Tech Note #28

Introduction

While MotionBASIC® does not have a built in circular interpolation function it is very easy to program an ORION™ Controller to execute circular motions. This Tech Note describes how to do it.

Definition of Arc

To move in a circle, or portion of a circle you must know four of the six parameters that can be used to define an arc. These parameters are:

1. The coordinates of the starting point \((X_1, Y_1)\)
2. The coordinates of the center of rotation \((I, J)\)
3. The angle through which you want to move \((\Theta)\)
4. The coordinates of the end point of the arc \((X_2, Y_2)\)
5. The radius of the arc \((R)\)
6. The length of the arc \((C)\)
7. Circumferencial speed \((S)\).

As indicated you need only specify four of these six. The most commonly used three are the starting point, center of rotation, the angle and the circumferencial speed. If you want to specify the arc using parameters other than the four suggested, you will need to figure how to calculate these four from the parameters you have.

If you are interested in the mathematics behind circular interpolation, read the next section. If you are only interested in an example of how to do it, consider jumping to the section titled "Implementation".

Interpolation Solution

Circular interpolation involves defining the arc you want and then calculating the positions the axes need to pass through in order create the arc. Typically, the program needs to calculate these positions and issue the motion commands at a regular interval of about 20-50 ms in order to generate a smooth motion.

Let's first define the X and Y distances from the center of rotation to the starting position as \(X_c\) and \(Y_c\) respectively.

\[
X_c = X_1 - I \\
Y_c = Y_1 - J
\]

The radius of the arc \(R\), is given by,

\[
R = \sqrt{X_c^2 + Y_c^2}
\]
To calculate the starting angle we need to know in which quadrant of the circle the line \( I,J \) to \( X_1,Y_1 \) lies.

**Case 1,** \( X_c > 0 \)

\[ \Theta_1 = \arctan \left( \frac{Y_c}{X_c} \right) \]

**Case 2,** \( X_c < 0 \)

\[ \Theta_1 = \pi + \arctan \left( \frac{Y_c}{X_c} \right) \]

**Case 3,** \( X_c = 0 \) and \( Y_c > 0 \),

\[ \Theta_1 = \frac{\pi}{2} \]

**Case 4,** \( X_c = 0 \) and \( Y_c < 0 \),

\[ \Theta_1 = -\frac{\pi}{2} \]

**Case 5,** \( X_c = 0 \) and \( Y_c = 0 \)

This case is not possible since it represents a radius of zero.

We are going to move along the arc at a speed \( S \), which is in inches per second, so the angular velocity is,

\[ \omega = \frac{S}{R} \text{ radians/second.} \]

Let's assume we calculate a new set of positions every 20 ms. This gives us an angular move for each update of,

\[ \Delta \Theta = \frac{0.02 \cdot S}{R} \text{ radians} \]

and a total number of updates \( N \) for the complete move, of

\[ N = \frac{\Theta}{\Delta \Theta} \]

The position we need to move to in each update are are,

\[ X_n = I + R \cdot \cos(\Theta_n) \]
\[ Y_n = J + R \cdot \sin(\Theta_n) \]

So we will divide the move up into \( N \) segments, increment the angle by \( \Delta \Theta \) for each segment and calculate a new value for the X and Y positions.
Implementation

This technique breaks the move up into small, 20 ms pieces, and commands a series of moves to each successive position. MotionBASIC® would normally turn each piece of the move into a trapezoidal move with an acceleration, cruise and deceleration. In our case we do not want the axes to decelerate to a stop between each piece of the move so we will command each piece with a zero acceleration and deceleration time. Since the speed change from each piece of the move to the next is quite small, instantaneous acceleration is not a problem during the move itself. It is however a problem at the start and end of the arc. Depending on the starting and ending angle, the axes may have to instantaneously accelerate to or from the maximum speed.

To avoid having these accelerations become a problem, you need to turn velocity feed forward off by setting KVF@=0 for both axes. You will also need to set ACL.MAX@=0 and DCL.MAX@=0 for both axes. Finally you will need to set the KP@ value to provide a reasonable, identical position gain for both axes. A reasonable value for position gain is 1 inch per minute/mil. You may use a higher gain for more responsive systems or a lower gain for heavier, more sluggish systems.

\[
G = \frac{395 \cdot KP@}{LOOP.RATE@ \cdot VLTC@} \text{ inches per min/mil}
\]

\[
KP@ = \frac{G \cdot LOOP.RATE@ \cdot VLTC@}{395} \% 
\]

Depending on the position gain you use and the characteristics of your mechanism, you may need to adjust or disable PERR.MAX@ position error checking to avoid nuisance position error faults.

It is very important for accuracy to have both axes with identical position gains.

Program Listing

POWERUP:
  MP.CONFIG
  X~={1}
  Y~={2}
  BOTH~={X~+Y~}
  ACL.MAX@(BOTH~)=0
  DCL.MAX@(BOTH~)=0
  POS.GAIN=1
  KVF@(BOTH~)=0
  KP@(X~)=POS.GAIN*LOOP.RATE@*VLTC@(X~)/395
  KP@(Y~)=POS.GAIN*LOOP.RATE@*VLTC@(Y~)/395
  PROG.INIT
  DEFINE.ARC
  SETUP.ARC
  ENABLE
  MOVE.TO.START
  MOVE.ARC
  WAIT UNTIL DSP.DONE@(BOTH~) AND IN.POS@(BOTH~)
  MODE@(BOTH~)=0

CSE Tech Note # 28 06/25/96 (1.0b) page 3 of 5
DEFINE.ARC:
  I&=0              'X coordinate of center (inches*1000)
  J&=0              'Y coordinate of center (inches*1000)
  X1&=1000          'X coordinate of starting position (inches*1000)
  Y1&=0             'Y coordinate of starting position (inches*1000)
  ARC.LEN.DEG!=360  'Arc length (degrees)
  S!=1              'speed (inches per sec)
RETURN

SETUP.ARC:
  XC&=X1&-I&
  YC&=Y1&-J&
  R!=SQR(XC&*XC&+YC&*YC&)/1000
  R1000!=R!*1000
  IF XC&>0 THEN
    START.ANG!=ATN(YC&/XC&)
  ELSEIF XC&<0 THEN
    START.ANG!=PI!+ATN(YC&/XC&)
  ELIF YC&>0 THEN
    START.ANG&=HALF.PI!
  ELSE
    START.ANG&=-HALF.PI!
  ENDIF
  ARC.LEN!=ARC.LEN.DEG!*2*PI!/360
  END.ANG!=START.ANG!+ARC.LEN!
  DELTA.THETA!=UPD.TIME.S!*S!/R!
  IF ARC.LEN.DEG!<0 THEN DELTA.THETA!=-DELTA.THETA!
  N&=ABS(INT(ARC.LEN!/DELTA.THETA!))
RETURN

ENABLE:
  OTL.FWD@=0:OTL.REV@=0:AFault@=0:FAULT@=0:WAIT 300:MODE@=5
RETURN

MOVE.TO.START:
  X.DIST&=X1&-POS.CMD@(X~)             '(inches*1000)
  Y.DIST&=Y1&-POS.CMD@(Y~)             '(inches*1000)
  MOVE.DIST&=SQR(X.DIST&^2+Y.DIST&^2)  '(inches*1000)
  MOVE.TIME&=MOVE.DIST&/S!             '(ms)
  MOVE X~ TO X1& IN MOVE.TIME&,0,0
  MOVE Y~ TO Y1& IN MOVE.TIME&,0,0
  WAIT UNTIL (DSP.DONE@(BOTH~) AND IN.POS@(BOTH~))
RETURN

MOVE.ARC:
  ANG!=START.ANG!
  COUNT&=1
  WHILE INKEY$="" AND COUNT&<=N&
    IF COUNT&=N& THEN
      ANG!=END.ANG!
    ELSE
      ANG!=ANG!+DELTA.THETA!
    ENDIF
AXIS.VAR@ (X-) = I$ + R1000! * COS(ANG!)
AXIS.VAR@ (Y-) = J$ + R1000! * SIN(ANG!)
MOVE BOTH- TO AXIS.VAR@ IN UPD.TIME.MS, 0, 0
COUNT$ = COUNT$ + 1
WEND
RETURN
'

PROG.INIT:
  PI! = 3.14159
  HALF.PI! = PI! / 2
  UPD.TIME.MS = 20  ' (ms)
  UPD.TIME.S! = UPD.TIME.MS / 1000 ' (seconds)
RETURN