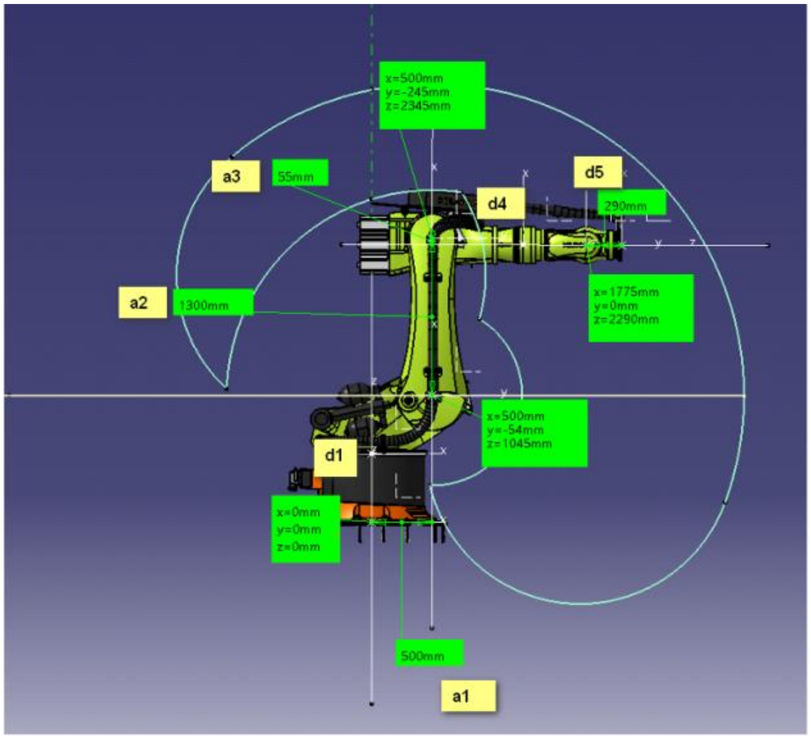
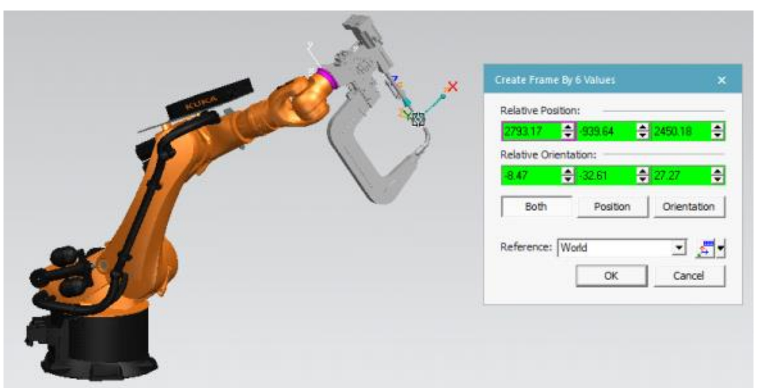
11. KUKA Robot Inverse Kinematic Calculation

Saturday, June 4, 2022 5:28 PM

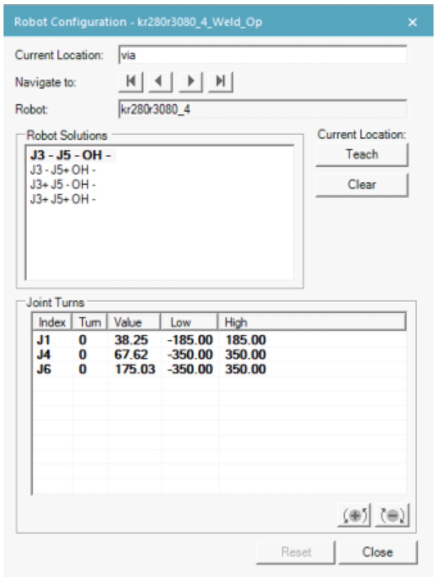


Take KUKA R3080 as example, all the parameters can be got from KUKA gov website, or you can download the cad to measure in the CATIA environment;

The calculation below , all the results are verified at Tecnomatix Process simulate , the calculation results are matching with simulation result;



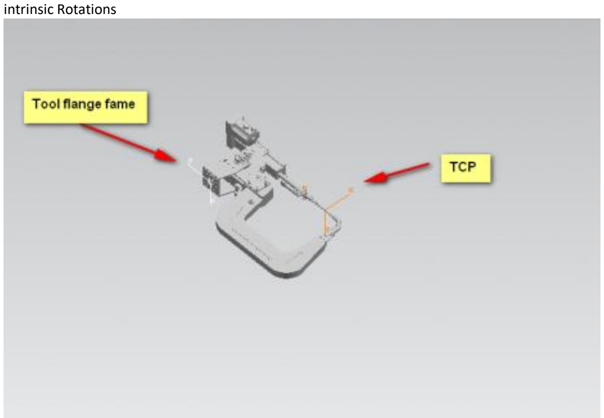
The J1-J6 totally have four solutions as below :

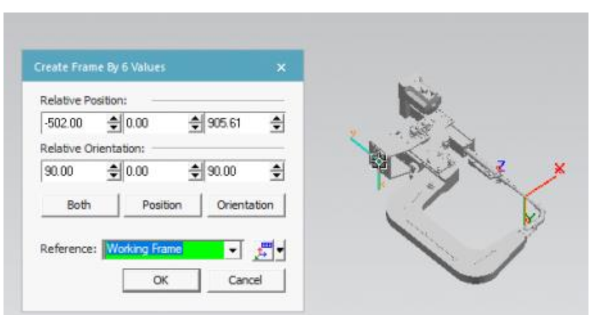


1. Got the translation matrix between TCP frame and Tool flange frame, it is defined during TOOL manufacturing phase;
   As the example showing below :
   The Tool flange frame can be got after the TCP through below steps:

- Move -502mm according to X direction;
- Move 905.61 according to Z direction;   Frame translation should be happened before rotation;

- Rotate 90 deg according to Z direction;
- Rotate 0 deg according to Y direction;
- Rotate 90 deg according to X direction; all is according to original TCP frame, which is named as intrinsic Rotations



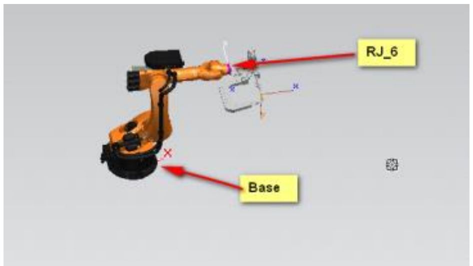


Coding as below :

```
rpy_raw << 90, 0, 90;
rpy_raw = rpy_raw * M_PI / 180;
Eigen::Isometry3d R_Tool = Eigen::Isometry3d::Identity();
R_Tool = (Eigen::AngleAxisd(rpy_raw[2], Eigen::Vector3d::UnitZ())*Eigen::AngleAxisd(rpy_raw[1], Eigen::Vector3d::UnitY())*Eigen::AngleAxisd(rpy_raw[0], Eigen::Vector3d::UnitX()));
R_Tool.pretranslate(Eigen::Vector3d(-502, 0, 905.61));//在这里平移需要放到轴的坐标上去,相对于旋转之前的坐标系方向
```

R_Tool will be the rotation matrix between TCP/ Flange frame, and RJ_6 frame data can be got through below coding;

```
// R_JT6坐标系相对于基坐标系的位置关系如下：
Eigen::MatrixXd RJ_6;
RJ_6 = Eigen::MatrixXd(4, 4);
RJ_6 = RTCP * R_Tool.matrix();
```



2. How to calculate the J1 angles to got the final RJ_6 end effector location ?

   According to Robot Forward Kinematics, we can got each Robot Joint translation matrix as below :

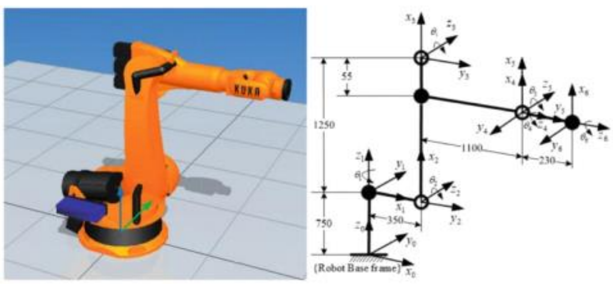


Fig. 1. DH model of the KUKA KR210-2 robot.

Coding as below :

```
Eigen::MatrixXd A01;
A01 = Eigen::MatrixXd(4, 4);

A01 << 0, c1, s1, 500 * c1,
       0, -s1, c1, -500 * s1,
       1,  0,  0,  1045,
       0, 0, 0, 1;

Eigen::MatrixXd A12;
A12 = Eigen::MatrixXd(4, 4);

A12 << c2, s2, 0, 1300 * c2,
    -s2, c2, 0, -1300 * s2,
    0, 0, 1, 0,
    0, 0, 0, 1;

Eigen::MatrixXd A23;
A23 = Eigen::MatrixXd(4, 4);

A23 << c3, 0, s3, -55 * c3 + 1275 * s3,
    -s3, 0, c3, 55 * s3 + 1275 * c3,
    0, -1, 0, 0,
    0, 0, 0, 1;

Eigen::MatrixXd A34;
A34 = Eigen::MatrixXd(4, 4);
A34 << c4, 0, -s4, 0,
    -s4, 0, -c4, 0,
    0, 1, 0, 0,
    0, 0, 0, 1;

Eigen::MatrixXd A45;
A45 = Eigen::MatrixXd(4, 4);
A45 << -c5, 0, s5, 290 * s5,
    s5, 0, c5, 290 * c5,
    0, 1, 0, 0,
    0, 0, 0, 1;

Eigen::MatrixXd A56;
A56 = Eigen::MatrixXd(4, 4);
A56 << c6, s6, 0, 0,
    -s6, c6, 0, 0,
    0, 0, 1, 0,
    0, 0, 0, 1;
```

```
A = A01 * A12*A23*A34*A45*A56;
```

A is the final translation matrix frame robot base frame to RJ_6 end effector frame;

Then can be got below calculation equitation :

$A01^{-1} * A * A56^{-1} = A12*A23*A34*A45;$

$A01^{-1} * A * A56^{-1} =$

```
// a31*c6+a32*s6,                    -s6*a31+a32*c6,               a33,          z0-1045,
```

```
// a31*c6+a32*s6,                              -s6*a31+a32*c6,                          a33,            z0-1045,
// c6*c1*a11-c6*s1*a21+s6*a12*c1-s1*s6*a22,  -s6*(c1*a11-a21*s1)+c6*(a12*c1-s1*a22), c1*a13-s1*a23,  c1*x0-s1*y0-500,
// (s1*a11+c1*a21)*c6+s6*(a12*s1+a22*c1),   -s6*(s1*a11+c1*a21)+c6*(s1*a12+c1*a22), s1*a13+c1*a23,  s1*x0+c1*y0,
//                                        0,                                       0,             0,             1;
```

A12*A23*A34*A45=

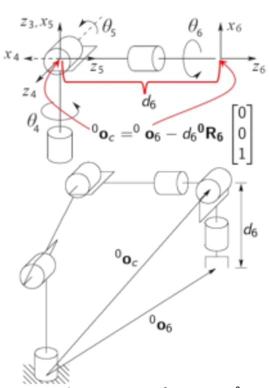
```
//A15 << -c23 * c4*c5 + s23 * s5,   -c23 * s4,    c23*c4*s5 + s23 * c5,    290 * s5*c23*c4 + 290 * c5*s23 - 55 * c23 + 1275 * s23 + 1300 * c2,
//      s23*c4*c5 + c23 * s5,       s23*s4,      -s23 * c4*s5 + c23 * c5, -290 * s5*s23*c4 +290 * c5*c23 + 55 * s23 + 1275 * c23 - 1300 * s2,
//       -s4 * c5,                   c4,         s4*s5,                    290 * s5*s4,
//       0,                          0,          0,                        1;
```

Note :
c1=cos(J1); S1=sin (J1)
c2= cos(J2); S2=sin(J2);
C23=COS(J2+J3); S23=sin(J2+J3)
a11,a12, ……..a33, x0, y0, z0 is the matrix for RJ_6 which is calculated previous;

```
// R_JT6坐标系相对于基坐标系的位置关系如下：
Eigen::MatrixXd RJ_6;
RJ_6 = Eigen::MatrixXd(4, 4);
RJ_6 = RTCP * R_Tool.matrix();

//将RJ_6的值分别赋值给逆运算矩阵(J6法兰盘TCP)
// a11, a12, a13, x0,
// a21, a22, a23, y0,
// a31, a32, a33, z0,
//   0,   0,   0,  1;


double a11 = RJ_6(0, 0);
double a12 = RJ_6(0, 1);
double a13 = RJ_6(0, 2);
double x0 = RJ_6(0, 3);

double a21 = RJ_6(1, 0);
double a22 = RJ_6(1, 1);
double a23 = RJ_6(1, 2);
double y0 = RJ_6(1, 3);

double a31 = RJ_6(2, 0);
double a32 = RJ_6(2, 1);
double a33 = RJ_6(2, 2);
double z0 = RJ_6(2, 3);
```

Then will be got below data equitation :
A15(2,3)/A15(2,2)=d5;
(s1*x0+c1*y0)/(s1*a13+c1*a23)=d5;
tanJ1=(d5*a23-y0)/(x0-d5*a13);
J1=atan2(((*d5) * a23 - y0), (x0 - (*d5) * a13));
from now on we have got the 1st Joint deg for robot;

3. How to calculate J2/ J3 joint deg to got the final RJ_6 end effector location ?



According to upper information , ^0O_c is RJ_5 joint vector means can got the RJ_5 joint position information but can not got the rotation matrix;
Coding below can get the ^0O_c RJ_5 joint location:

```
Eigen::Vector3d TCP;

TCP << x0, y0, z0;

Eigen::Vector3d w(0,0,1);
Eigen::MatrixXd TCP_Oritation;
TCP_Oritation = Eigen::MatrixXd(3, 3);

TCP_Oritation << a11, a12, a13,
    a21, a22, a23,
    a31, a32, a33;

Eigen::Vector3d J5_Location;

//J5_Location = TCP - 290 * TCP_Oritation*w;
J5_Location = TCP - (*d5) * TCP_Oritation*w;

std::cout << J5_Location << std::endl;
```

According to Robot Forward Kinematics, we can got RJ_5 Location as below :

```
//-c1 * s23*c4 + s1 * s4, c1*c23, c1*s23*s4 + s1 * c4, 55 * c1*s23 + 1275 * c1*c23 + 500 * c1 - 1300 * c1*s2,
//s1*s23*c4 + c1 * s4, -s1 * c23, c1*c4 - s1 * s23*s4, -55 * s1*s23 - 1275 * s1*c23 - 500 * s1 + 1300 * s1*s2,
//c23*c4, s23, -c23 * s4, -55 * c23 + 1275 * s23 + 1045 + 1300 * c2,
//0, 0, 0, 1;
//J5_Location(0,0)=55 * c1*s23 + 1275 * c1*c23 + 500 * c1 - 1300 * c1*s2;
//J5_Location(1,0)=-55 * s1*s23 - 1275 * s1*c23 - 500 * s1 + 1300 * s1*s2;
//J5_Location(2,0)= -55 * c23 + 1275 * s23 + 1045 + 1300 * c2;
//J1 = (J1)*M_PI / 180;
// 55*s23+1275*c23-1300*s2=J5_Location(0,0)/(cos(J1))-500;
// 55*c23-1275*s23-1300*c2=-J5_Location(2,0)+1045;
```

RJ_5 Matrix

can be got the J2/J3 data

4. How to Get J4/J5/J6 data

Upper calculation can be got the data for J1/J2/J3;

Then A03= A01*A12*A13;

A03*A36=A06=RJ_6;

A36 = (A03.inverse())*RJ_6;

```
//A36=A34*A45*A56;
//A36<< -c4*c5*c6+s4*s6,   -c4*c5*s6-s4*c6,  c4*s5,  (*d5)*c4*s5,
//       s4*c5*c6+c4*s6,   s4*c5*s6-c4*c6,   -s4*s5, -(*d5)*s4*s5,
//       s5*c6         ,   s5*s6,            c5,     (*d5)*c5,
//           0,                0,             0,      1;
```

Then we can got the J4/J5/J6 data value

1st solution for J4/J5/J6
J4 = atan2(-A36(1, 2), A36(0, 2));
J5= atan2(sqrt(A36(0, 2)*A36(0, 2)+ A36(1, 2)*A36(1, 2)), A36(2, 2));
J6= atan2(A36(2, 1), A36(2, 0));

2nd solution for J4/J5/J6

J4 = atan2(-A36(1, 2), -A36(0, 2));
J5 = atan2(-sqrt(A36(0, 2)*A36(0, 2) + A36(1, 2)*A36(1, 2)), A36(2, 2));
J6 = atan2(-A36(2, 1), -A36(2, 0));

5. Above calculation can be got all the 4 solutions for KUKA Robot