Move Modes

Linear Move Mode
Circular Move Mode
Rapid Move Mode
Spline Move Mode
PVT Move Mode

Linear Move Mode

- **Intended for point-to-point moves**
- **Can be blended with linear, circle, and PVT moves**
- **Commands:**

Linear	// Linear Move Mode
TM {*constant*}	// Move Time (msec); **Distance = Velocity * TM**
TA {*constant*}	// Acceleration Time (msec); Default value = **Coord[*x*].Ta**
TD {*constant*}	// (Final) deceleration time (msec); Default = **Coord[*x*].Td**
TS {*constant*}	// S-Curve Time (msec); Default value = **Coord[*x*].Ts**
F {*constant*}	// Feedrate (user unit/user time); Tool tip velocity
	// Move Time = Distance / **F**
	// **Coord[*x*].Tm** gets set to –**F** when **F** is set
Abs / Inc	// Absolute / Incremental endpoint specification

- **TA, TS, and TD can all be set = 0, and then motor limits (next slide) govern accelerations**
- **Moves can be segmented (Coord[*x*].SegMoveTime > 0) or not (=0)**
  - Must be segmented to use inverse kinematics
  - Must be segmented to blend with circle moves
  - Must be segmented to use Special Lookahead

Example:

```
linear                  // Linear move mode
TM 1000 TA 500 TS 0     // Move time 1000 msec, Acc. Time 500 msec, No S-Curve Time
abs                     // Absolute endpoint mode
X 20 Y 10               // Go to X=20, Y=10
```

Power PMAC Script

Linear Move Trajectory

- **Velocity Profile**

  For each move, total move time is **TM+TA**, or **TM+(TA+TD)/2** if using **TD**

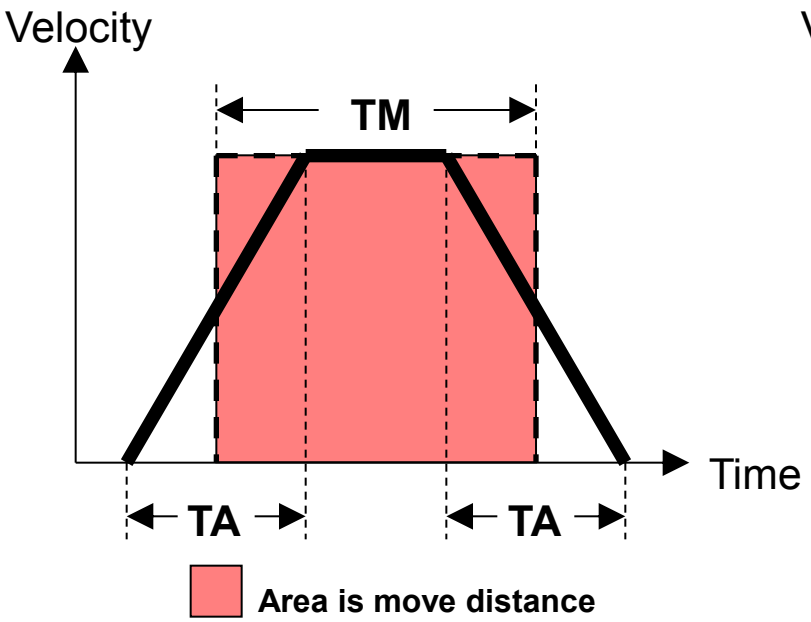
  Velocity profile is under the constraints of :

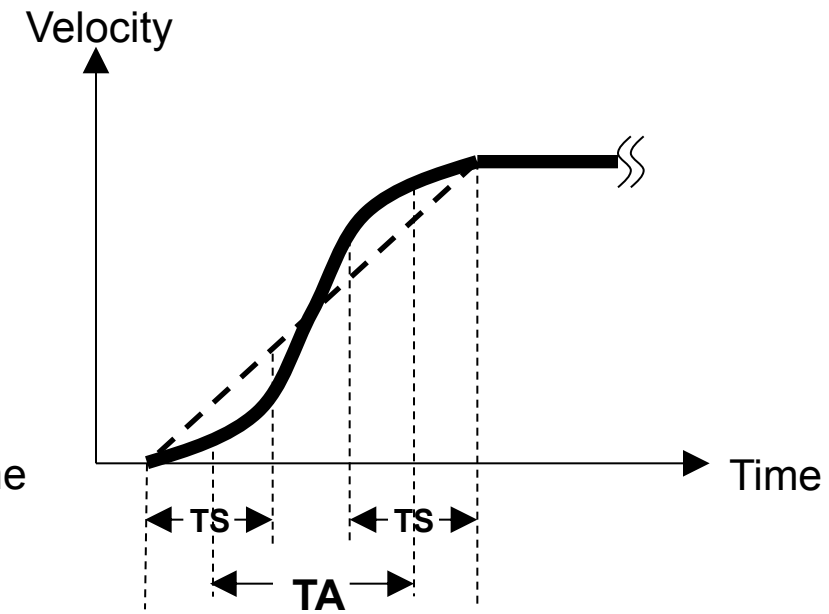
  Maximum program velocity: **Motor[*x*].MaxSpeed**

  Maximum program acceleration: **Motor[*x*].InvAMax**

  Maximum program deceleration: **Motor[*x*].InvDMax**

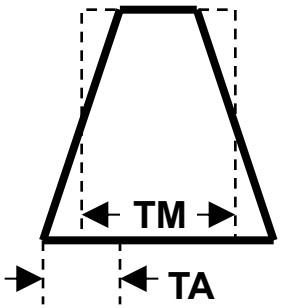
  Maximum program jerk: **Motor[*x*].InvJMax**





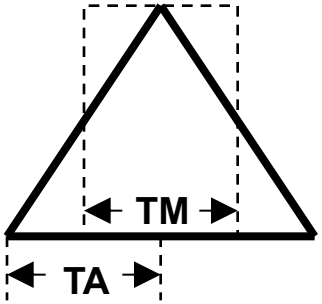
Move Rules

TA and TM (without TS)



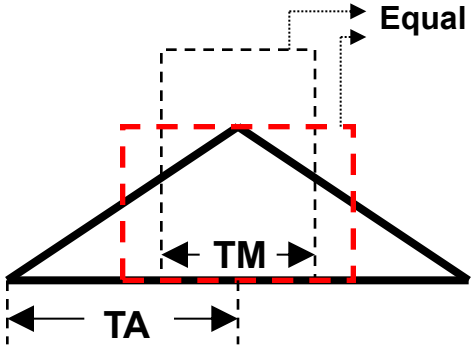
TM > TA

Total Time = TM + TA



TM = TA

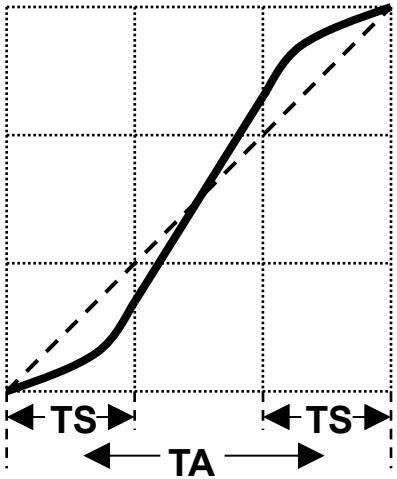
Total Time = TM + TA = 2 * TA



TM < TA

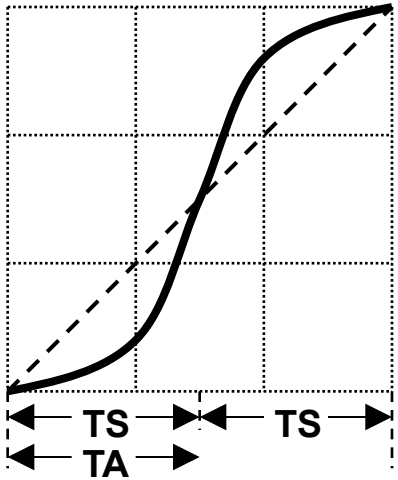
Total Time = 2 * TA

TA and TS



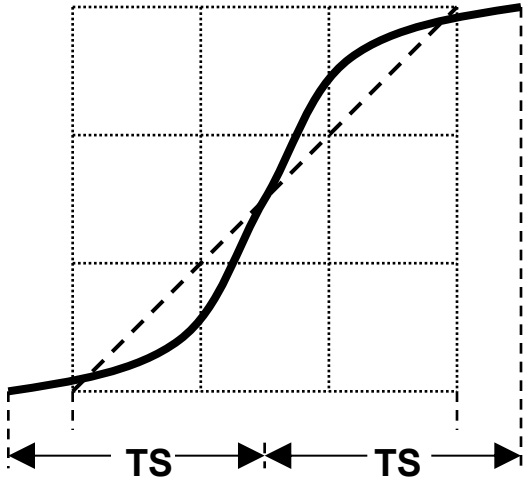
TA > TS

Total Acc. Time = TA + TS



TA <= TS

Total Acc. Time = 2 * TS

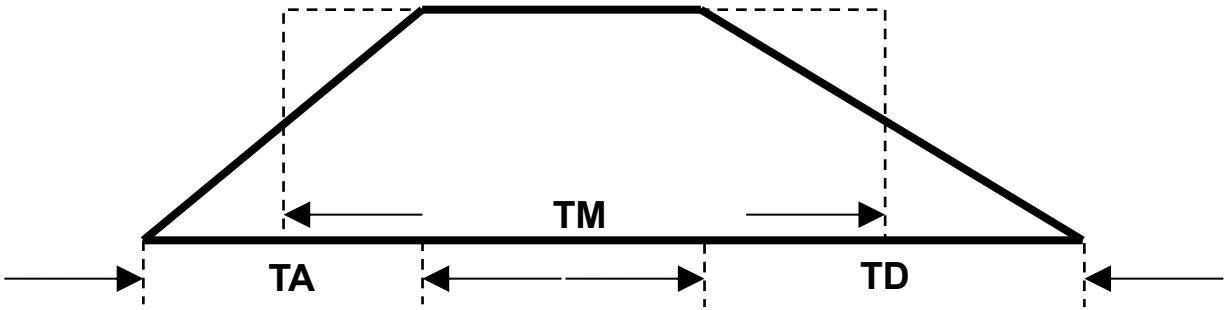


TA = 0

Total Acc. Time = 2 * TS

Move Rules

TA, TM, and TD

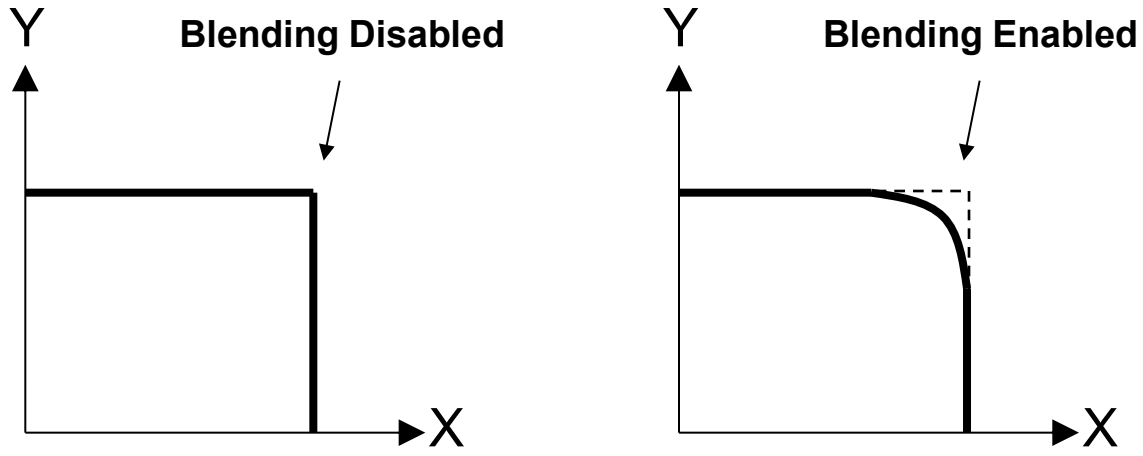


Total Time = TM + (TA + TD)/2

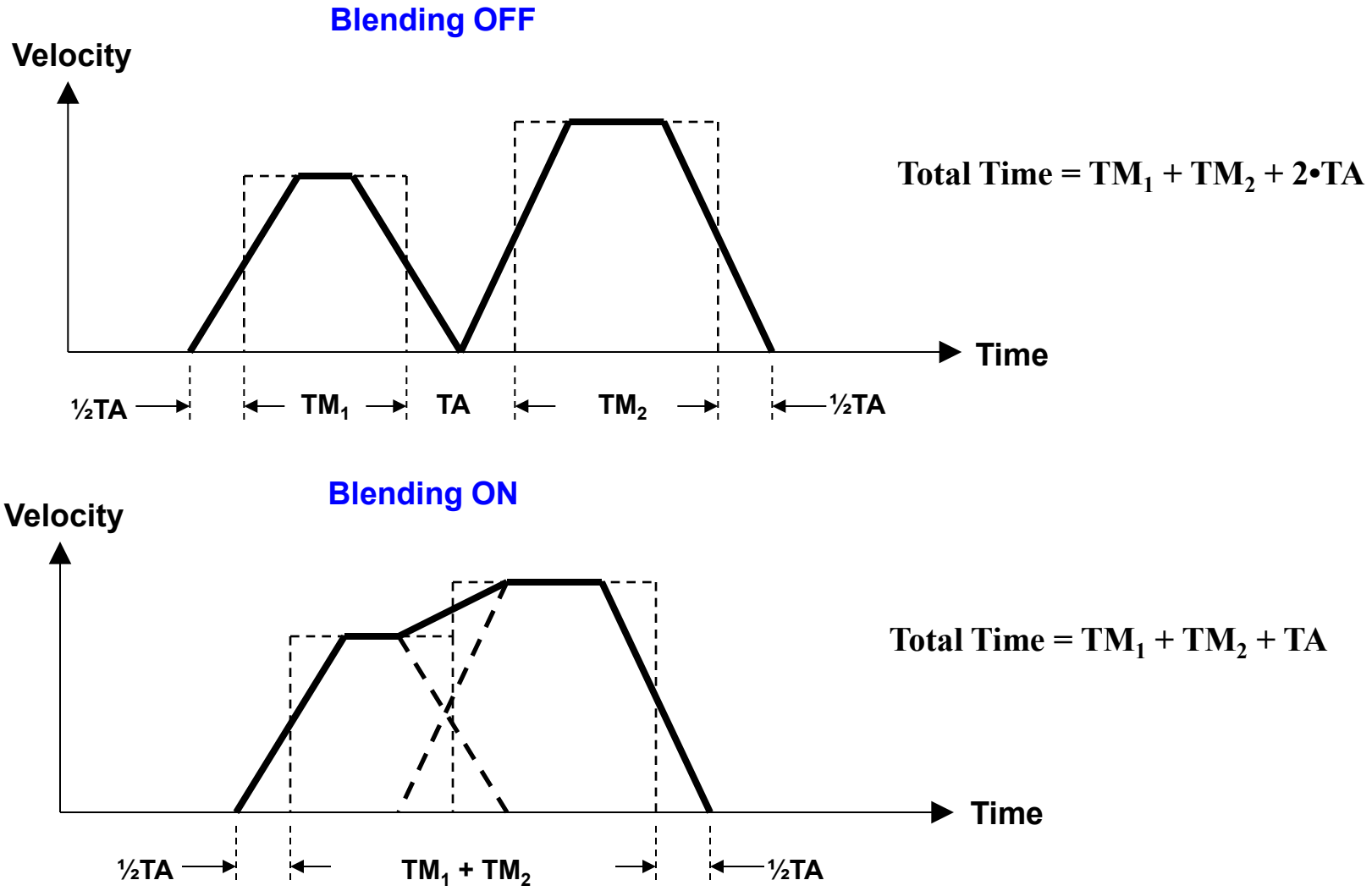
Linear Move Blending

- **PMAC can perform move blending between sequential linear, circular, and PVT moves**

- **Blending is not used if:**
  - Moves are separated by a **dwell** command
  - (**Coord[*x*].GoBack + 1)** jumps in the program from either **goto** or the end of a **while** loop
  - Move Blending disabled (**Coord[*x*].NoBlend** = 1)



Linear Move Blending



Constraints on Linear/Circle Moves

- If *accel time* is 0.0, no arbitrary minimum move time, tries to accelerate that quickly (limited by **Motor[*x*].InvAmax** and **InvJmax**)
- If *move time* < **Sys.ServoPeriod** (servo update period), the move will be calculated, but then skipped over by trajectory servo interpolation
- In segmentation mode, if *move time* < **Coord[*x*].SegMoveTime**, move will be calculated, but then skipped over by trajectory segment interpolation
- With very short moves, user must be careful not to overwhelm Power PMAC real-time calculation capabilities (run-time error would result and program would halt)

Circular Move Mode

- **Two Cartesian axis sets for circular interpolation per C.S.**
  - X/Y/Z: Main Cartesian axis set
  - XX/YY/ZZ: Secondary Cartesian axis set
- **Two sets of vector components (for normal and center vectors)**
  - I/J/K: for X/Y/Z axis set
  - II/JJ/KK: for XX/YY/ZZ axis set
- **Center vector must be specified from move start point**
- **"R" radius specification permitted for X/Y/Z set only**
- **Trajectory must be segmented with Coord[*x*].SegMoveTime > 0 (typical values are 5 to 10 msec)**
- **Additional features:**
  - Any plane in 3D Cartesian space can be defined
  - Other axes are linearly interpolated: helical (e.g. adding **Z**), tangent axes (e.g. adding **A/B/C**)
  - Automatic spiral generation when *end radius* != *start radius*
  - With virtual "cross" axis, can use for simple sinusoidal profile generation
  - Minimum arc length can be specified in **Coord[*x*].MinArcLen**
  - Circle acceleration can be limited by **Coord[*x*].MaxCirAccel**
  - Spiraling can be limited with **Coord[*x*].RadiusErrorLimit**

Circular Move Mode

- **Commands:**

Interpolation plane definition

Normal {*vector*}{*data*} // Vector: [I,J,K] for Axes [X,Y,Z]
Normal I10 J 10 K -5 // [10,20,-5] is the normal vector of the plane

End point definition mode

Abs / Inc // Absolute / Incremental end point

Center point definition

Always incremental with reference to starting point

Circle direction

Circle1 / Circle2 // Clockwise / Counter-clockwise for X, Y, Z
Circle3 / Circle4 // Clockwise / Counter-clockwise for XX, YY, ZZ

Circle Commands

X{*X Pos.*} Y{*Y Pos.*} Z{*Z Pos.*} I{*data*} J{*data*} K{*data*}
X{*X Pos.*} Y{*Y Pos.*} Z{*Z Pos.*} R{*Radius*}

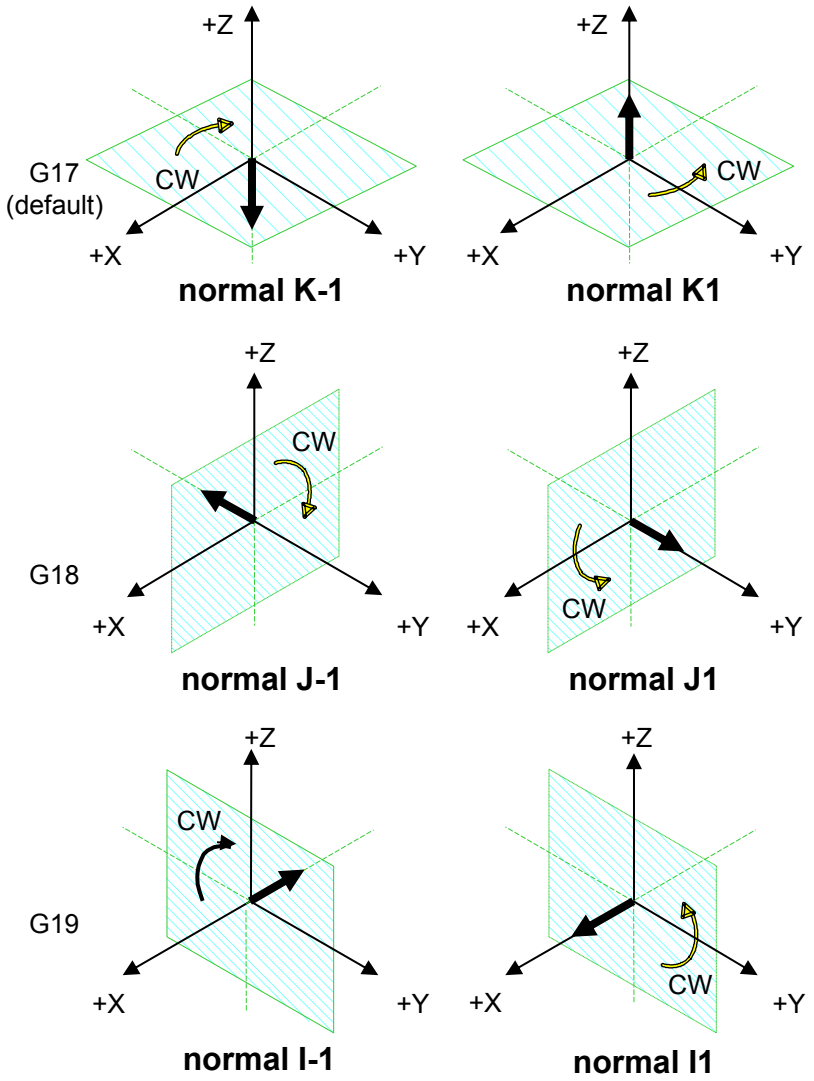

Note

Specifying R instead of I, J, K defines the arc radius along the arc from beginning point to endpoint

Interpolation Plane

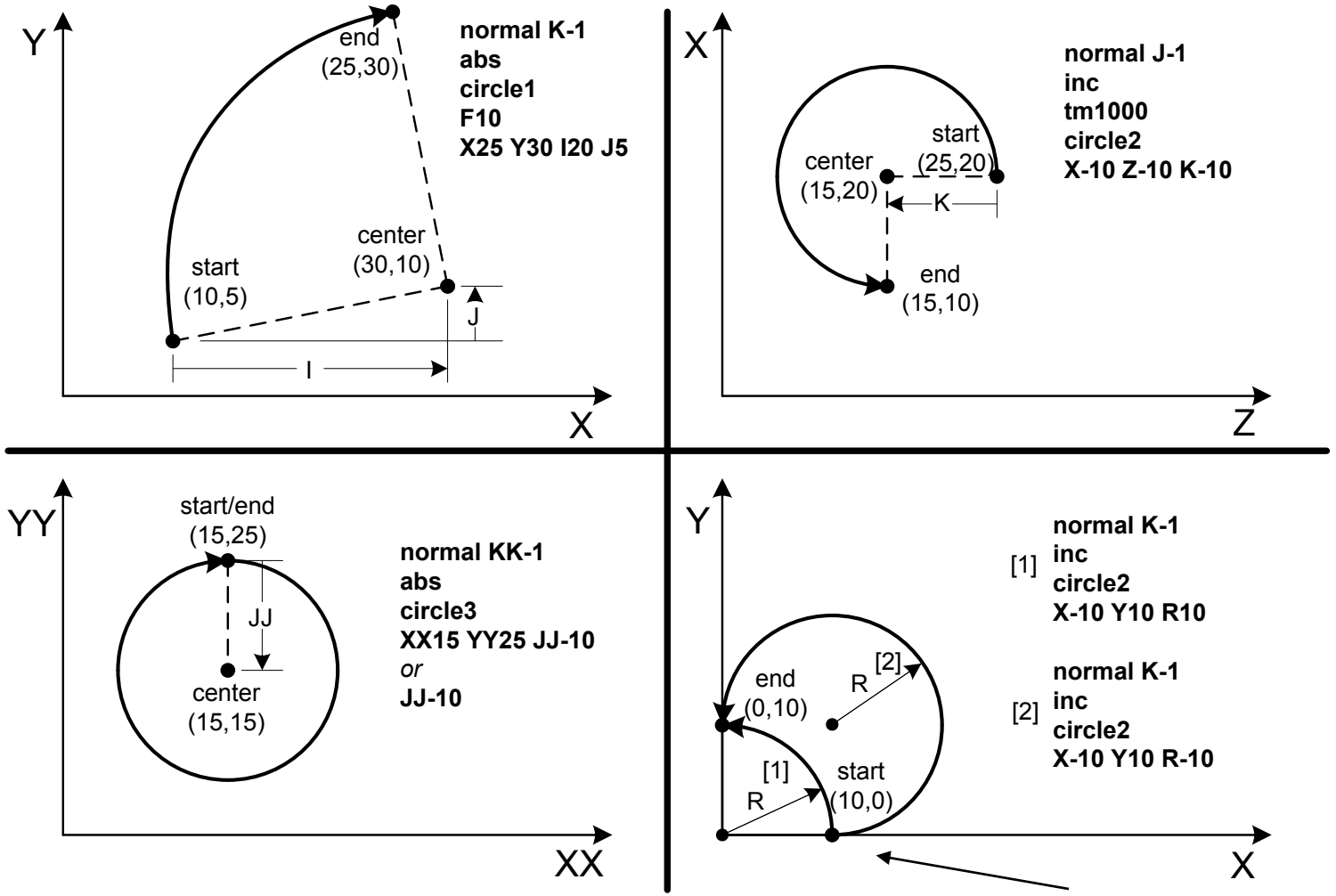
- **Command:** **Normal {*vector*} {*data*}**

  Once the normal vector of the plane is determined, the right-hand rule determines which direction is clockwise



- **Figures show "single-component" vectors with the planes and clockwise arc sense they define**
- **Magnitude of vector does not matter**
- **Negative vector defines clockwise sense according to standards**
- **Combinations of vector components can be used to define "tilted" planes (e.g. normal K-0.866 J-0.5)**
- **Same plane used for 2D tool-radius compensation and corner angle calculations**

Circular Move Examples



Note

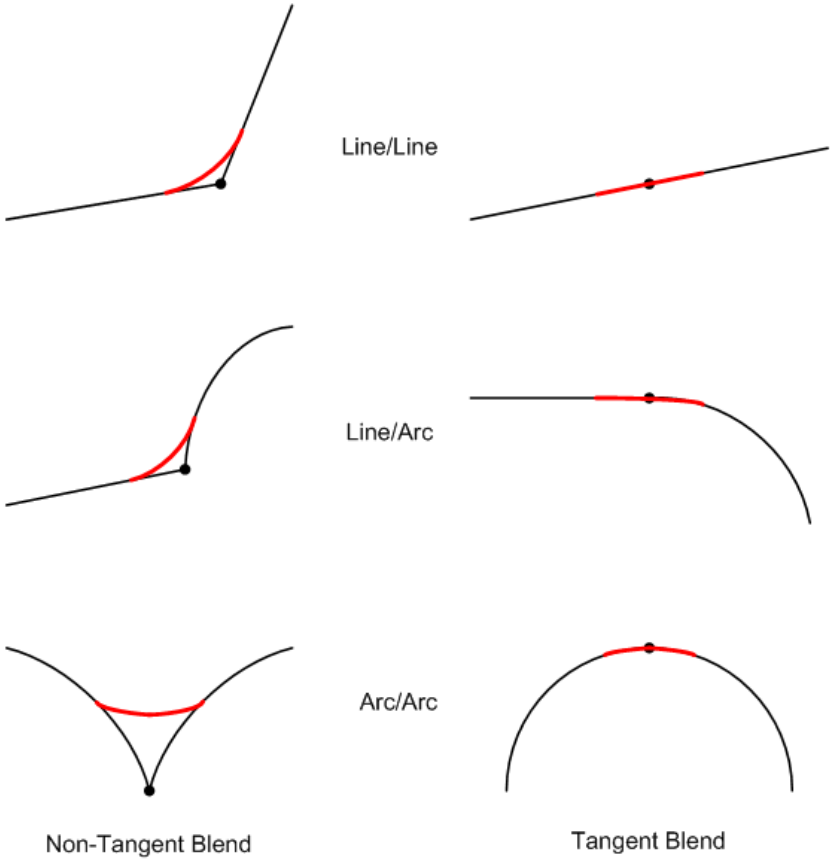
Starting point is specified from previous move. After the Circle command, only the endpoint, and vectors to the Center Point, or Radius of arc, are needed.

R > 0 , path is less than 180°
R < 0 , path is greater than 180°
R should **NOT** be used for generating a full-circle path

Path Blending (Linear/Circle)

- Commanded blended path same in both directions
- If unblended moves would provide sharp corner, blended path provides a rounding to the inside
- Ends of blended path are at points where deceleration to stop would begin, and acceleration from stop would end
- Can control blending with the following parameters:

  Coord[*x*].CornerBlendBp
  Coord[*x*].CornerDwellBp
  Coord[*x*].InPosTimeout
  Coord[*x*].CornerAccel
  Coord[*x*].CornerRadius



Rapid Move Mode

- **Main purpose is minimum-time point-to-point moves, given velocity, acceleration, and jerk constraints**
- **Rapid is the only move mode that can be commanded from PLC programs**
  - No need to declare **rapid** mode in PLC program
  - Any move command automatically changes CS's move mode to **rapid**
- **No blending of rapid move with any other move**
- **Can break into rapid move at any time, even in the middle of a move**
  - Programmed triggered moves (e.g. **X1000^-30**)
    - Occurrence of trigger causes Power PMAC to break into pre-trigger move
    - Post-trigger move of specified distance from position at trigger
    - Cannot use with axes defined by kinematic subroutines
  - Newly issued move command ("altered destination")
    - From PLC program to C.S. specified by program's **Ldata.Coord** value
    - From on-line "**cx**" command (e.g. **cx X19.93 Y31.25**) to addressed C.S.
    - Note that motion program execution suspended until **rapid** move ends
    - Can execute new **rapid** move command every servo cycle

Rapid Move Mode

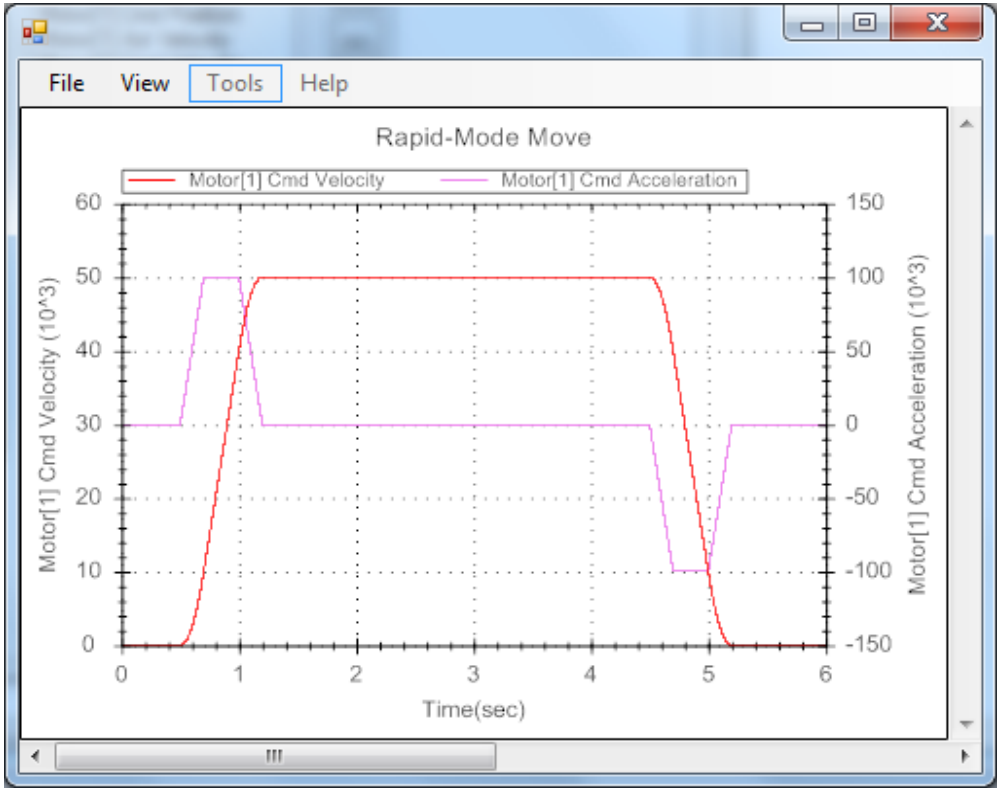
- **Same underlying algorithms as jogging and homing-search moves**
- **Velocity control elements**
  - **Motor[*x*].RapidSpeedSel** specifies which variable controls speed
    - = 0 (default): **Motor[*x*].MaxSpeed** controls magnitude
    - = 1: **Motor[*x*].JogSpeed** controls magnitude
- **Acceleration control elements (used for jogging and homing too)**
  - **Motor[*x*].JogTa** >= 0 sets accel time in msec
  - **Motor[*x*].JogTa** < 0 sets (inverse) accel rate in msec^2/motor unit
- **Jerk control elements (used for jogging and homing too)**
  - **Motor[*x*].JogTs** >= 0 sets S-curve time in msec
  - **Motor[*x*].JogTs** < 0 sets (inverse) jerk rate in msec^3/motor unit
- **When specifying acceleration and jerk by time:**
  - If **JogTa** > **JogTs**, total accel. time = **JogTa** + **JogTs** (with some constant acceleration)
  - If **JogTa** < **JogTs**, total accel. time = 2 * **JogTs** (with no constant acceleration)
- **All parameters are floating-point values; no arbitrary limits**
- **Note: If JogTs < 0, then JogTa must be < 0**

Rapid Mode Move Profile

inc x200

Power PMAC Script

- **Distance = 200,000 m.u.**
- **MaxSpeed = 50 m.u./msec**
- **JogTa = -10 msec^2/m.u. (= 0.1 m.u./msec^2)**
- **JogTs = -2000 msec^3/m.u. (= 0.0005 m.u./msec^3) (for 200 msec to A_{max})**



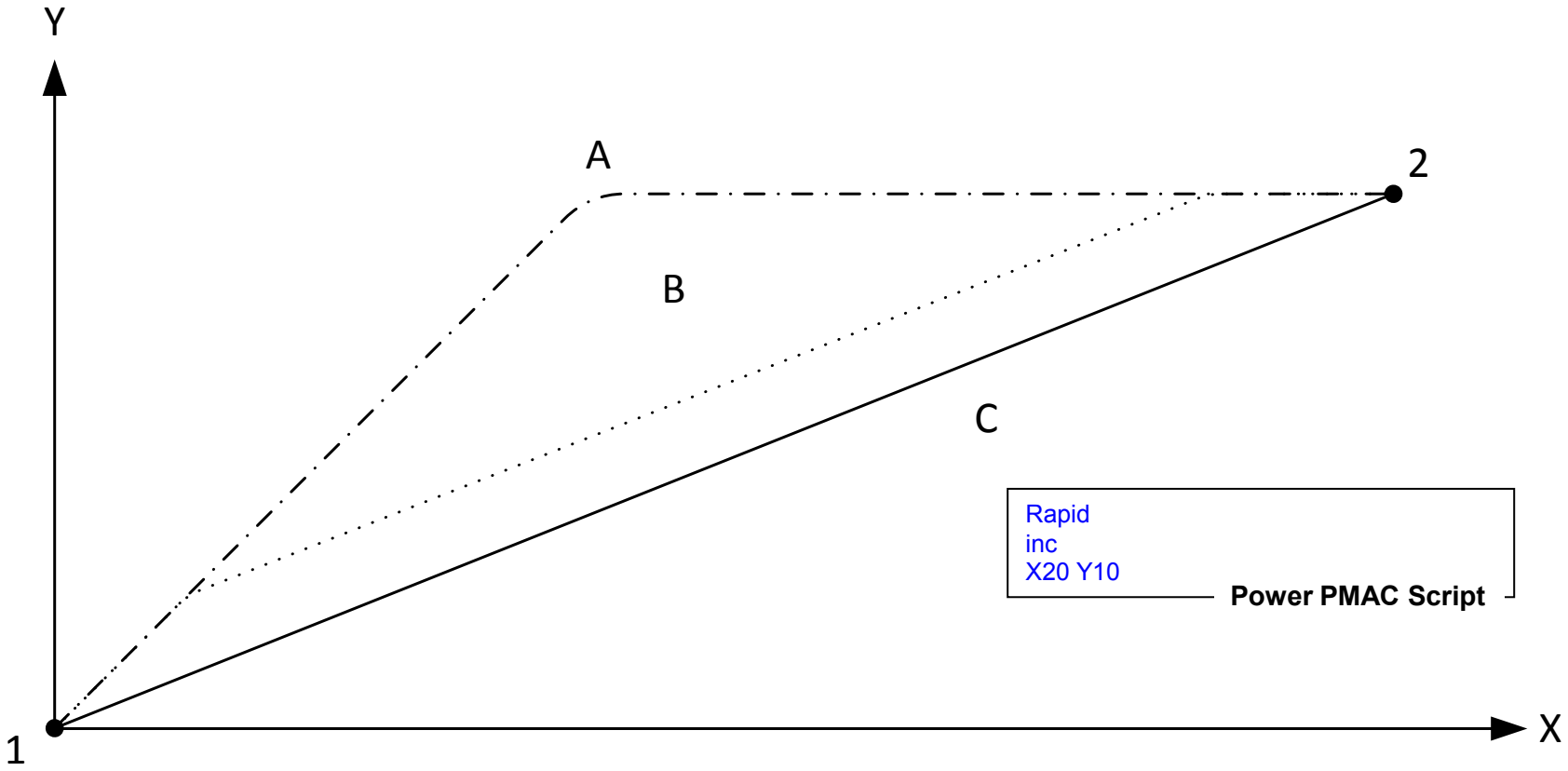
Rapid Mode Triggered Move Profile

```
inc x200^-50
```
Power PMAC Script

- Pre-Trigger Distance <= 200,000 m.u.
- Post-Trigger Distance = -50,000 m.u.
- MaxSpeed = 50 m.u./msec
- JogTa = -10 msec^2/m.u. (= 0.1 m.u./msec^2)
- JogTs = -2000 msec^3/m.u. (= 0.0005 m.u./msec^3) (for 200 msec to A_{max})

Rapid Move Path Control

- **Trajectory from point 1 to point 2**
  - Path A: **Coord[*x*].RapidVelCtrl** = 0 (all motors move at rapid speed)
  - Path B: **Coord[*x*].RapidVelCtrl** = 1 (only "longest" motor moves at rapid speed), accels. specified by the slowest axis's rate
  - Path C: **Coord[*x*].RapidVelCtrl** = 1, all accels. specified by the slowest axis's time
  - Note there is no "vector speed" control in rapid mode



Spline Move Mode

- Spline moves provide cubic B-splines (cubic in terms of the position-vs.-time equations) to blend together a series of points on an axis
- Position, Velocity, and Acceleration are continuous at move boundaries
- Commanded path is to the inside of programmed points

- Each spline move comprises 3 segments of time duration (see next slide)
- Flexible method of specifying segment times:

  spline*{data0}* sets all 3 times to ***{data0}***

  spline*{data0}***spline***{data1}* sets T0 to ***{data0}***, T1 & T2 to ***{data1}***

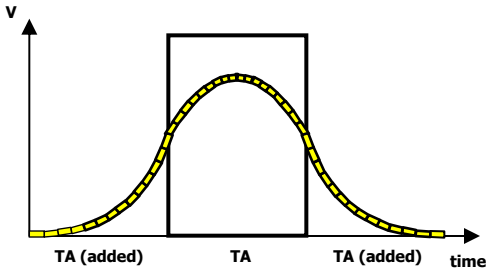
  spline*{data0}***spline***{data1}***spline***{data2}* sets T0 to ***{data0}***, T1 to ***{data1}*** , T2 to ***{data2}***

  Segment 0 for the move is calculated and executed

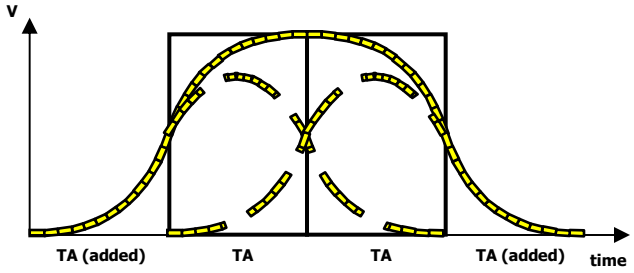
  Segments 1 and 2 are tentatively calculated but not yet executed

  Next programmed spline move can change these times (its T0 is this move's T1; its T1 is this move's T2)

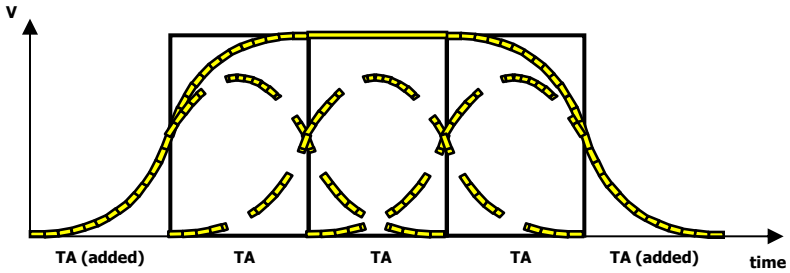
Cubic Spline Move Trajectory



One Programmed Segment



Two Programmed Segments

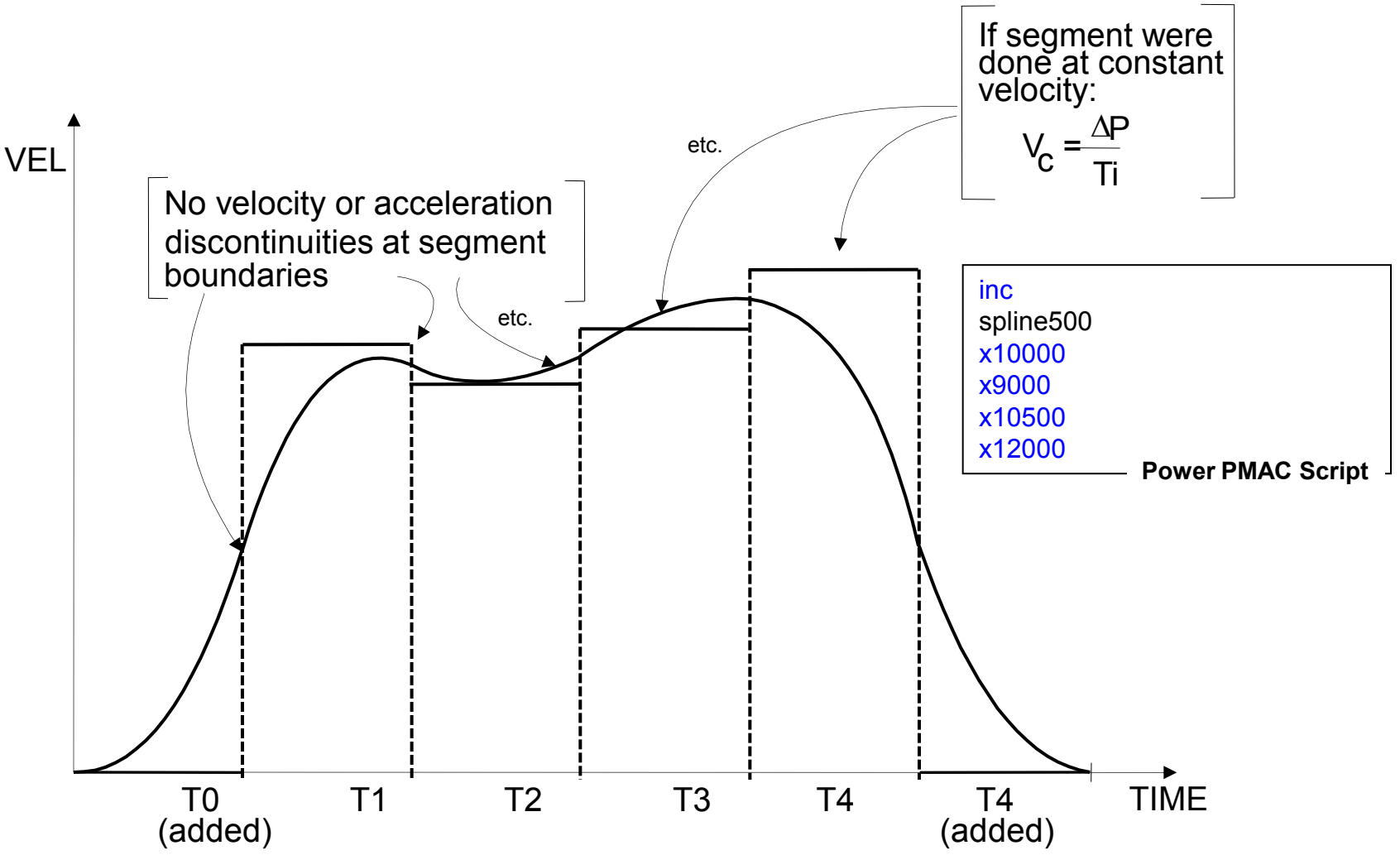


Three Programmed Segments

Note

In the examples above, spline{*data0*} is used to make all three zones for each segment equal (i.e. T0 = T1 = T2 = TA).

Building a Cubic Spline Profile



PVT Move Mode

- PVT stands for Position-Velocity-Time
- PVT Move Mode lets users specify exact position and velocity at the boundaries of moves
- Motion profile passes exactly through programmed positions
- Generates a Hermite-Spline path useful for creating arbitrary profiles (parabolic velocity trajectory). Best mode for contouring.
- Velocity is in axis units per user time unit (as set by Coord[*x*].FeedTime)

➢ **Commands**

pvt {*Time*} // PVT move with Move Time ***Time***

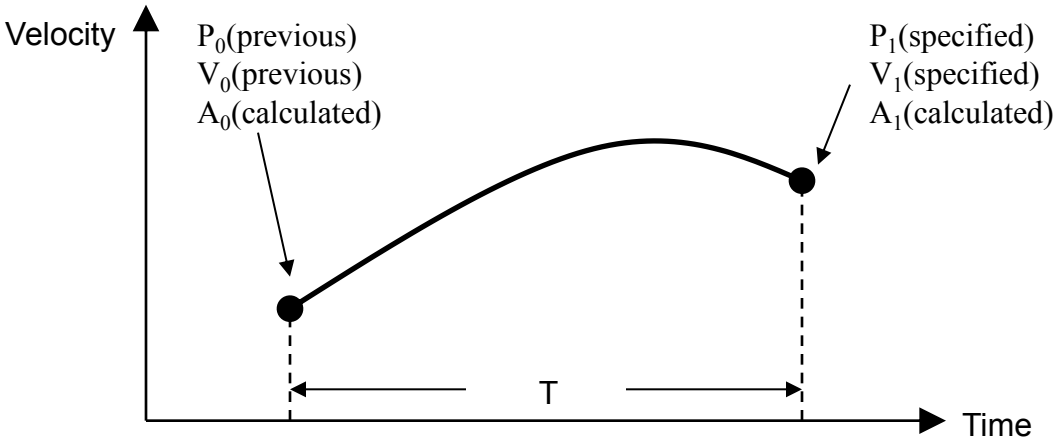
{Axis}{Position}:{Velocity}
// Move statement, endpoint position and
// velocity specified

Example:

```
Inc             // Incremental Move
pvt 1000        // 1000 msec move time
X 20:1.5        //  X endpoint is 20 user units and endpoint speed is 1.5 feedrate units
```

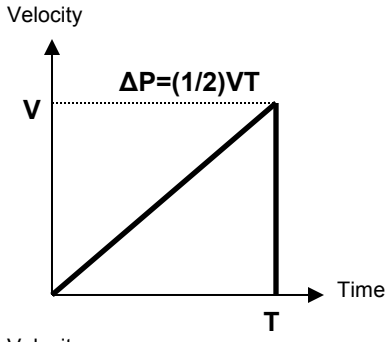
Power PMAC Script

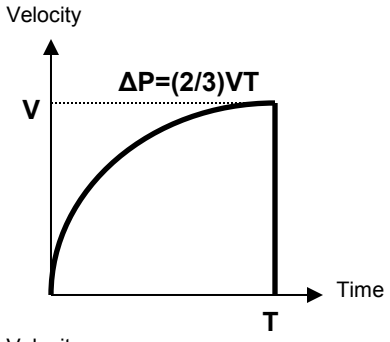
PVT Move Trajectory

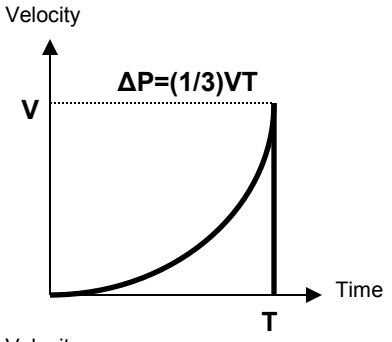


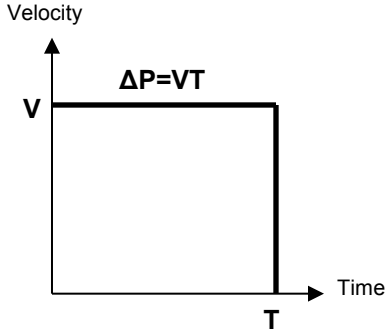
P: Cubic Position
V: Parabolic Velocity
A: Linear Acceleration
T: Move Time

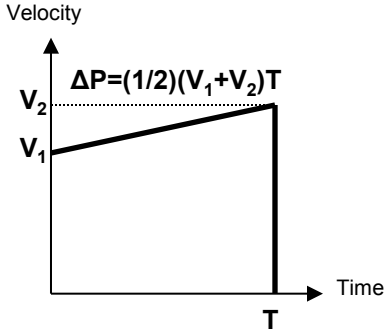
Common Segment Shapes

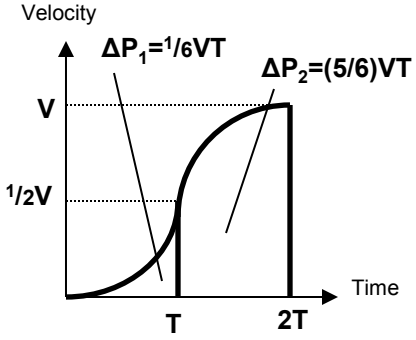




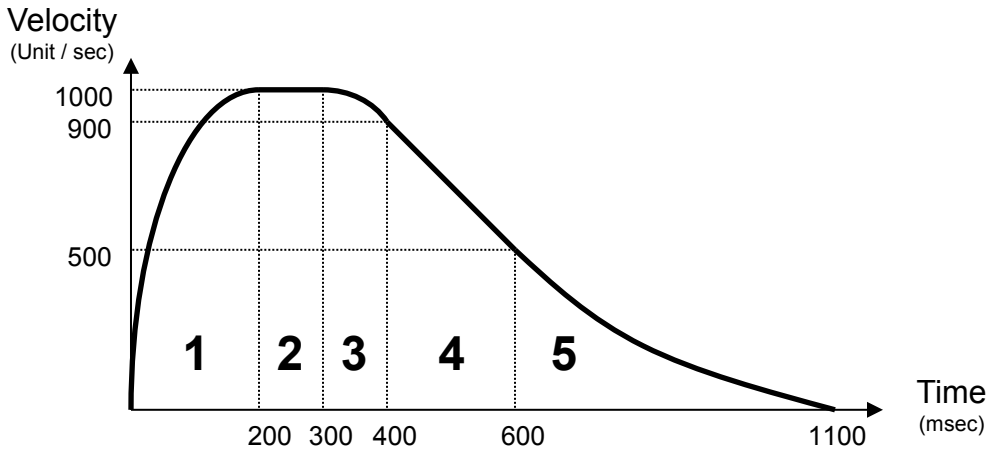








PVT Move Example




```
undefine all
&1              // Address C.S. 1
#1->100X        // 100 counts is 1 user unit for X axis
open prog 4     // Open Program 4 buffer and clear
inc             // Incremental endpoint definition
PVT 200         // PVT mode with move time T=200 msec
X 133.333:1000              // Move 1: ΔP=133.333, V=1000
PVT 100         // Change PVT move time to T=100 msec
X 100:1000      // Move 2: ΔP=100, V=1000
X 96.667:900// Move 3: ΔP=96.667, V=900
PVT 200         // Change PVT move time to T=200 msec
X 140:500       // Move 4: ΔP=140, V=500
PVT 500         // Change PVT move time to T=500 msec
X 83.333:0      // Move 5: ΔP=83.333, V=0
close           // Close program buffer
```

Power PMAC Script

Enhanced Inter-Mode Blending

- **Keeps ability to blend on the fly between linear and circle-mode moves**
  Uses **Coord[*x*].Ta** and **Coord[*x*].Ts** values in force at the time

- **Blend on the fly between linear/circle-mode moves and pvt-mode moves**
  - Useful for creating custom accel./decel. profiles
  - Direct transition between **pvt** move and constant-velocity portion of **linear** or **circle** move; does not use **Coord[*x*].Ta** or **Coord[*x*].Ts**
  - When blending from **pvt** to **linear** or **circle**, must match axis velocities exactly or will get step change at transition
  - When blending from **linear** or **circle** to **pvt**, ending axis velocity of incoming move is automatically used as starting velocity for **pvt** move
  - Segmentation mode must be active (**Coord[*x*].SegMoveTime** > 0)

Example Linear Move with PVT Accel//Decel

```
inc;
pvt1000;
x20:30;
linear;
x30 f30;
pvt1000;
x20:0;
```

Power PMAC Script