

Correcting For Lost Distance When Changing Direction

One feature of the PMAC controller is its capability to blend moves in a fashion that there are no discontinuities in velocity. It does this using acceleration time as programmed by the user. This provides for excellent vector path control in motion. Once it is understood how this functions, you can determine what the motion results will be.

To do this blending, the acceleration (or deceleration) has to start at a distance determined by the acceleration time.

In the diagram below the following is shown:

V1 = the current velocity

V2 = the new velocity

TA = The programmed acceleration time.

For example, with a single axis, the velocity change will start at a point where the velocity change will happen at an acceleration rate which meets the acceleration time programmed. Therefore, the acceleration rate is calculated as follows:

If going to 0 velocity: $A1 = (V1 - 0) / TA$

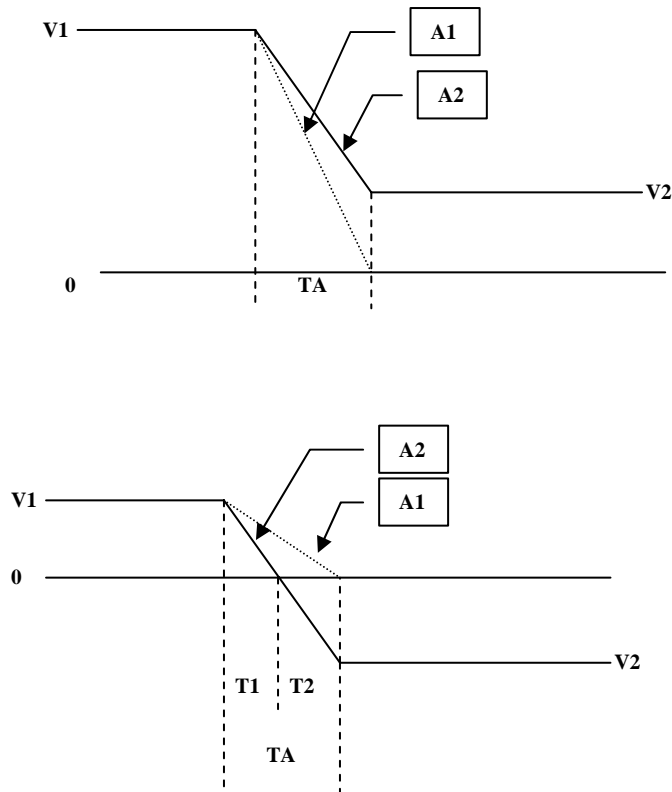
If going to a different velocity: $A2 = (V1 - V2) / TA$

When reversing direction, one thing that happens is you will not reach the programmed distance for the axis. Deceleration will happen in T1 time, at which point the velocity reverses. Deceleration continues for T2 time, at which point the new velocity is reached. The distances can be calculated as follows:

S1 = Distanced traveled when decelerating from V1 to 0 with an acceleration of A1

S2 = Distanced traveled when decelerating from V1 to 0 with an acceleration of A2

S3 = Is the difference in distance traveled (S1 - S2).



Our objective here is to insure we reach the end point programmed when the velocity changes. This is critical, for example, in a winding application where fiber or wire is being wound back and forth on a mandrel and we need to insure the edge is always at the same point on the mandrel, regardless of the speeds and accelerations.

To travel the same distance, we need to add distance S3 to the original move distance. The result will then be what we desire. The following equation is how this distance, S3, is calculated.

$$S3 = \frac{V1 * V2 * TA}{2 * (V1 + V2)}$$

Notice if V1 = V2 the equation simplifies to:

$$S3 = \frac{V1 * TA}{4}$$

Note: When doing high accelerations and/or short move times, you may have to reduce I8 = 0 and set Isx13 low. You will be very close to hitting the desired turn around point. S-Curve Acceleration also will contribute to missing the point, so reducing this to a small number will also help.

To test this, 2 simple programs have been created as follows:

```

***** Test Prog 1.pmc *****
;
#define S1      P100
#define S2      P101
#define vel_1   P102
#define vel_2   P103

undefine all
&1
#1->1000X
#2->1000Y

p1=10

close
del gat
open prog 1 clear
X(-p1)
While ( 1 < 2 )
  ta1000
  tm5000
  vel_1 = p1/10000
  x(p1)
  ta1000
  tm5000
  x(-p1)
  vel_2 = p1/10000
ENDW

close
;
open prog 2 clear
f10
X(-p1)
While ( 1 < 2 )
  ta1000
  tm5000
  vel_1 = p1/10000
  ; S3 = ( 2 * P1 ) * TA / (( 2 * TM ) * 2 )
  ;   = P1 * TA / ( 2 * TM )
  x(p1 + ( P1 * 1000 ) / ( 5000 * 2 ))
  ta1000
  tm5000
  x(-p1 - ( P1 * 1000 ) / ( 5000 * 2 ))
  vel_2 = p1/10000
ENDW

close

```

The results are shown in the following plots from the controller:

