**Lab 2 Robot Kinematics and Adept V+ Commands**

**9-10-14**

**Objectives**

* Learn the Adept V+ commands for kinematics including frame rotation, translation, and forward and backward calculations of joint motion.
* Manually calculate joint by joint transformation matrices from the frames set up in Lab 1.
* Compare the results predicted by that the transformation series with the actual extracted from the robot joint movement data. Trace and explain any discrepancies in the comparison.

**Part A**

1. Complete the Devavit-Hartenberg tables based on the robot frame assignments from Lab 1. There are three tables for three robot configurations - Cartesian, SCARA, and articulating.. Make any corrections and adjustments on the frames before finalizing the D-H tables.
2. Build a 4x4 transformation matrix from joint i-1 to joint i for all i in the robot using θi or di as the variables. Transformations occur only in Z direction as the way the frames are set up. For six axes robot, split the propagation into two parts – Joints 1-3 and Joints 4-6.

**Part B**

Using one of the manipulator controllers, use the commands listed in the attached V+ Command Summary for Kinematics. Some commands require the robot being turned on and calibrated.

**Part C** (*See the notes 1-3*)

Take one of the three robot types, combine two or three joint motions into one via joint value additions or transformation. See if successive Rx(α): Dx(a):Ry(β):Dy(b):Rz(γ):Dz(c) will yield the result predicted by . Use a set of actual values (such as θ1=30º, θ2=45º,… and d3=100mm,…) for verification. Do the following:

1) Perform MatLab or Excel calculations of the equations.

2) Carry out the transformations on the selected Adept robot.

3) Compare the two results. For any discrepancies found, provide a plausible explanation for the differences, such as a mismatched frame assignment, a flipped pitch, a measurement error, etc.

*Note 1.*  The wrist joint frames of all of the six axes robots in Lab 192 and 194 are coincidental and follows the Z-Y-Z (yaw, pitch, roll) Euler angle rotations in the textbook (p.45). In the SCARA and Cartesian robots, the yaw(on XB) and the pitch(on YB) are fixed 0º and 180º, respectively.

*Note 2*. A location transformation is stored in a 48-byte long data stream in a “.lc” file. The routine in the attachment will yield a screen display of input location transformation P.

*Note 3. Adept Definition of Yaw, Pitch, and Roll:*

Yaw = a rotation of the local reference frame about its Z axis.

Pitch = a rotation of the local reference about its Y axis, with the yaw applied.

Roll = a rotation of the local reference frame about the Z axis with the yaw and pitch applied

*Flipped Pitch (180º about Y) 🡪 Reversed orientation on both X and Z axes.*

**Report**

Due 9/17/14. The format is similar to the Lab 1 report. Pair the joint frames and the D-H tables.

Include the results from Part A and C and a short description of the team’s experience with Part B.

**Adept V+ Command Summary for Kinematics Analysis**

*(Search on line for “Adept Quick” for details)*

DECOMPOSE V[i] = P (#P) ; Decomposes location P (or #P) and store in array V.

DRIVE joint, angle, speed ; Drives a joint by the angle at the percent of the motor speed.

DX(d), DY(d), DZ(d) ; Gives displacement value of value d in X, Y, Z.

FLTB($S,k) ; Returns the value of a 48-byte string containing a floating point number

FRAME(p1, p2, p3, p4) ; Yields a transformation value defined by four positions.

HERE P (or #P) ; Defines the current gripper position as Cartesian P or angular #P

INRANGE(P) ; Determines if location P is reachable by robot arm.

LISTL ; Lists the currently defined locations and their values.

MOVE (or MOVES) P ; Joint interpolated (or straight line) move to a location P.

#PPOINT(j1, j2, …) ; Precision point (location) set by the specified joint angles or offsets.

RX(v), RY(v), RZ(v) ; Rotates X, Y, Z axis by the angle v. XYZ are world coordinates.

SET P (or #P) = expression ; Defines a location vector P (or #P) with TRANS or #PPOINT.

SHIFT(P BY x, y, z) ; Translates location P by amount specified in XYZ direction

SOLVE.ANGLES P ; Finds the joint rotation angles for given transformation P.

SOLVE.FLAGS V[ ] ; Finds config.bits (left-right, above-below, pitch flip) in joint array V.

SOLVE.TRANS #P ; Finds the transformation for given joint (angular) values of #P.

TRANSB ; Returns a transformation value represented by a 48-byte string.

TRANS(x, y, z, n, p, r) ; Transformation (location) of a six-axis Cartesian vector

*Relative Positioning – Matrix addition:*

SET S = P:Q:R ; Additions of Cartesian coordinate values (x, y, z, n, p, r)

*Translation and Rotation:*

SET Q = P:DX(a):DY(b):DZ(c) ; Shift P by a along X, then by b along Y, then by c along Z

SET #Q = #P:RX(α):RY(β):RZ(γ) ; Rotate #P by α on X, then by β on Y’, then by γ on Z’’

SET #Q = #P:RX(α): DX(a):RY(β):DY(b) ; Combined rotation and translation Q 🡪.

*6-axis robot wrist transformation (Euler Z-Y-Z angle transformation):* RZ(α):RY(β):RZ(γ)

**Useful V+ Programs and Commands for Kinematics Calculation**

**.PROGRAM SHOWTRAN( )**

; Displays a 3x4 transformation matrix from location P stored in a 48 byte data stream.

FOR i = 0 TO 2  
FOR j = 0 TO 3   
index = 4\*i+12\*j +1  
TYPE /F10.5, FLTB($TRANSB(P),index), /S ; 10 spaces, 5 digits each. No line feed.  
END

TYPE  
END

.END

**.PROGRAM STRIPJNT(N)**

; Strips the Nth parameter value from location array P. A precision point #P may be used for P.

;

DECOMPOSE V[ ]=P ; (or #P) Split P into its component parameter values and store in V.

FOR I=0 TO 5

IF I==N THEN

V [I]=0

END

SET P=TRANS(V[0], V[1], V[2], V[3], V[4], V[5]) ; Strip P of its nth parameter value.

; SET #P=#PPOINT(V[0], V[1], V[2], V[3], V[4], V[5]) ; For joint values #P

.END

**.PROGRAM SHOWCOMP(N)**

; Displays 1x6 or 1x4 array of P or #P.

;

DECOMPOSE V[ ]=P (#P) ; Split P (or #P) into its component values and store in V.

FOR I=0 TO 5

TYPE /F10.5, P, /S ; 10 spaces, 5 digits each. No line feed.

END

**Forward Kinematics Calculation**

Given the angular or linear joint values stored in array V[i], i=0,…,5, find the transformation P from V with a report on any errors.

**.PROGRAM SOLVTRAN( )**

; Gives a Cartesian transformation of a set of joint values with any displacements.

;

PROMPT “Input up to six joint values for #P: “, v[0],v[1],v[2],v[3],v[4],v[5]

PROMPT “Input displacement values for #P: “, w[0],w[1],w[2],w[3],w[4].w[5]

FOR I=0 TO 5

V[I]=V[I]+W[I]

END

SOLVE.TRANS P, ERR=V[ ]

**Backward Kinematics Calculation**

Given the Cartesian coordinates of the end effector position in array V[i], i=0,…,5, find the joint angle array ANG[i], i=0,…,5 with configuration flags (left-right, above-below, pitch flip) set in V[ ] and any error report.

*Note: The inverse calculations are complicated and there often are multiple solutions. The flags are used to specify the selections made along the backward path of joint calculations.*

.PROGRAM SOLVANGL( )

;

FOR i = 0 TO 5

v[i] = 0

END

DECOMPOSE v[ ] = P

PROMPT "Input the displacement values for p: ", w[0], w[1], w[2], w[3], w[4], w[5]

FOR i = 0 TO 5

v[i] = v[i]+w[i]

END

;

SOLVE.ANGLES ANG[ ], flags, error = P, v[ ], SOLVE.FLAGS(v[ ])

TYPE

FOR i = 0 TO 5

TYPE /F9.4, ANG[i], /S

END

TYPE

.END