DELTA
COMPUTER SYSTEMS, INC.

Quadratic (Second Order) Interpolation

This document analyzes quadratic or second order interpolation. Quadratic interpolation is used by some motion controllers to interpolate between positions sent to the motion controller from a master motion profile planner and coarse time intervals, Δt. The motion controller then interpolates between these coarse positions to generate fine positions at fine update intervals, δt. The purpose of this document is to point out the limitations of quadratic interpolation.

Quadratic interpolation usually uses three positions to generate three coefficients for a second order equation in the form of:

$s(t)=a+b*t+c*t^2$

The three coefficients can be solved for by solving three simultaneous equations. Using Mathcad makes this easy. The time between the three points, Δt, is constant and assumed to be the coarse update time.

Given

$$a + b\cdot(-\Delta t) + c\cdot(-\Delta t)^2 = s0$$

Previous coarse position at Δt

$$a = s1$$

Current coarse position at 0

$$a + b\cdot\Delta t + c\cdot\Delta t^2 = s2$$

Future coarse position at Δt

$$\text{Find}(a,b,c) \rightarrow \begin{pmatrix} s1 \\ \frac{1}{2}\cdot\frac{s2 - s0}{\Delta t} \\ \frac{-1}{2}\cdot\frac{2\cdot s1 - s2 - s0}{\Delta t^2} \end{pmatrix}$$

$$a = s1$$

$$b = \frac{1}{2}\cdot\frac{s2 - s0}{\Delta t}$$

$$c = \frac{1}{2}\cdot\frac{s2 - 2\cdot s1 + s0}{\Delta t^2}$$

Quadratic Interpolation

The motion controller receives a new point, s2, from the motion profile generator just as the interpolated position reaches the middle point, s1. At this point the previous s1 is copied to s0 and the previous s2 is copied to s1. The controller then interpolates between s1 and the new s2 for the next coarse update period.

The velocity and acceleration must also be computed as well as the position at each fine update point. The velocity and acceleration are necessary computing the velocity and acceleration feed forward terms of the control output. The formulas for the velocity and acceleration are:

$v(t)=b+c*t$ and $a(t)=2*c$

One should notice that the position profile is a parabola, the velocity a straight line or linear ramp and the acceleration is constant. Computing a jerk, the derivative of acceleration, is not possible because the derivative of a constant acceleration is zero. One can tell if the motion controller uses second order interpolation by setting all the gains to zero, except for the acceleration feed forward gain, and making a point to point move. If the control output changes in steps then the controllers interpolation or motion profile generator is second order. See the graphs of the interpolated accelerations to see an example of what the interpolated acceleration looks like when using quadratic interpolation. On many systems the user doesn't have access to the position, velocity and accelerations so the only way to check is to use an oscilloscope and look at the analog control output if there is one.

Graphing Quadratic Motion Profiles

Below are 4 examples of quadratic motion profiles where the master uses a third order motion profile , X(t), to generate the exact coarse position, velocity and acceleration at each coarse update. The controller then uses the coarse position to interpolate position, velocity and acceleration between the coarse update point at fine update intervals. The first three examples show the motion profile of the last 20 milliseconds of a move to 2 inches and the 20 millisecond after the exact motion profile is complete. Each example show how the resolution of the coarse updates affects the interpolated fine position, velocity and acceleration. This highlight some of the flaws in using only positions at each coarse update. A coarse update time of 10 millisecond was chosen so there are 5 coarse update points (0-4). One can see that the interpolated motion profile has a tendency to over shoot the set point of 2 inches at 20 milliseconds. The velocity and acceleration are not zero until the coarse update at 30 milliseconds. The fourth example shows how well quadratic interpolates higher order functions like a sinusoid.

Quadratic Interpolation Using Floating Point

The motion profile shows 40 milliseconds as the target approaches the command position on a point to point move. The motion profile is assumed to be a third order motion profile. The interpolation errors can be significant when a lower order interpolation is used to follow a third order motion profile. A third order motion profile was chosen because most motion controllers use third order motion profiles.

Quadratic Interpolation

$j := 10000$

Jerk, typical value to reach 100 in/sec^2 in 10 milliseconds

$$X(t) := \left| \begin{array}{ll} \begin{bmatrix} 2 + \frac{j}{6}\cdot(t - 0.020)^3 \\ \frac{1}{2}\cdot j\cdot(t - 0.020)^2 \\ j\cdot(t - 0.020) \end{bmatrix} & \text{if } t < 0.020 \\ \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} & \text{otherwise} \end{array} \right.$$

Exact 3rd order target position, velocity and acceleration approaching a command position of 2 inches over a period of 0.02 seconds

$\Delta t := 0.010$

Coarse update interval

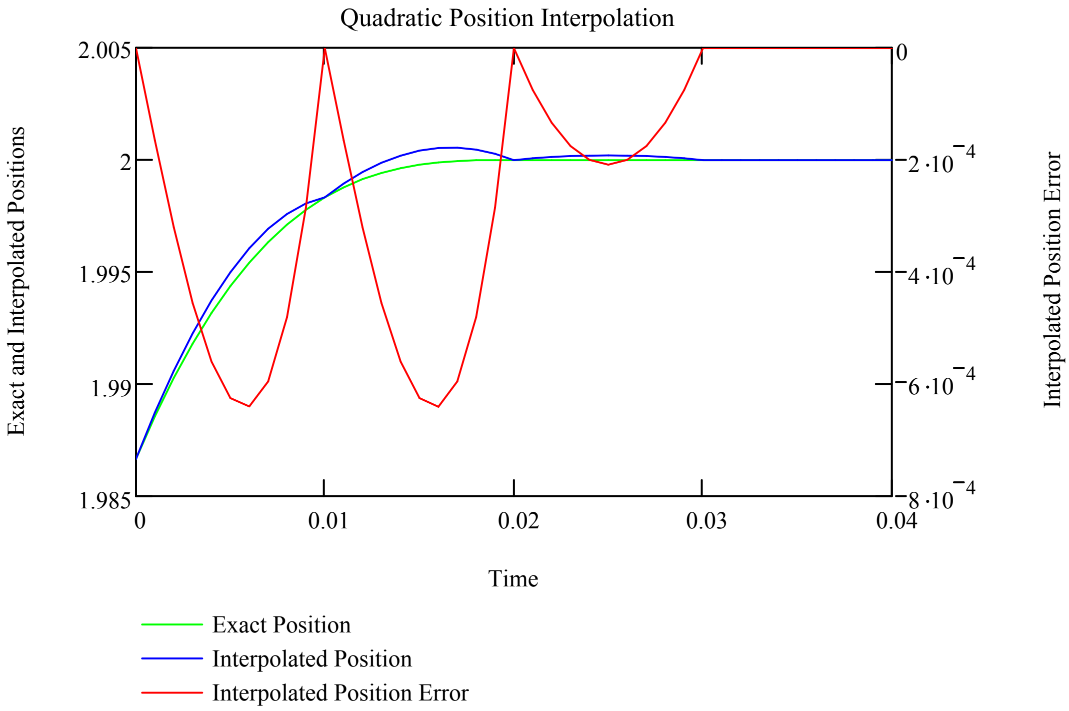
$\delta t := 0.001$

Fine update interval

```
x(t) :=
    $m \leftarrow \mathrm{floor}\left(\frac{t}{\Delta t}\right)$          Computer the update interval
    $t \leftarrow t - m\cdot\Delta t$                                      Computer the time within the coarse update
    $x_{m1} \leftarrow X[(m-1)\cdot\Delta t]_0$                            $x_{m1}$=the previous coarse update position
    $x_0 \leftarrow X(m\cdot\Delta t)_0$                                   $x_0$=the current coarse update position
    $x_1 \leftarrow X[(m+1)\cdot\Delta t]_0$                               $x_1$=the next coarse update position
    $A \leftarrow x_0$                                                     A=the current coarse update position
    $B \leftarrow \frac{x_1 - x_{m1}}{2\cdot\Delta t}$                     B=velocity at the current coarse update
    $C \leftarrow \frac{x_1 - 2\cdot x_0 + x_{m1}}{2\cdot\Delta t^2}$      C=acceleration/2
    $\begin{pmatrix} A + B\cdot t + C\cdot t^2 \\ B + 2\cdot C\cdot t \\ 2\cdot C \end{pmatrix}$    Return position
                                                                           velocity and
                                                                           acceleration
```

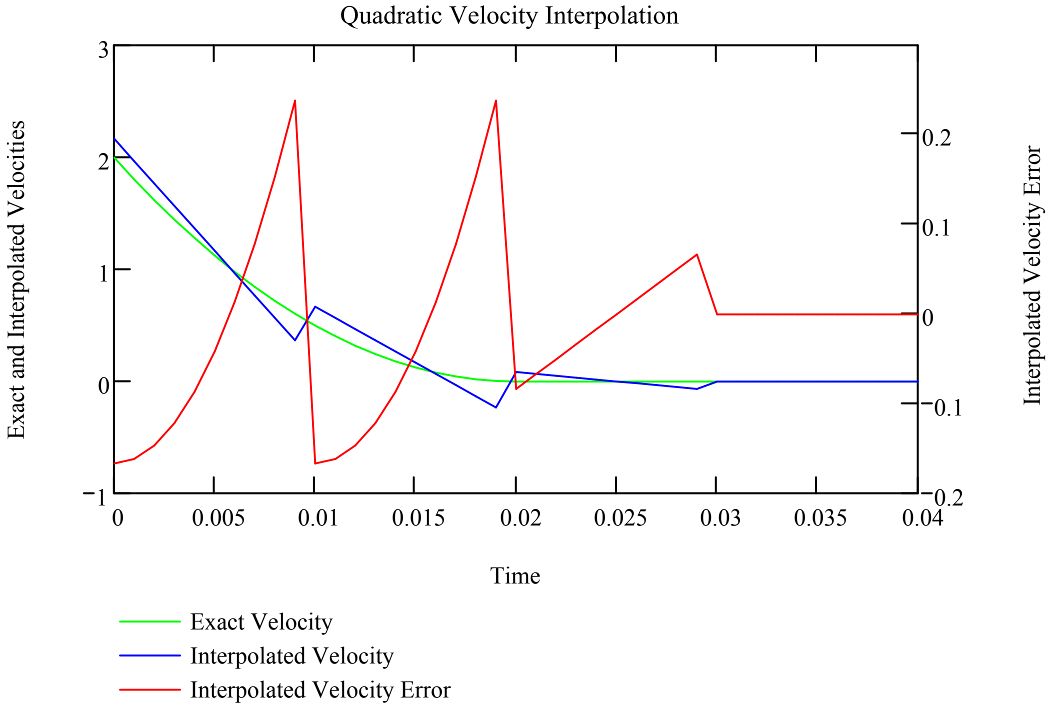
Quadratic Interpolation

$t := 0, \delta t .. 0.040$



The motion profile for the positions does not look smooth. It looks bumpy and over shoots the command position until the next coarse update at 0.030 seconds. The error is not great but it is not good for machine tool work. This graph shows the errors as a function of time between the prefect motion profile and and the interpolated motion profile. The errors are not big in this case

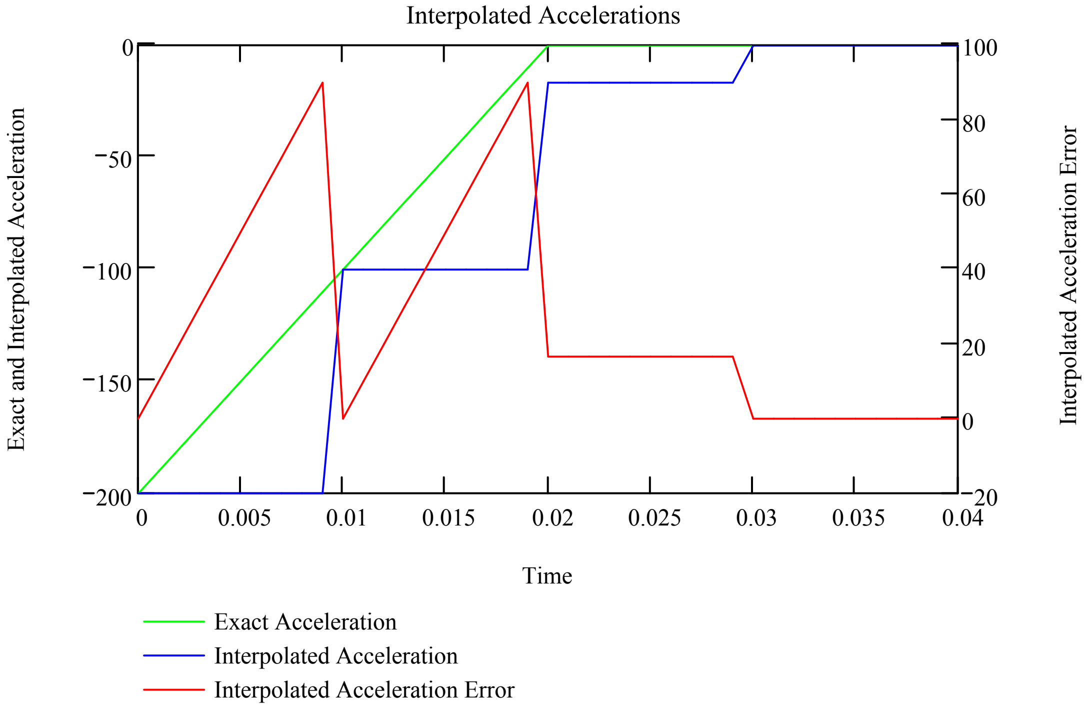
Quadratic Interpolation



The interpolation errors become more apparent at the higher derivatives. The maximum velocity error is 0.23833 inches per second. Multiply this error by 10 volts and divide by the maximum speed to compute the ripple in the control output due to using feed forwards on an imperfect target velocity.

$$\frac{0.23833\cdot\frac{\text{in}}{\text{sec}}\cdot 10\cdot\text{volt}}{30\cdot\frac{\text{in}}{\text{sec}}} = 0.079443\,\text{volt}$$ Not good but not too bad.

Quadratic Interpolation



One can see that quadratic interpolation for accelerations doesn't work very well. The acceleration changes in steps of about 100 inches/second^2 which is more than one tenth of a g. The acceleration interpolation error is as much as 90 in/second$^{2.}$ Using the same example above with the maximum velocity of 30 inches per second and assuming the time constant of the system is 0.1 seconds then control output error due to acceleration feed forwards is:

$$\frac{90\cdot\frac{\text{in}}{\text{sec}^2}\cdot 10\cdot\text{volt}\cdot 0.1\cdot\text{sec}}{30\cdot\frac{\text{in}}{\text{sec}}} = 3\,\text{volt}$$

A 3 volt error in the control output is significant. It can excite oscillations and cause wear and tear on valves. Normally the user would simply reduce the acceleration feed forward gains or not use them at all resulting in poor performance.

Quadratic Interpolations Using Encoder Counts and Integers

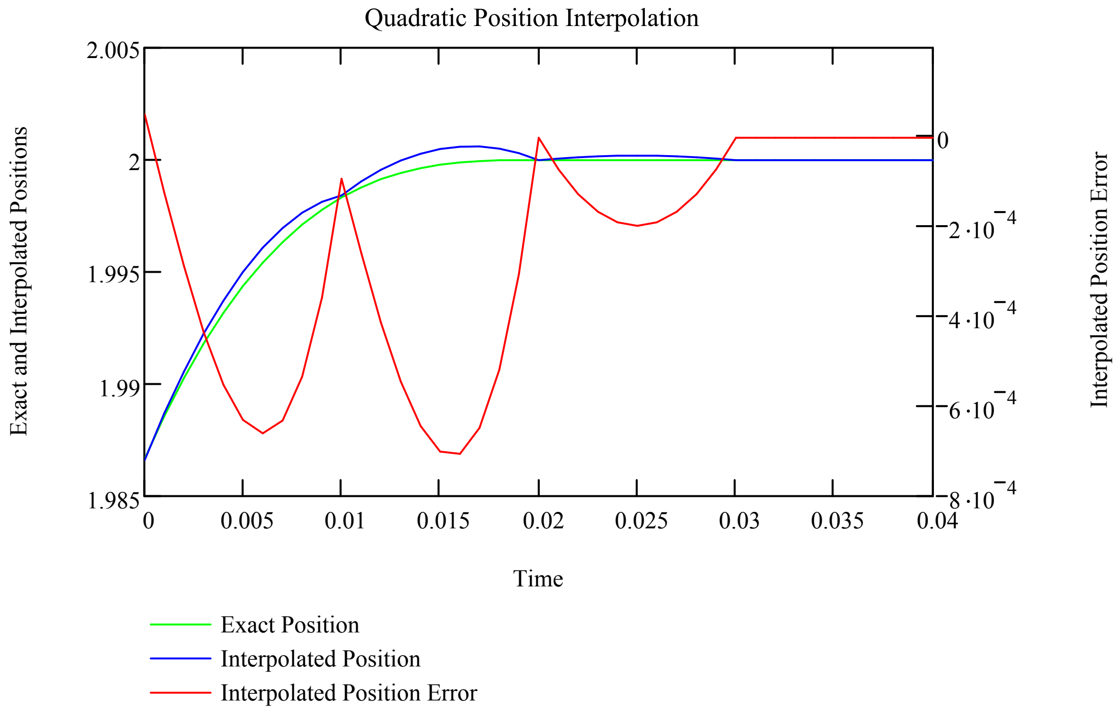
The following are the same examples except that the perfect positions from X(t) are truncated to 5 microns or 0.000197 inches. X(t) is the exact third order position velocity and acceleration. x(t) is the interpolated position, velocity and acceleration taking into account that the exact position have be rounded to the nearest feed back count.

$\text{resolution} := \frac{5}{25400}$ $\text{resolution} = 0.000197$ Feed back resolution of a 5 micron SSI rod

```
$x(t) :=$
    $m \leftarrow \text{floor}\left(\frac{t}{\Delta t}\right)$    Computer the update interval
    $t \leftarrow t - m \cdot \Delta t$    Computer the time within the coarse update
    $x_{m1} \leftarrow \text{Round}\left[X\left[(m-1)\cdot\Delta t\right]_0, \text{resolution}\right]$    $x_{m1}$=the previous coarse update position
    $x_0 \leftarrow \text{Round}\left(X(m\cdot\Delta t)_0, \text{resolution}\right)$    $x_0$=the current coarse update position
    $x_1 \leftarrow \text{Round}\left[X\left[(m+1)\cdot\Delta t\right]_0, \text{resolution}\right]$    $x_1$=the next coarse update position
    $A \leftarrow x_0$    A=the current coarse update position
    $B \leftarrow \frac{x_1 - x_{m1}}{2\cdot\Delta t}$    B=velocity at the current coarse update
    $C \leftarrow \frac{x_1 - 2\cdot x_0 + x_{m1}}{2\cdot\Delta t^2}$    C=acceleration/2
    $\begin{pmatrix} A + B\cdot t + C\cdot t^2 \\ B + 2\cdot C\cdot t \\ 2\cdot C \end{pmatrix}$    Return position velocity and acceleration
```

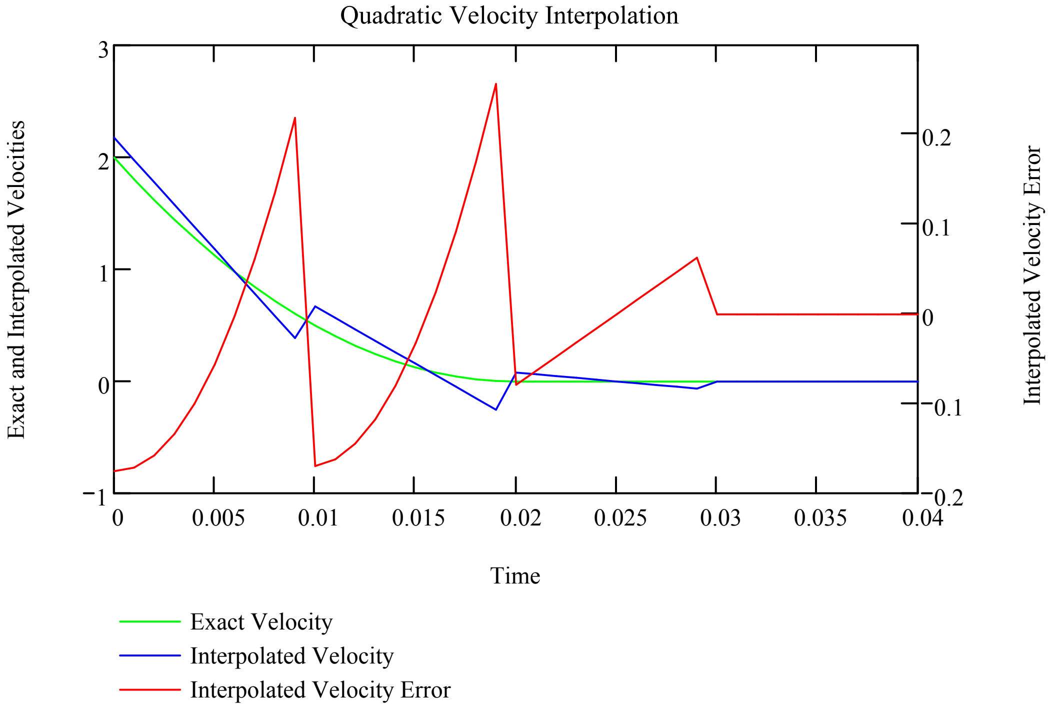
Quadratic Interpolation

$t := 0, \delta t .. 0.040$



The motion profile generated using integer position instead of floats is similar to the motion profile using floats because the resolution of the SSI encoder is relatively high. The same over shoot problem can be seen but it isn't too bad.

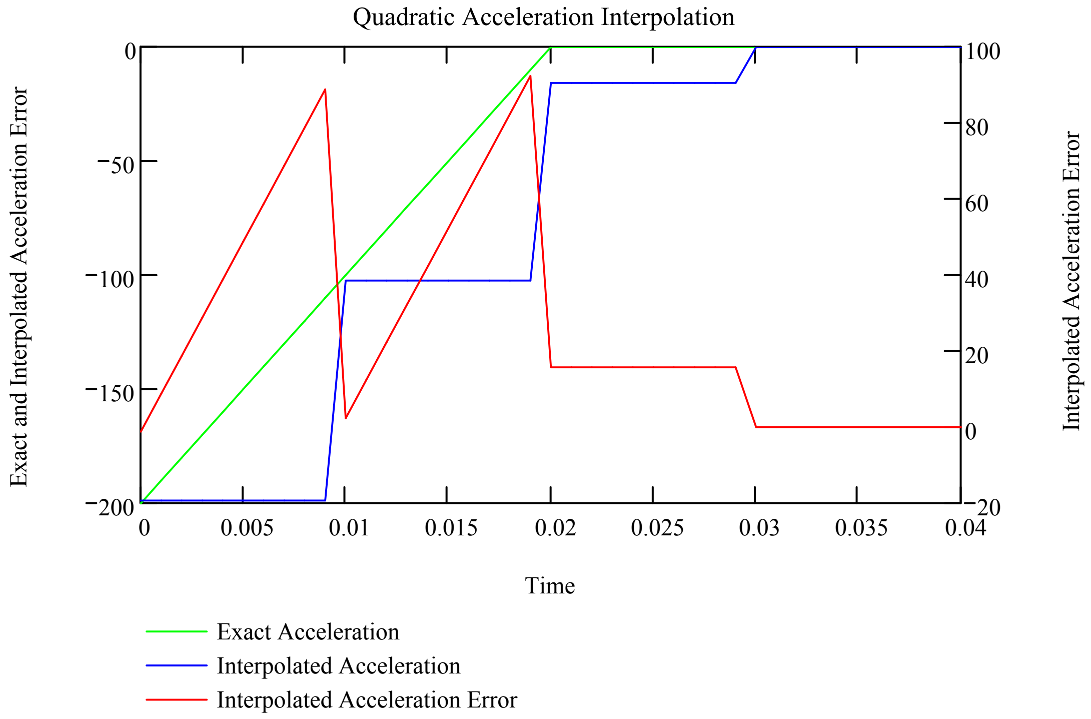
Quadratic Interpolation



The interpolation errors become more apparent at the higher derivatives. The maximum velocity error is 0.25697 inches per second. This error is much worse than the interpolation error using floating point numbers. Multiply this error by 10 volts and divide by the maximum speed to computer the ripple on the control output due to using feed forwards on an imperfect target velocity

$$\frac{0.25697\cdot\frac{\mathrm{in}}{\mathrm{sec}}\cdot 10\cdot\mathrm{volt}}{30\cdot\frac{\mathrm{in}}{\mathrm{sec}}} = 0.085657\,\mathrm{volt}$$ Not good but not too bad.

Quadratic Interpolation



One can see that quadratic interpolation for accelerations doesn't work very well. The acceleration changes in steps of about 100 inches/second^2 which is more than one tenth of a g. The acceleration interpolation error is as much as 92.362 in/second$^{2\cdot}$ This too is noticeably worse using integers rather that floats. Using the same example above with the maximum velocity of 30 inches per second and assuming the time constant of the system is 0.1 seconds then control output error due to acceleration feed forwards is:

$$\frac{92.362\cdot\frac{\mathrm{in}}{\mathrm{sec}^2}\cdot 10\cdot \mathrm{volt}\cdot 0.1\cdot \mathrm{sec}}{30\cdot\frac{\mathrm{in}}{\mathrm{sec}}} = 3.078733\,\mathrm{volt}$$

This is very bad

A 3.078733 volt error in the control output is significant. It can excite oscillations and cause wear and tear on valves. Normally the user would simply reduce the acceleration feed forward gains or not use them at all resulting in poor performance.

Quadratic Interpolations Using Encoder Counts and Integers

The following are the same examples except that the perfect positions from s(t) are truncated to 0.001 inches. X(t) is the exact third order position velocity and acceleration. x(t) is the interpolated position, velocity and acceleration taking into account that the exact position have be rounded to the nearest feed back count.

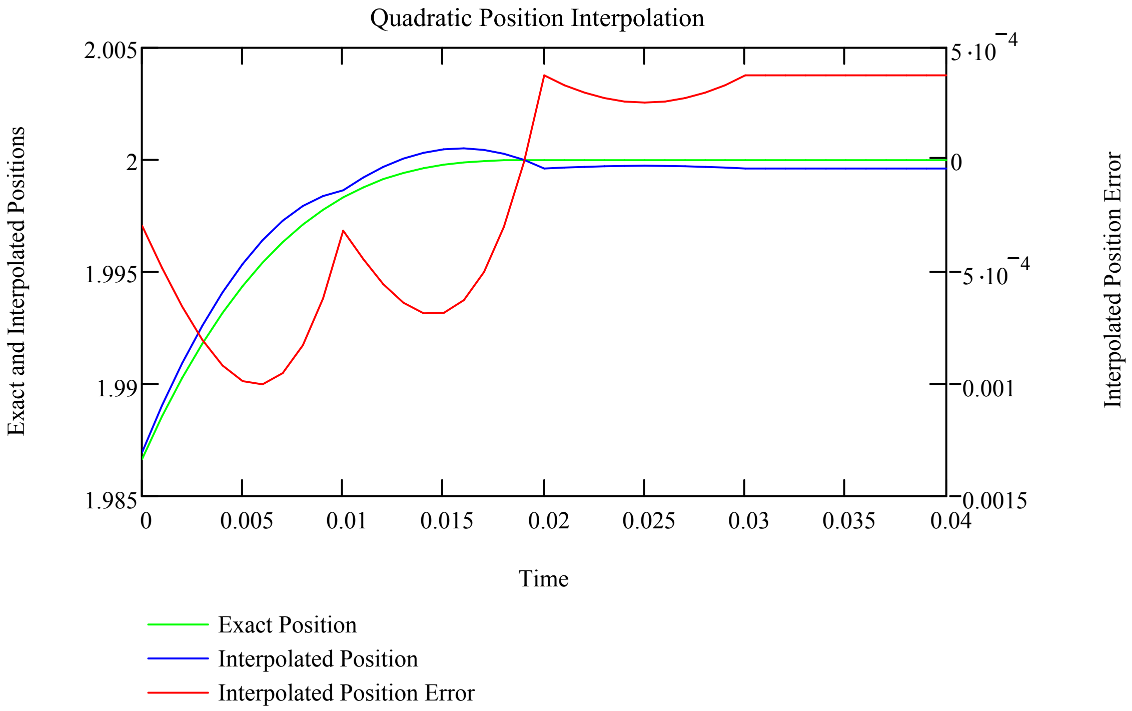
$\text{resolution} := 0.000974$

Feed back resolution in inches. A little better than 0.001 inches. It depends on the rod gradient

```
x(t) :=
    $m \leftarrow \text{floor}\left(\frac{t}{\Delta t}\right)$                                  Computer the update interval
    $t \leftarrow t - m \cdot \Delta t$                                              Computer the time within the coarse update
    $x_{m1} \leftarrow \text{Round}\left[X\left[(m-1)\cdot\Delta t\right]_0, \text{resolution}\right]$       $x_{m1}$=the previous coarse update position
    $x_0 \leftarrow \text{Round}\left(X(m\cdot\Delta t), \text{resolution}\right)_0$                 $x_0$=the current coarse update position
    $x_1 \leftarrow \text{Round}\left[X\left[(m+1)\cdot\Delta t\right]_0, \text{resolution}\right]$        $x_1$=the next coarse update position
    $A \leftarrow x_0$                                                   A=the current coarse update position
    $B \leftarrow \frac{x_1 - x_{m1}}{2\cdot\Delta t}$                                     B=velocity at the current coarse update
    $C \leftarrow \frac{x_1 - 2\cdot x_0 + x_{m1}}{2\cdot\Delta t^2}$                              C=acceleration/2
    $\begin{pmatrix} A + B\cdot t + C\cdot t^2 \\ B + 2\cdot C\cdot t \\ 2\cdot C \end{pmatrix}$                       Return position
                                                             velocity and
                                                             acceleration
```

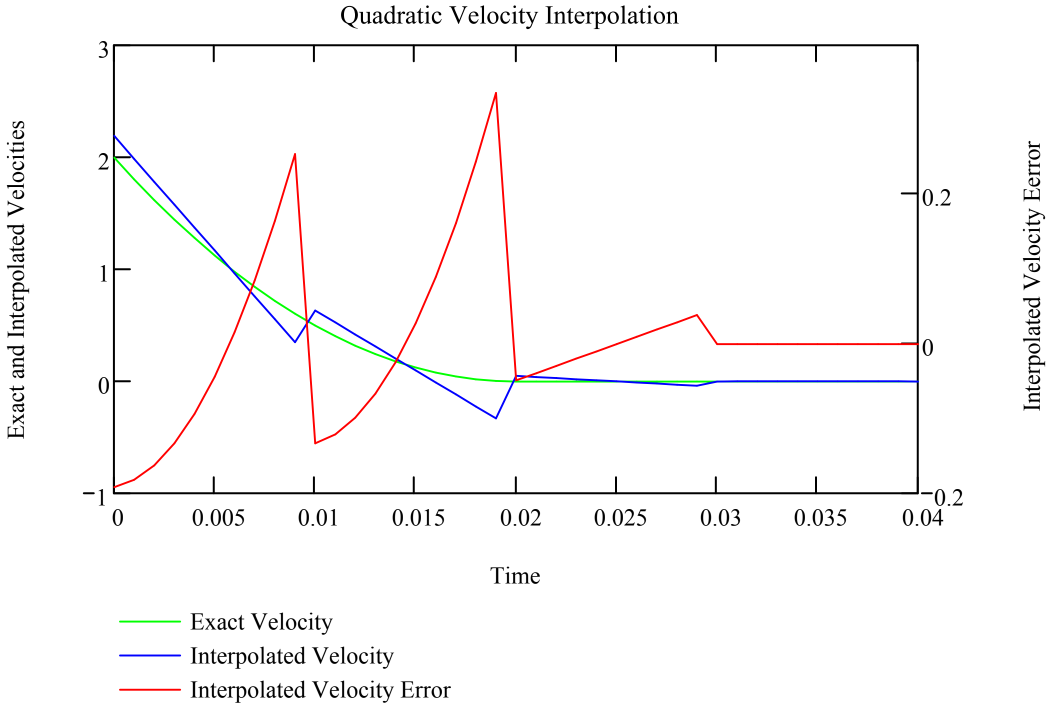
Quadratic Interpolation

$t := 0, \delta t .. 0.040$



The motion profile for the positions does not look smooth. The target position does not reach the command position because the command position does not equate to an integer number of counts. The closest count is 0.0004 inches away from the command position. Still the interpolated target does steady state until 30 milliseconds when it should be stopped at 20 milliseconds. The error is not great but it is not good for machine tool work.

Quadratic Interpolation

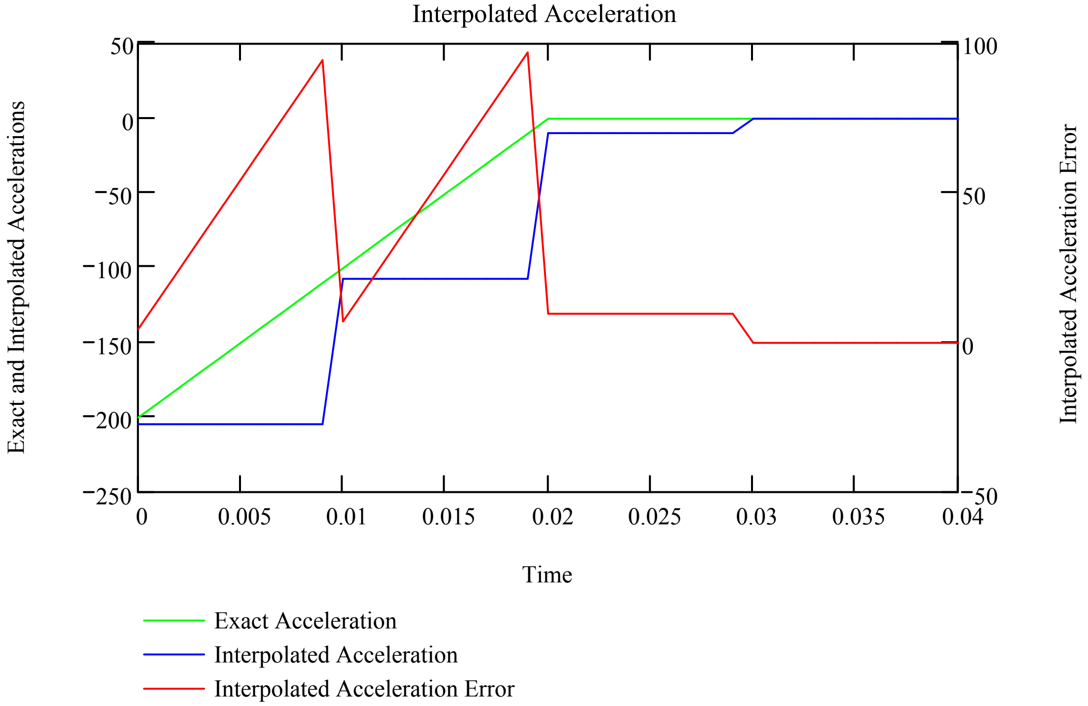


The interpolation errors become more apparent at the higher derivatives. The maximum velocity error is 0.33616 inches per second. Multiply this error by 10 volts and divide by the maximum speed to computer the ripple on the control output due to using feed forwards on an imperfect target velocity. In this case the velocity error went down.

$$\frac{0.33616\cdot\frac{\text{in}}{\text{sec}}\cdot 10\cdot\text{volt}}{30\cdot\frac{\text{in}}{\text{sec}}} = 0.112053\,\text{volt}$$

Not good. That is significant error in the control output

Quadratic Interpolation



One can see that quadratic interpolation for accelerations doesn't work very well. The acceleration changes in steps of about 100 inches/second^2 which is more than one tenth of a g. The acceleration interpolation error is as much as 97.14 in/second$^{2\cdot}$ Using the same example above with the maximum velocity of 30 inches per second and assuming the time constant of the system is 0.1 seconds then control output error due to acceleration feed forwards is:

$$\frac{97.14\cdot\frac{\text{in}}{\text{sec}^2}\cdot 10\cdot\text{volt}\cdot 0.1\cdot\text{sec}}{30\cdot\frac{\text{in}}{\text{sec}}} = 3.238\,\text{volt}$$

Very bad

A 3.238 volt error in the control output is significant. It can excite oscillations and cause wear and tear on valves. Normally the user would simply reduce the acceleration feed forward gains or not use them at all resulting in poor performance.

Quadratic Interpolations Using Encoder Counts and Integers for a Sine Wave

The following are the same examples except that the perfect positions from s(t) are truncated to roughly 0.001 inches. The actual resolution depends on the MDT rod gradient and the controllers counter frequency but normally the resolution is better than 0.001 inches

$\mathrm{Amp} := 1$

$\mathrm{Hz} := 2$

$$X(t) := \mathrm{Amp}\cdot\begin{bmatrix} \sin(2\cdot\pi\cdot\mathrm{Hz}\cdot t) \\ \cos(2\cdot\pi\cdot\mathrm{Hz}\cdot t)\cdot 2\cdot\pi\cdot\mathrm{Hz} \\ -\sin(2\cdot\pi\cdot\mathrm{Hz}\cdot t)\cdot(2\cdot\pi\cdot\mathrm{Hz})^2 \end{bmatrix}$$

Perfect target, velocity and acceleration generated from a sine function

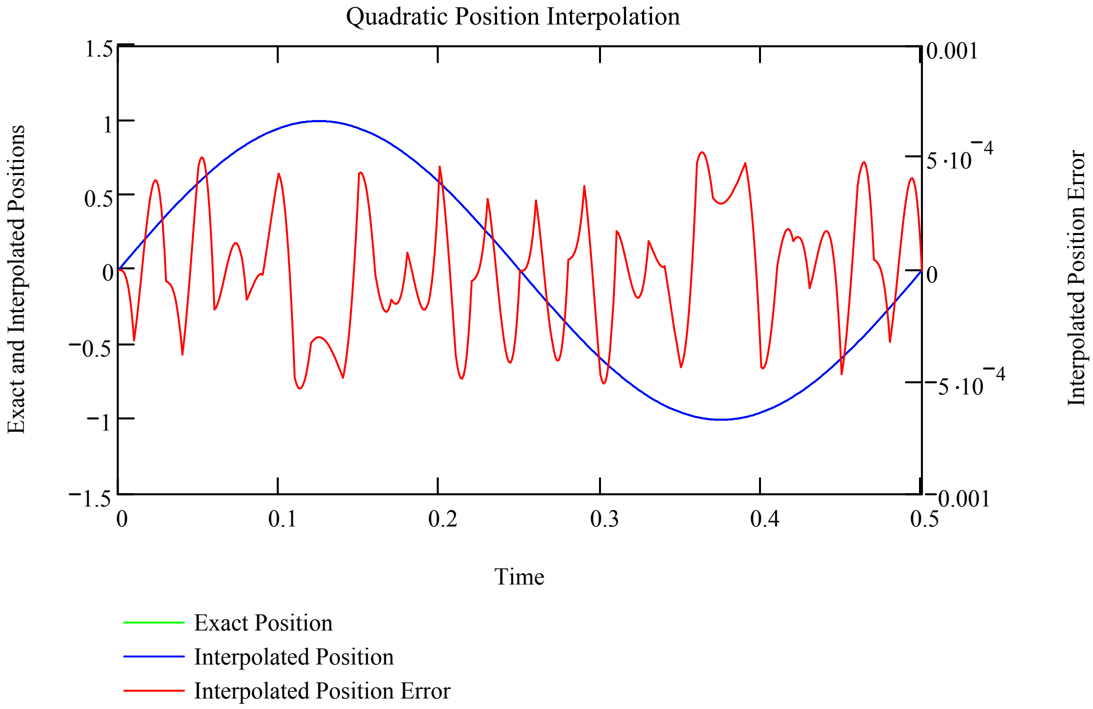
$\mathrm{resolution} := 0.000974$

Feed back resolution in inches. A little better than 0.001 inches. It depends on the rod gradient

```
x(t) :=  $m \leftarrow \mathrm{floor}\left(\frac{t}{\Delta t}\right)$    Computer the update interval
         $t \leftarrow t - m\cdot\Delta t$    Computer the time within the coarse update
         $x_{m1} \leftarrow \mathrm{Round}\left[X[(m-1)\cdot\Delta t]_0, \mathrm{resolution}\right]$    $x_{m1}$=the previous coarse update position
         $x_0 \leftarrow \mathrm{Round}\left(X(m\cdot\Delta t)_0, \mathrm{resolution}\right)$    $x_0$=the current coarse update position
         $x_1 \leftarrow \mathrm{Round}\left[X[(m+1)\cdot\Delta t]_0, \mathrm{resolution}\right]$    $x_1$=the next coarse update position
         $A \leftarrow x_0$    A=the current coarse update position
         $B \leftarrow \frac{x_1 - x_{m1}}{2\cdot\Delta t}$    B=velocity at the current coarse update
         $C \leftarrow \frac{x_1 - 2\cdot x_0 + x_{m1}}{2\cdot\Delta t^2}$    C=acceleration/2
         $\begin{pmatrix} A + B\cdot t + C\cdot t^2 \\ B + 2\cdot C\cdot t \\ 2\cdot C \end{pmatrix}$    Return position velocity and acceleration
```

Quadratic Interpolation

$t := 0, \delta t .. 0.5$

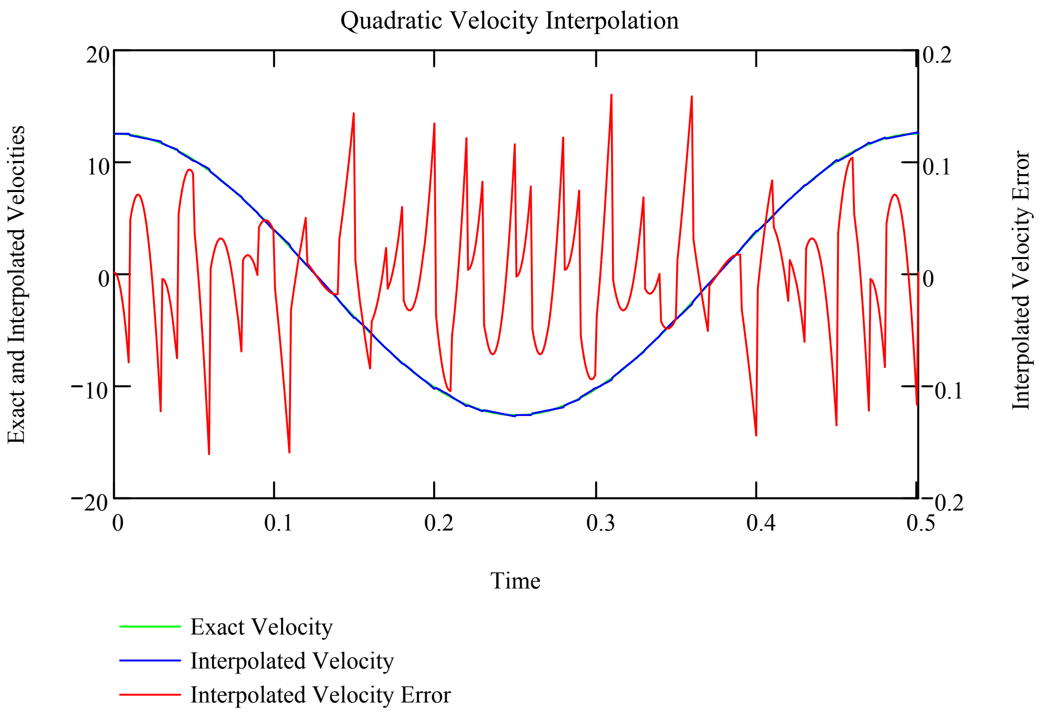


The motion profile looks smooth because the move is over a larger distance but the error is quite large and approaches the resolution of the feed back. The perfect position profile is covered by the interpolated position because the pixel resolution isn't fine enough to show the error. The error increases with the frequency at 4 Hz the error would be twice as big and the error would change up and down twice as fast.

Basically the coarse update time must be fast compared to the rate of change in order to approximate the motion profile more accurately. At higher sine wave frequencies the quadratic interpolation is unusable.

The interpolation errors would be a severe problem in testing applications where sinusoidal motion is often used or following arbitrary higher order curves like spines or cam tables.

Quadratic Interpolation



The interpolation errors become more apparent at the higher derivatives. The maximum velocity error is 0.16046 per second. Multiply this error by 10 volts and divide by the maximum speed to computer the ripple on the control output due to using feed forwards on an imperfect target velocity. In this case the velocity error went down.

$$\frac{0.16046\cdot\frac{\mathrm{in}}{\mathrm{sec}}\cdot 10\cdot\mathrm{volt}}{30\cdot\frac{\mathrm{in}}{\mathrm{sec}}} = 0.053487\,\mathrm{volt}$$

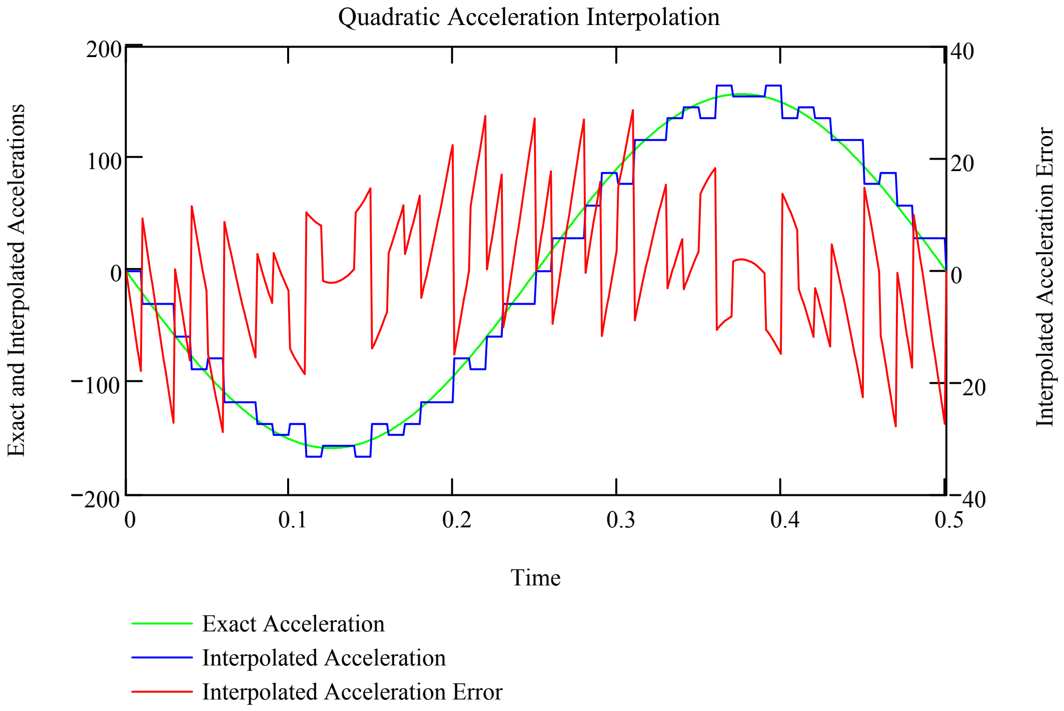
At 2 Hz. Not good.

$$\frac{0.47399\cdot\frac{\mathrm{in}}{\mathrm{sec}}\cdot 10\cdot\mathrm{volt}}{30\cdot\frac{\mathrm{in}}{\mathrm{sec}}} = 0.157997\,\mathrm{volt}$$

At 4 Hz

The velocity interpolation error increases with frequency but at a faster rate than proportionally.

Quadratic Interpolation



One can see that quadratic interpolation for accelerations doesn't work very well. The acceleration changes in steps and they aren't always increasing when they should be. This is due to trying to take the second derivative of quantized data. The control output would look very noisy if acceleration feed forwards are used. The maximum error in interpolating the acceleration is 28.724 in/sec^2

$$\frac{28.724\cdot\frac{\text{in}}{\text{sec}^2}\cdot 10\cdot\text{volt}\cdot 0.1\cdot\text{sec}}{30\cdot\frac{\text{in}}{\text{sec}}} = 0.957467\,\text{volt}$$

at 2 HZ

$$\frac{151.05\cdot\frac{\text{in}}{\text{sec}^2}\cdot 10\cdot\text{volt}\cdot 0.1\cdot\text{sec}}{30\cdot\frac{\text{in}}{\text{sec}}} = 5.035\,\text{volt}$$

at 4 Hz

A 0.957467 volt error can be a problem depending on the system being controlled. 5 volts is way to much noise. It can excite oscillations and cause wear and tear on valves. Normally the user would simply reduce the acceleration feed forward gains or not use them at all resulting in poor performance.

The RMC75 and RMC150 do not normally use interpolation and when it does it uses fifth order interpolation. Most of the time the RMC75 and RMC150 use a fifth order polynomial.
The Delta RMC75 and RMC150 use trig functions such as the sin() and cos() to calculate exact values for position, velocity, acceleration and jerk for sinusoidal motions just as the exact sine wave was generate in this document.

$s(t)=A*\sin(2*\pi*Hz*t)$

$v(t)=A*\cos(2*\pi*Hz*t)*2*\pi*Hz$

$a(t)=-A*\sin(2*\pi*Hz*t)*(2*\pi*Hz)^2$

$j(t)=-A*\cos(2*\pi*Hz*t)*(2*\pi*Hz)^3$

The calculations actually go quickly as terms can be calculated once and reused. Caclulating accurate target velocities is necessary for using the velocity feed forward and derivative gains. Caluclating accurate accelerations is necessary for using the acceleration feed forward and second derivative gains.   Calculating an accurrate jerk is necessary for using the jerk feed forward which is important when controlling hydraulic systems with a low natural frequency and little damping.

Conclusion

It is clear that the inaccuracies of quadratic interpolation can create erroneous control output signals that look like noise on the output. The only way to see if the this noise is truly noise or errors in the interoplator is to set the PID gains to zero and make a point to point move with only the feed forards. The control output should look clean and smooth. If not then a motion controller with a higher target generator may be required.