Mini Project 1 —ABB IRB460 Robot

苗子良，周晋徽，刘洪磊，林卓垠

2021 年 3 月 22 日

目录

1 Abstract **1**

2 Coordinate transformation g_{ab} **2**

3 spatial velocity V_{ab} **6**

4 Torque of joint points(static) **12**

5 Torque of joint points(movement) **17**

6 Work distribution **21**

1 Abstract

The IRB 460 is the world' s fastest palletizing robot, capable of significantly shortening cycle times and raising productivity for end-of line and bag palletizing. Here, by using MATLAB amd MSC Adams, a series of important physical quantity are analyzed. After analysis, the coordinate transformation g_{ab},spatial velocity $\hat{V}_{ab}$, three joint torques$\tau_{1(static)}$,$\tau_{2(static)}$,$\tau_{3(static)}$, and end-effector $\tau_{end(static)}$ at static configuration are computed.In the end, given the joint trajectories (angle, speed and acceleration) of the robot, we computed the three joint torques$(\tau_{1(movement)},\tau_{2(movement)},\tau_{3(movement)})$.

2 Coordinate transformation g_{ab}

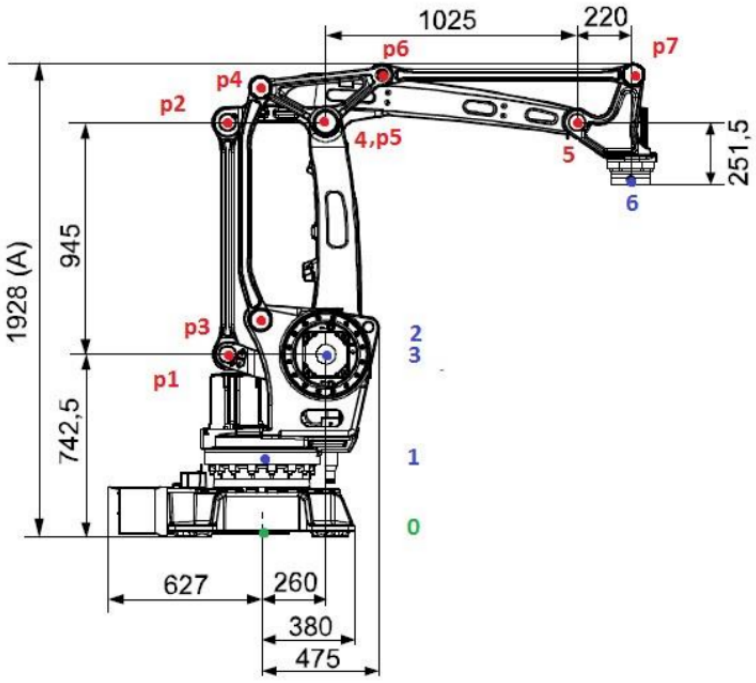

图 1: The Universe

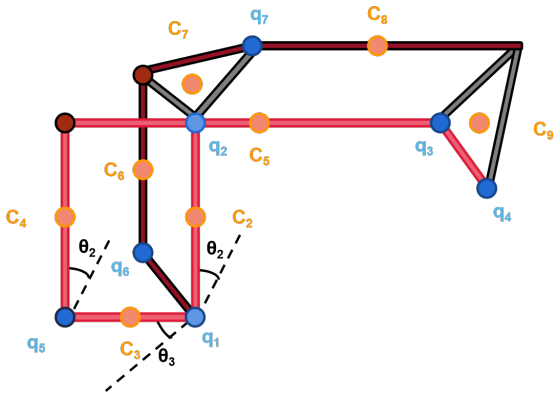

图 2: the comparison diagram of each bars

At the beginning:

$$\theta_1 = 0$$
$$\theta_2 = 40$$
$$\theta_3 = 0$$
$$q_{00} = [0, 0, 0]$$
$$q_{10} = [0, 260, 742, 5]$$
$$q_{20} = [0, 260, 742.5 + 945]$$
$$q_{30} = [0, 260 + 1025, 742.5 + 945]$$

Compute the theoretical formula:

$$q_1 = q_{10} \tag{1}$$
$$q_2 = R(q_{10}, \theta_1) q_{20} \tag{2}$$
$$\hat{\xi}_1 = \begin{bmatrix} \hat{z} & q_{10} \times z \\ 0 & 0 \end{bmatrix} R(q_{10}, \theta_1) = e^{\hat{z}\theta_1} = \begin{bmatrix} e^{\hat{z}\theta_1} & (I - e^{\hat{z}\theta_1}) q_{10} \\ 0 & 0 \end{bmatrix} \tag{3}$$
$$g_{ab} = R(q_{10}, \theta_1) g_{ab_0} \tag{4}$$

Induce to all class:

$$\hat{\xi}_i = e^{\hat{z}\theta_i} = \begin{bmatrix} e^{\hat{z}\theta_i} & (I - e^{\hat{z}\theta_i}) q_{i0} \\ 0 & 0 \end{bmatrix} \tag{5}$$
$$\tag{6}$$

Move to four link robot:

$$g_{ab} = e^{\xi_1 \hat{\theta}_1} e^{\xi_2 \hat{\theta}_2} e^{\xi_3 \hat{\theta}_3} e^{\xi_4 \hat{\theta}_4} g_{ab_0} \tag{7}$$

Here is the Matlab calcution:

```
%clc; clear all; close all;
theta1 = 0;
theta2 = 40;
theta3 = 0;
global q00 q10 q20 q30 V1 V2
```

```
q00 = [0; 0; 0];
q10 = [0; 260; 742.5];
q20 = [0; 260; 742.5+945];
q30 = [0; 260+1025; 742.5+945];
Q4 = CoordinateTransformation(theta1,theta2,theta3) *
    [0;0;0;1];
q40 = Q4(1:3,:);

function T = CoordinateTransformation(theta1, theta2,
    theta3)
     global q00 q10 q20 q30
     theta1 = theta1/180*pi;
     theta2 = theta2/180*pi;
     theta3 = theta3/180*pi;
     gab0 = [eye(3), [0;1025+260+220;945+742.5-217.5];0
          0 0 1];
     gab = RotationZ(q00, theta1) * RotationX(q10,
         theta2) * RotationX(q20,(theta3-theta2)) *
         RotationX(q30,(-theta3)) * gab0;
     T = gab;
end
function R = RotationX(q_axis,theta)
     x_hat = [0 0 0; 0 0 -1; 0 1 0];
     x = [1; 0; 0];
     % Caculate Method 1 (Use method "expm")
% twist = [x_hat, cross(q_axis, x); 0000];
% R = expm(twist * theta);
     % Caculate Method 2 (Simplify by Rodrigues Formula)
     r = eye(3) + x_hat * sin(theta) + (x_hat)^2 * (1 -
          cos(theta));
     R = [r, (eye(3) - r) * q_axis; 0 0 0 1];
end
function R = RotationZ(q_axis,theta)
```

```
    z_hat = [0 -1 0; 1 0 0; 0 0 0];
    z = [0; 0; 1];
    r = eye(3) + z_hat * sin(theta) + (z_hat)^2 * (1 -
        cos(theta));
    R = [r, (eye(3) - r) * q_axis; 0 0 0 1];
end
```

Example of task 1:

$\theta_1 = 25d$

$\theta_2 = 40d$

$\theta_3 = 20d$

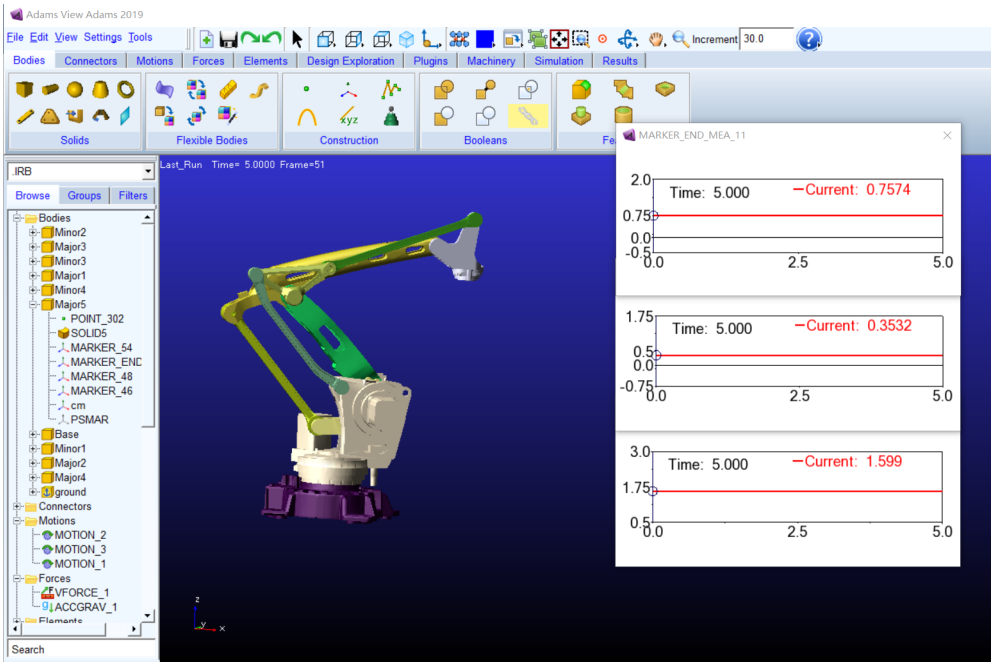

图 3: simulation of task 1(1)

```
%% Task1 - by Group8 MIAO Ziliang 11911901
clc; clear all; close all;
theta1 = 25;
theta2 = 40;
theta3 = 20;
global q00 q10 q20 q30 gab

%输入各个joint
q00 = 0.001*[0; 0; 0];
q10 = 0.001*[0; 260; 742.5];
q20 = 0.001*[0; 260; 742.5+945];
```

F
G
gab
J
m1
m2
m3
m4
m5
m6
m7
m8
m9
q00
q10
q100
q20
q30
q4
Q4

图 4: simulation of task 1(2)

3 spatial velocity V_{ab}

In chapter2, we have computed the coordinate transformation g_{ab},

$$g_{ab} = e^{\xi_1\hat{\theta}_1}e^{\xi_2\hat{\theta}_2}e^{\xi_3\hat{\theta}_3}e^{\xi_4\hat{\theta}_4}g_{ab_0}$$

$$\begin{aligned}\hat{V}_{ab} = \dot{g_{ab}}g_{ab}^{-1} &= \hat{\xi}\dot{\theta}_1 + e^{\hat{\xi_1}\theta_1}\hat{\xi_2}e^{-\hat{\xi_1}\theta_1}\dot{\theta}_2 \\ &+ (e^{\hat{\xi_1}\theta_1}e^{\hat{\xi_2}\theta_2})\hat{\xi_3}(e^{-\hat{\xi_2}\theta_2}e^{-\hat{\xi_1}\theta_1})\dot{\theta}_3 \\ &+ (e^{\hat{\xi_1}\theta_1}e^{\hat{\xi_2}\theta_2}e^{\hat{\xi_3}\theta_3})\hat{\xi_4}(e^{-\hat{\xi_3}\theta_3}e^{-\hat{\xi_2}\theta_2}e^{-\hat{\xi_1}\theta_1})\dot{\theta}_4\end{aligned}$$

Then computed the Jacobian matrix:

$$g\hat{V}g^{-1} = Ad_g\hat{V} \tag{8}$$

$$Ad_g\hat{V} = \begin{bmatrix} R, \hat{p}R \\ 0, R \end{bmatrix} \tag{9}$$

for g= $\begin{bmatrix} R, p \\ 0, 1 \end{bmatrix}$ a rigid motion and V= $\begin{bmatrix} v \\ \omega \end{bmatrix}$ a twist.
Here we taked the hat of $\hat{V}_{ab}$ off.

$$V_{ab} = \xi_1\dot{\theta}_1 + Ad_{e^{\hat{\xi_1}\theta_1}}\xi_2\dot{\theta}_2 + Ad_{e^{\hat{\xi_1}\theta_1}e^{\hat{\xi_2}\theta_2}}\xi_3\dot{\theta}_3 + Ad_{e^{\hat{\xi_1}\theta_1}e^{\hat{\xi_2}\theta_2}e^{\hat{\xi_3}\theta_3}}\xi_4\dot{\theta}_4 \tag{10}$$

in which

$$\xi_2^{'} = Ad_{e^{\hat{\xi_1}\theta_1}}\xi_2$$
$$\xi_3^{'} = Ad_{e^{\hat{\xi_1}\theta_1}e^{\hat{\xi_2}\theta_2}}\xi_3$$
$$\xi_4^{'} = Ad_{e^{\hat{\xi_1}\theta_1}e^{\hat{\xi_2}\theta_2}e^{\hat{\xi_3}\theta_3}}\xi_4$$

$$V_{ab} = [\xi_1, \xi_2^{'}, \xi_3^{'}, \xi_4^{'}] \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{bmatrix} \tag{11}$$

Here is the Matlab calcution:

```
%theta1_dot = 0;
theta2_dot = 8;
theta3_dot = 0;

%在此情况下计算
V = SpatialVelocity(theta1, theta2, theta3, theta1_dot
    , theta2_dot, theta3_dot);
disp(V);

%以下操作为了方便 ADAMS 仿真验证（提取变换矩阵的最后一列中的前
    三行：坐标）
V40 = V * [q40;0];
v40 = V40(1:3,:);
function V = SpatialVelocity(theta1, theta2, theta3,
    theta1_dot, theta2_dot, theta3_dot)
    %声明变量
    global q00 q10 q20 q30 V1
    %进行角度弧度制转换
    theta1 = theta1/180*pi;
    theta2 = theta2/180*pi;
    theta3 = theta3/180*pi;
    theta1_dot = theta1_dot/180*pi;
    theta2_dot = theta2_dot/180*pi;
    theta3_dot = theta3_dot/180*pi;

    %输入节点旋转轴
    x = [1; 0; 0];
    z = [0; 0; 1];

    %输入单位角速度
    x_hat = [0 0 0; 0 0 -1; 0 1 0];
    z_hat = [0 -1 0; 1 0 0; 0 0 0];
```

```
30
31      %计算旋转矩阵
32      R1 = RotationZ(q00, theta1);
33      R2 = RotationX(q10,theta2);
34      R3 = RotationX(q20,(theta3-theta2));
35      R4 = RotationX(q30,(-theta3));
36
37      %计算旋量
38      twist1 = [z_hat, cross(q00,z);0 0 0 0];
39      twist2 = [x_hat, cross(q10,x);0 0 0 0];
40      twist3 = [x_hat, cross(q20,x);0 0 0 0];
41      twist4 = [x_hat, cross(q30,x);0 0 0 0];
42
43      %计算 gab0、gab_dot gab_inverse
44      gab0 = [eye(3), [0;1025+260+220;945+742.5-217.5];0
            0 0 1];
45      gab_dot = (twist1 * theta1_dot * R1 * R2 * R3 * R4
            ...
46      + R1 * twist2 * theta2_dot * R2 * R3 * R4...
47      +R1 * R2 * twist3 * (theta3_dot - theta2_dot) * R3
            * R4...
48      + R1 * R2 * R3 * twist4 * (-theta3_dot) * R4) *
            gab0;
49      gab_inverse = inv(gab0) * inv(R4) * inv(R3) * inv(
            R2) * inv(R1);
50
51      %最后计算空间速度
52      V1 = gab_dot * gab_inverse;
53      V = twist1 * theta1_dot + R1*twist2*theta2_dot*inv
            (R1) + ...
54          R1*R2*twist3*(theta3_dot-theta2_dot)*inv(R2)*
                inv(R1) + ...
55          R1*R2*R3*twist4*(-theta3_dot)*inv(R3)*inv(R2)*
```

```
            inv(R1);
56      %V = twist2 *
            theta2_dot + R2 * twist3 * inv(R2) * (theta3_dot - theta2_dot) +
            (R2 * R3) * twist4 * (inv(R3) * inv(R2)) * (-theta3_dot);
57  end
58
59  %坐标系变换函数
60  function T = CoordinateTransformation(theta1, theta2,
        theta3)
61      global q00 q10 q20 q30
62
63      %进行角度弧度制转换
64      theta1 = theta1/180*pi;
65      theta2 = theta2/180*pi;
66      theta3 = theta3/180*pi;
67
68      %输入 end effector 的初始坐标
69      gab0 = [eye(3), [0;1025+260+220;945+742.5-217.5];0
            0 0 1];
70
71      %引用 Rotation 函数进行变换叠加
72      gab = RotationZ(q00, theta1) * RotationX(q10,
            theta2) * RotationX(q20,(theta3-theta2)) *
            RotationX(q30,(-theta3)) * gab0;
73      T = gab;
74  end
75
76  %计算 twist:X
77  function t = twist(q_axis)
78   %x 代表旋转轴
79      x = [1; 0; 0];
80      %x_hat
81      x_hat = [0 0 0; 0 0 -1; 0 1 0];
82
```

```
    %t = [x_hat, cross(q_axis, x); 0000];
end

%旋转矩阵计算函数: X
function R = RotationX(q_axis, theta)
    %x 代表旋转轴
    x = [1; 0; 0];
    %x_hat
    x_hat = [0 0 0; 0 0 -1; 0 1 0];

    %
    %r = eye(3) + x_hat * sin(theta) + (x_hat)^2 * (1 - cos(theta));
    R = [r, (eye(3) - r) * q_axis; 0 0 0 1];
end
```

Example of task2

$\theta_1 = 0d \qquad \dot{\theta}_1 = 0d$

$\theta_2 = 40d \qquad \dot{\theta}_2 = 8d$

$\theta_1 = 20d \qquad \dot{\theta}_1 = 4d$

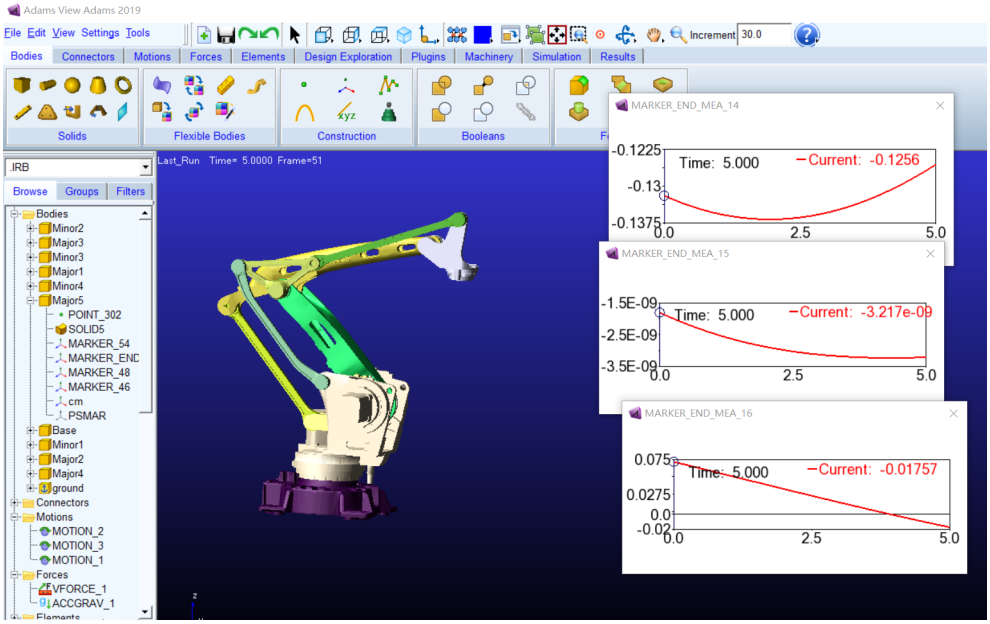

图 5: simulation of task 2(1)

图 6: simulation of task 2(2)

4 Torque of joint points(static)

Since we have already computed twists in task 1 and 2, we have to compute the J matrix in the following code:

```
%输入 payload
```

```
F = [0;0;-800;0;0;0];

 %输入节点旋转轴
 x = [1;  0;  0];
 z = [0;  0;  1];

%输入单位角速度
x_hat = [0  0  0;  0  0  -1;  0  1  0];
z_hat = [0  -1  0;  1  0  0;  0  0  0];

%计算旋量
twist1 = [z_hat, cross(q00,z);0  0  0  0];
twist2 = [x_hat, cross(q10,x);0  0  0  0];
twist3 = [x_hat, cross(q20,x);0  0  0  0];
twist4 = [x_hat, cross(q30,x);0  0  0  0];

%计算伴随矩阵
Ad1 = Adjoint(RotationZ(q00, theta1));
Ad2 = Adjoint(RotationX(q10,theta2));
Ad3 = Adjoint(RotationX(q20,(theta3-theta2)));

twist1_ = [cross(q00,z);z];
twist2_ = [cross(q10,x);x];
twist3_ = [cross(q20,x);x];
twist4_ = [cross(q30,x);x];

%计算 "Twist in new location"
twist1_new = twist1_;
twist2_new = (Ad1) * twist2_;
twist3_new = (Ad1*Ad2) * twist3_;
twist4_new = (Ad1*Ad2*Ad3) * twist4_;

%输出雅可比矩阵 (需要简化为三个自由度——末端为旋转)
```

```
%J = [twist1_new, twist2_new, twist3_new, twist4_new];
J = [twist1_new, twist2_new-twist3_new, twist3_new-
    twist4_new];
%此处可能需要对矩阵进行化简（至三个自由度）
disp(J);
% Vab = J*[theta1_dot; theta2_dot; theta3_dot];
% Vab_hat2 = [[0, -Vab(6), Vab(5); Vab(6), 0, -Vab(4);
-Vab(5), Vab(4), 0], [Vab(1); Vab(2); Vab(3)]; 0, 0, 0, 0];
% V402 = Vab_hat2 * [q40; 1];

%计算各个质心的坐标表达式
syms theta1 theta2 theta3
c1 = RotationZ(c10, theta1)*[c10; 1];
c2 = RotationZ(c10, theta1)*RotationX(q10, theta2)*[c20
    ; 1];
c3 = RotationZ(c10, theta1)*RotationX(q10, theta3)*[c30
    ; 1];
c4 = RotationZ(c10, theta1)*RotationX(q10, theta3)*
    RotationX(q50, theta2-theta3)*[c40; 1];
c5 = RotationZ(c10, theta1)*RotationX(q10, theta2)*
    RotationX(q20,-theta2+theta3)*[c50; 1];
c6 = RotationZ(c10, theta1)*RotationX(q60, theta2)*[c60
    ; 1];
c7 = RotationZ(c10, theta1)*RotationX(q10, theta2)*
    RotationX(q20,-theta2)*[c70; 1];
c8 = RotationZ(c10, theta1)*RotationX(q10, theta2)*
    RotationX(q20,-theta2)*RotationX(q70, theta3)*[c80
    ; 1];
c9 = RotationZ(c10, theta1)*RotationX(q10, theta2)*
    RotationX(q20,-theta2+theta3)*RotationX(q30,-theta3
    )*[c90; 1];

%计算静态下重力势能
```

```
57 %先声明各个质量
58 m1 = 102.7282598605;
59 m2 = 12.4193410149;
60 m3 = 5.2824118449;
61 m4 = 2.9463248039;
62 m5 = 9.6977245306;
63 m6 = 1.4433507166;
64 m7 = 2.8514044732;
65 m8 = 1.4155689868;
66 m9 = 6.0134646985;
67 G = 9.80665;
68 V = G*(m1*[0;0;1;0]'*c1 + m2*[0;0;1;0]'*c2 + m3
       *[0;0;1;0]'*c3 + m4*[0;0;1;0]'*c4 + m5*[0;0;1;0]'*
       c5 + ...
69      + m6*[0;0;1;0]'*c6 + m7*[0;0;1;0]'*c7 + m8
            *[0;0;1;0]'*c8 + m9*[0;0;1;0]'*c9);
70
71 %计算力矩
72 V_diff_theta1 = diff(V,theta1);
73 V_diff_theta2 = diff(V,theta2);
74 V_diff_theta3 = diff(V,theta3);
75 V_diff_theta = [V_diff_theta1;V_diff_theta2;
       V_diff_theta3];
76 torque_nogravity = J'*F;
77 torque_gravity = -V_diff_theta + J'*F;
78 %赋值
79 vpa(subs(torque_gravity,{theta1,theta2,theta3
       },{0,0,0}))
80
81 %计算对应的伴随矩阵
82 function A = Adjoint(Rotation)
83      R = Rotation([1,2,3],[1,2,3]);
84      pI = Rotation([1,2,3],(4));
```

```
85      p = [0,-pI(3),pI(2);pI(3),0,-pI(1);-pI(2),pI(1)
            ,0];
86      A = [R,p*R;zeros(3),R];
87  end
```

Example of task3

$F_x = 100N$ $F_y = 100N$

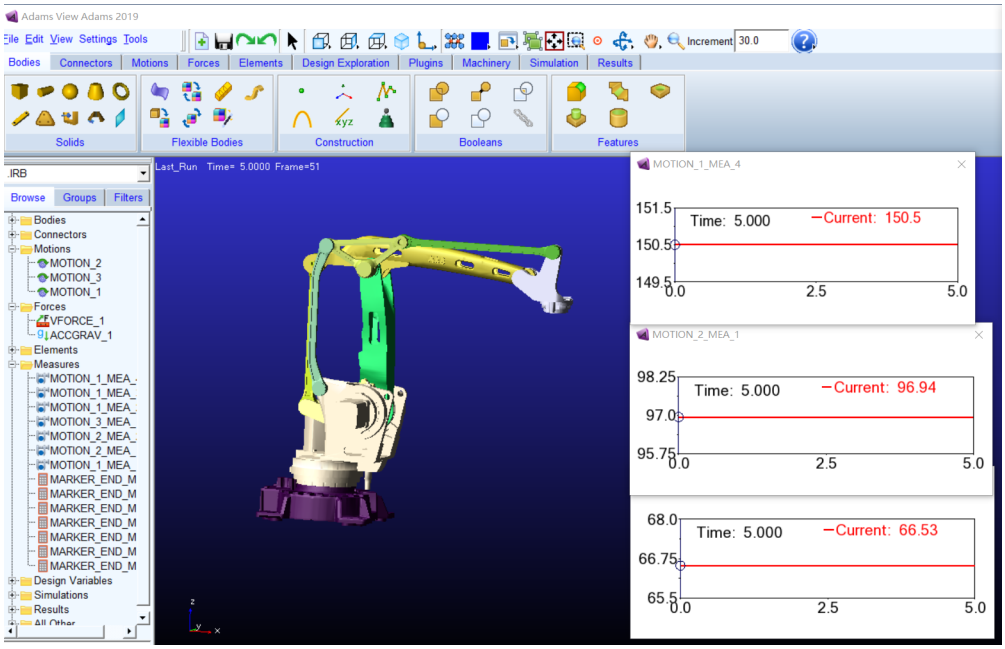

图 7: simulation of task 3

5 Torque of joint points(movement)

In task3, We have computed:

$$\tau = \frac{\partial V}{\partial \theta} + J^T F \qquad \theta = (\theta_1, \theta_2, \theta_3)^T$$

When computed τ at movement:

$$\tau = \frac{d}{dt}(\frac{\partial L}{\partial \dot{\theta}}) - \frac{\partial L}{\partial \theta} \tag{12}$$

First we compute T and V. Here is the comparison diagram of each bars:

for bars in a chain: (13)

$$c_n = \prod_{i=1}^{n} e^{\hat{\xi_i}\theta_i} c_{i0} \tag{14}$$

$$\dot{c}_n = \begin{cases} \dot{c}_1 = \dot{\theta}_1 \hat{\xi}_1 c_1 & n = 1 \\ \dot{c}_n = (\dot{\theta}_1 \hat{\xi}_1 + \sum_{i=2}^{n} \dot{\theta}_i \hat{\xi_i'}) c_n \end{cases} \tag{15}$$

$$V = -\sum_{i=1}^{n} m_i g c_{i,y} \tag{16}$$

$$T = \sum_{i=1}^{n} \frac{1}{2} m_i \dot{c}_i^T \dot{c}_i \tag{17}$$

$$\frac{\partial L}{\partial \theta} = \frac{\partial (T - V)}{\partial \theta} \tag{18}$$

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{\theta}}) = \frac{d}{dt}\frac{\partial T}{\partial \dot{\theta}} \tag{19}$$

The Euler-Langrange equation with m links, n joints and p actuators is :

$$\mathcal{L}(i,j)\frac{d}{dt}(\frac{\partial L_i}{\partial \dot{\theta}_j}) - \frac{\partial L_i}{\partial \theta_j}, \qquad i = 1, 2, ..., m, j = 1, ...n \tag{20}$$

$$\tau_k = \sum_{i=1,j=1}^{m,n} \mathcal{L}(i,j)\frac{\partial \theta_j}{\widetilde{\theta}_k} \tag{21}$$

Here comes the matlab compute:

```
1
2  %Task4 - by Group8 Liu HongLei 11912905
3
4  %计算动能
5  %先声明各个 joint 角速度
6  syms dot_theta1 dot_theta2 dot_theta3
7  dot_c1 = diff(c1,theta1)*dot_theta1 + diff(c1,theta2)*
       dot_theta2 + diff(c1,theta3)*dot_theta3;
8  dot_c2 = diff(c2,theta1)*dot_theta1 + diff(c2,theta2)*
       dot_theta2 + diff(c2,theta3)*dot_theta3;
9  dot_c3 = diff(c3,theta1)*dot_theta1 + diff(c3,theta2)*
       dot_theta2 + diff(c3,theta3)*dot_theta3;
10 dot_c4 = diff(c4,theta1)*dot_theta1 + diff(c4,theta2)*
       dot_theta2 + diff(c4,theta3)*dot_theta3;
11 dot_c5 = diff(c5,theta1)*dot_theta1 + diff(c5,theta2)*
       dot_theta2 + diff(c5,theta3)*dot_theta3;
12 dot_c6 = diff(c6,theta1)*dot_theta1 + diff(c6,theta2)*
       dot_theta2 + diff(c6,theta3)*dot_theta3;
13 dot_c7 = diff(c7,theta1)*dot_theta1 + diff(c7,theta2)*
       dot_theta2 + diff(c7,theta3)*dot_theta3;
14 dot_c8 = diff(c8,theta1)*dot_theta1 + diff(c8,theta2)*
       dot_theta2 + diff(c8,theta3)*dot_theta3;
15 dot_c9 = diff(c9,theta1)*dot_theta1 + diff(c9,theta2)*
       dot_theta2 + diff(c9,theta3)*dot_theta3;
16
17 T = 0.5*m1*(dot_c1)'*(dot_c1) + 0.5*m2*(dot_c2)'*(
       dot_c2) + 0.5*m3*(dot_c3)'*(dot_c3) + 0.5*m4*(
       dot_c4)'*(dot_c4) + 0.5*m5*(dot_c5)'*(dot_c5) +
       0.5*m6*(dot_c6)'*(dot_c6) + 0.5*m7*(dot_c7)'*(
       dot_c7) + 0.5*m8*(dot_c8)'*(dot_c8) + 0.5*m9*(
       dot_c9)'*(dot_c9);
18 %动能对角速度求导
19 T_diff_dot_theta1 = diff(T,dot_theta1);
```

```
T_diff_dot_theta2 = diff(T,dot_theta2);
T_diff_dot_theta3 = diff(T,dot_theta3);
T_diff_dot_theta = [T_diff_dot_theta1;
    T_diff_dot_theta2;T_diff_dot_theta3;];
%动能对角速度求导再对 t 求导
%先声明角加速度
ddot_theta1 = 0*pi/180;
ddot_theta2 = 1*pi/180;
ddot_theta3 = 0*pi/180;
T_diff_t1 = diff(T_diff_dot_theta1,theta1)*dot_theta1
   + diff(T_diff_dot_theta1,theta2)*dot_theta2 + ...
    + diff(T_diff_dot_theta1,theta3)*dot_theta3 + diff
        (T_diff_dot_theta1,dot_theta1)*ddot_theta1 +
        ...
    + diff(T_diff_dot_theta1,dot_theta2)*ddot_theta2 +
         diff(T_diff_dot_theta1,dot_theta3)*ddot_theta3
        ;
T_diff_t2 = diff(T_diff_dot_theta2,theta1)*dot_theta1
   + diff(T_diff_dot_theta2,theta2)*dot_theta2 + ...
    + diff(T_diff_dot_theta2,theta3)*dot_theta3 + diff
        (T_diff_dot_theta2,dot_theta1)*ddot_theta1 +
        ...
    + diff(T_diff_dot_theta2,dot_theta2)*ddot_theta2 +
         diff(T_diff_dot_theta2,dot_theta3)*ddot_theta3
        ;
T_diff_t3 = diff(T_diff_dot_theta3,theta1)*dot_theta1
   + diff(T_diff_dot_theta3,theta2)*dot_theta2 + ...
    + diff(T_diff_dot_theta3,theta3)*dot_theta3 + diff
        (T_diff_dot_theta3,dot_theta1)*ddot_theta1 +
        ...
    + diff(T_diff_dot_theta3,dot_theta2)*ddot_theta2 +
         diff(T_diff_dot_theta3,dot_theta3)*ddot_theta3
        ;
```

```
37 T_diff_t = [T_diff_t1;T_diff_t2;T_diff_t3];
38
39 %动能对角度求导
40 T_diff_theta1 = diff(T,theta1);
41 T_diff_theta2 = diff(T,theta2);
42 T_diff_theta3 = diff(T,theta3);
43 T_diff_theta = [T_diff_theta1;T_diff_theta2;
       T_diff_theta3;];
44 %力矩
45 torque_dynamics = T_diff_t - T_diff_theta +
       V_diff_theta;
46 %赋值
47 vpa(subs(torque_dynamics,{theta1,theta2,theta3,
       dot_theta1,dot_theta2,dot_theta3},{0*pi/180,17.5*pi
       /180,0*pi/180,0*pi/180,1*pi/180,0*pi/180}))
```

Example of task4

$\theta_1 = 0d \quad \dot{\theta}_1 = 0d \quad \ddot{\theta}_1 = 0d$

$\theta_2 = 17.5d \quad \dot{\theta}_2 = 1d \quad \ddot{\theta}_2 = 1d$

$\theta_3 = 22.5d \quad \dot{\theta}_1 = 2d \quad \ddot{\theta}_1 = 1d$

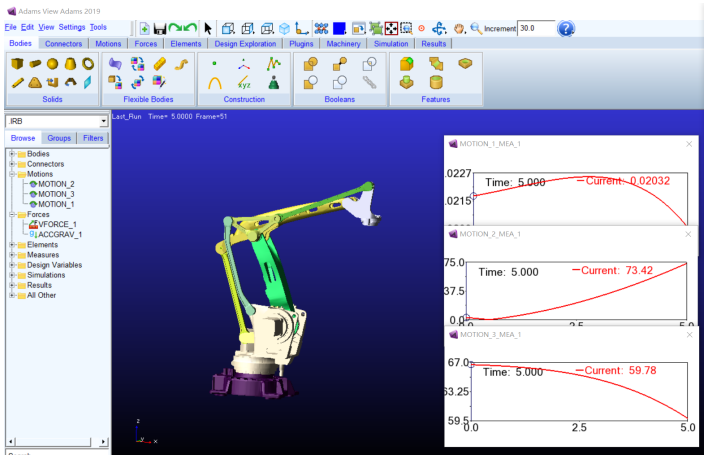

图 8: simulation of task4

6 Work distribution

Group 8 Work Distributions							
Member	Task1(%)	Task2(%)	Task3(%)	Task4(%)	Task5(%)	Vedio	Report
苗子良	60	40	10	10	55	拍摄	副编
周晋徽	20	40	20	20	10	导演	副编
林卓垠	10	10	30	30	10	拍摄	主编
刘洪磊	10	10	40	40	25	拍摄	副编