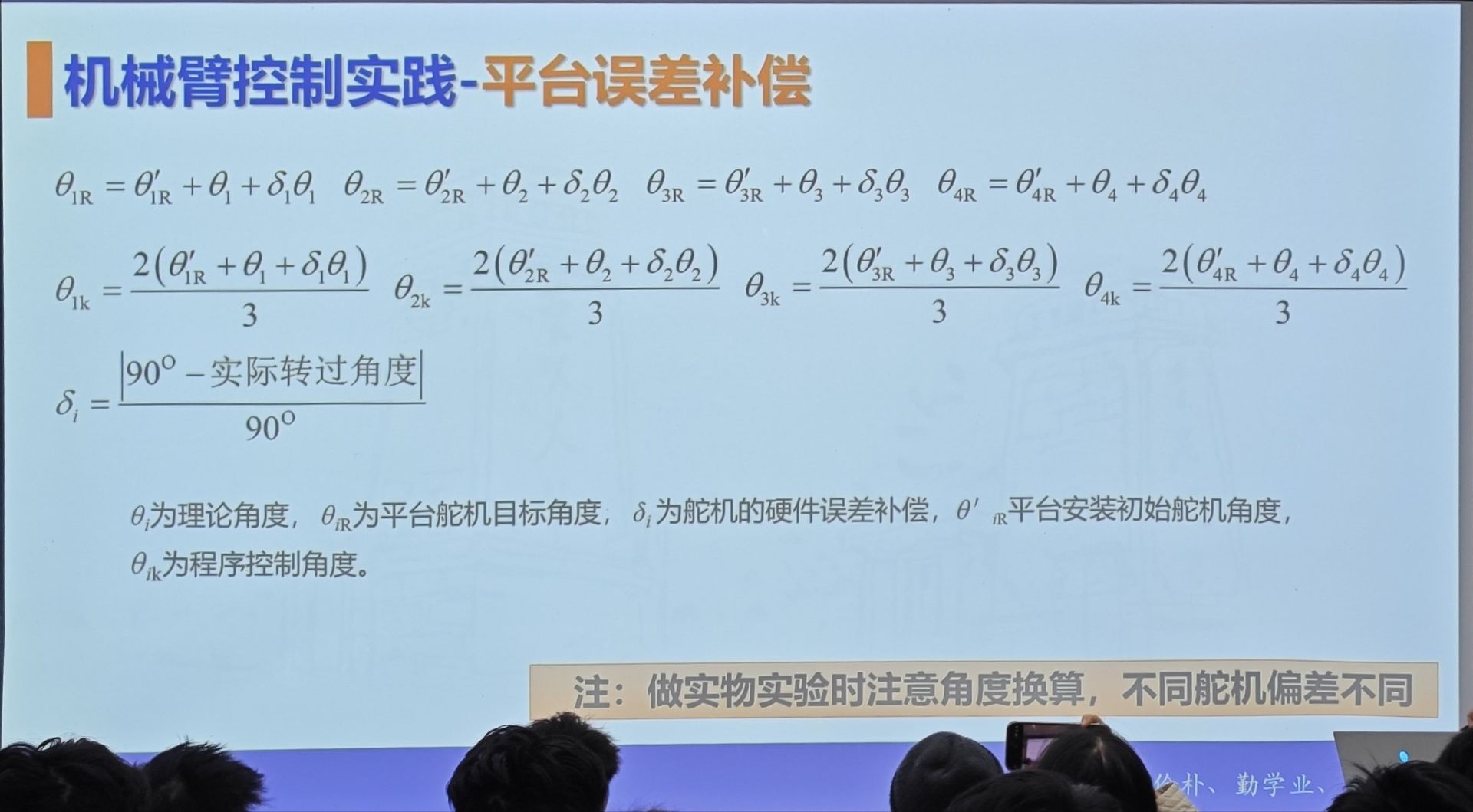
系统误差：卷尺精度量角器精度等

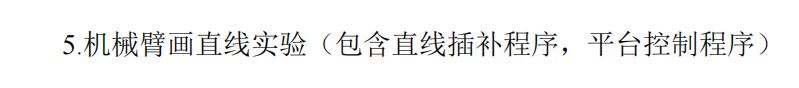
舵机舵盘松动

（上螺丝胶？）



由于单片机对浮点数的精确化能力有限，每个关节的误差累计，会对末端产生较大影响





% 示例使用

% 起点坐标

x0 = 0;

y0 = 0;

z0 = 0;

% 终点坐标

x1 = 10;

y1 = 10;

z1 = 10;

% 中间点的数量

n = 11;

% 调用函数

[xq, yq, zq] = linear\_interpolation\_3d(x0, y0, z0, x1, y1, z1, n);

% 输出结果

disp('插值点的坐标：');

for i = 1:length(xq)

fprintf('点 %d: (%f, %f, %f)\n', i, xq(i), yq(i), zq(i));

end

function [xq, yq, zq] = linear\_interpolation\_3d(x0, y0, z0, x1, y1, z1, n)

% x0, y0, z0: 起点坐标

% x1, y1, z1: 终点坐标

% n: 中间点的数量

% xq, yq, zq: 输出的插值点坐标数组

% 计算x, y, z的增量

delta\_x = (x1 - x0) / (n + 1);

delta\_y = (y1 - y0) / (n + 1);

delta\_z = (z1 - z0) / (n + 1);

% 初始化插值点坐标数组

xq = zeros(1, n + 2);

yq = zeros(1, n + 2);

zq = zeros(1, n + 2);

% 填充起点坐标

xq(1) = x0;

yq(1) = y0;

zq(1) = z0;

% 计算中间点坐标

for i = 1:n

xq(i + 1) = x0 + i \* delta\_x;

yq(i + 1) = y0 + i \* delta\_y;

zq(i + 1) = z0 + i \* delta\_z;

end

% 填充终点坐标

xq(n + 2) = x1;

yq(n + 2) = y1;

zq(n + 2) = z1;

end

void draw\_stroke(float begin\_X,float begin\_Y,float begin\_Z,float end\_X,float end\_Y,float end\_Z)//画横线

{

  float x[POINT\_NUM], y[POINT\_NUM], z[POINT\_NUM];

  float point\_begin[3] = {begin\_X,begin\_Y,begin\_Z}; // 第一个点的坐标

  float point\_end[3] = {end\_X,end\_Y,end\_Z}; // 第二个点的坐标

  for (int i = 0; i < POINT\_NUM; i++) {

    // 计算每个维度的插值 得到每个差值点的坐标

    x[i] = point\_begin[0] + ((point\_end[0] - point\_begin[0]) \* i / (POINT\_NUM-1));

    y[i] = point\_begin[1] + ((point\_end[1] - point\_begin[1]) \* i / (POINT\_NUM-1));

    z[i] = point\_begin[2] + ((point\_end[2] - point\_begin[2]) \* i / (POINT\_NUM-1));

    x[0] = point\_begin[0];

    y[0] = point\_begin[1];

    z[0] = point\_begin[2];

    x[POINT\_NUM-1] = point\_end[0];

    y[POINT\_NUM-1] = point\_end[1];

    z[POINT\_NUM-1] = point\_end[2];

  }

  for (int i = 0; i < POINT\_NUM; i++) {//开始遍历所有点 准备控制

    bool valid = false;//将所有的点设置为无效

    int current\_angle\_adjust = ANGLE\_ADJUST;//初始角度

    Angles\_OUT angles;

    while (!valid)

    {

      angles=backcalculate(int(x[i]), int(y[i]), int(z[i]),current\_angle\_adjust);//计算出当前的角度 //不对不对

      if (isnan(angles.theta1) || isnan(angles.theta2) || isnan(angles.theta3) || isnan(angles.theta4))//判断是否有解

      {

        if (current\_angle\_adjust > 0) {

          current\_angle\_adjust -= 1; // 减小偏移角度

          Serial.print("Invalid solution, reducing ANGLE\_ADJUST to: ");//调整到一个角度

          Serial.println(current\_angle\_adjust);

        }

        else {

          Serial.println("Cannot find valid solution, skipping this point.");

          break;//跳出while循环 舍弃这个点

        }

      }

      else //直到算出可行的解 如果可解 那么这个点有效

      {

        valid = true;

      }

    }

  // Angles\_OUT angles = backcalculate(x[i], y[i], z[i]);

  // control\_arm(angles.theta1,-angles.theta2,angles.theta3,angles.theta4,0,0);

  // delay(GENERAL\_GESTURE\_DELAY);

    if (valid) {//如果是一个有效的点

      Serial.print("Valid solution found with ANGLE\_ADJUST: ");

      Serial.println(current\_angle\_adjust);

      control\_arm(angles.theta1,-angles.theta2,angles.theta3,angles.theta4,0,0);//通过计算出的点 控制机械臂

      delay(GENERAL\_GESTURE\_DELAY);

    }

  }

}

float to\_deg(float angle)//弧度转角度

{ return angle\*180/PI ;}

float height\_adjust(float z,float angle\_adjust)//末端偏移

{

  float h=0;

  angle\_adjust=angle\_adjust\*PI/180;//角度转弧度

  h=z+L3\*sin(angle\_adjust);//和水平面的角度

  return h;

}

float length\_adjust(float angle\_adjust)

{return L3\*cos(angle\_adjust\*PI/180);}

Angles\_OUT backcalculate(float x,float y,float z,int angle\_adjust)

{

  Angles\_OUT angles;

  float l1=0;

  float h0=0;

  float theta0=0;

  float theta1=0;

  float theta2=0;

  float theta3=0;

  float theta4=0;

  float x0=x;

  float y0=y;

  float z0=z-L0;//减去底座高度

  h0=height\_adjust(z0,angle\_adjust);//得到第三个关节的高度

  float l0 = sqrt(pow(x0, 2) + pow(y0, 2)) - length\_adjust(angle\_adjust);//调整第三个关节的水平位移

  //l1 = sqrt(pow(l1, 2) + pow(h0, 2));//得到第三个关节到原点的距离 此段代码存疑

  l1 = sqrt(pow(l0, 2) + pow(h0, 2));

  theta0=atan(h0/l0);//第三个关节关于原点的夹角，是中间量

  // theta3=PI-asin(l1\*sin(theta2-theta0)/l2);

  angles.theta3 = PI - acos((pow(L1, 2) + pow(L2, 2) - pow(l1, 2)) / (2 \* L1 \* L2));//目前是锐角 不知其他的解怎么搞

  angles.theta2=(L2\*sin(angles.theta3 )/l1)+theta0;//第一根杆的水平夹角

  angles.theta1=atan(y0/x0);//云台转动夹角

  angles.theta4=-(PI-angles.theta2-(PI-angles.theta3)-(angle\_adjust\*PI/180));

  // Serial.print(theta4\*180/PI);

  angles.theta1 = to\_deg(angles.theta1);//全部转化为角度制

  angles.theta2 = to\_deg(angles.theta2);

  angles.theta3 = to\_deg(angles.theta3);

  angles.theta4 = to\_deg(angles.theta4);

  Serial.print("云台转动夹角=");

  Serial.println(angles.theta1);

  Serial.print("第一根杆的夹角=");

  Serial.println(angles.theta2);

  Serial.print("第二根杆的夹角=");

  Serial.println(angles.theta3);

  Serial.print("第三根杆的夹角=");

  Serial.println(angles.theta4);

  return angles;

}