Smooth Polynomial Ramp for SEI Turnon

Brian Lantz

T1300510-v2, June, 12 2013

1 Summary

This document describes and derives the fifth order polynomial used to turn on the Cartesian basis biases for the displacement sensors on the ISI platforms. These biases are not filter modules, so the standard smooth biases changes are not available. I've derived a simple, smooth ramp to change the biases, and implemented the ramp in a C-code function on the front end. This ramp is called the P5 ramp, since it is based on at fifth order polynomial. It is a good, smooth ramp. The general shape of the ramp is shown below in figure 1. It is likely that more optimal ramps can be derived for particular conditions, but this ramp is well suited for the bias changes because:

1. It is smooth
2. The smoothness leads to minimal high frequency content.
3. It is easy to compute.
4. It is easy to predict various important features such as maximum velocity and peak frequency content.

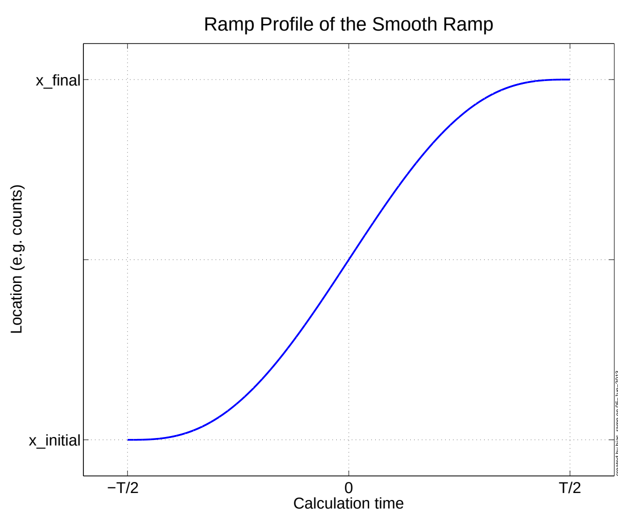

Figure 1: Profile of the P5 smooth ramp.

There is a great deal of discussion in the literature about minimizing the 'jerk' term, or the time derivative of acceleration, for a ramp. However, it seems more useful to try and reduce the spectral overlap of the acceleration (or force) with the response of payload items. We strongly suggest that the ramp time, T, be at least 3 times the longest period of sensitive payloads affected by the ramp. For example, if we are moving the optical table with a ramp, and the table is supporting an optic with a lowest frequency of 0.67 Hz, or a period of 1.5 seconds, the ramp time used be longer than 4.5 seconds. HAM-ISI tables typically use 5 second ramp times.

2 Derivation

The derivation is straightforward. Make a ramp which goes from value x_i to value x_f in time T. To make the ramp 'smooth' we impose the constraints that:

1. The initial velocity, v_i and final velocity, v_f, are zero.
2. The initial acceleration, $\dot{v}_i$, and final acceleration, $\dot{v}_f$, are also zero.

2.1 Velocity Profile Derivation

There are an infinite number of solutions which satisfy the constraints above. Because it is simple, we use a polynomial to define the velocity. We choose for the polynomial to run from time $-T/2$ to $+T/2$ and to be symmetric about $T = 0$. The lowest order polynomial which meets all these conditions will be a fourth order polynomial (which has 3 inflection points) with only even-order terms, so the velocity, v, as a function of calculation time, t_c, will be:

$$v(t_c) = v_4\, {t_c}^4 + v_2\, {t_c}^2 + v_0 \tag{1}$$

The velocity will reach its greatest magnitude at $t_c = 0$ because of the polynomial's order and symmetry. We define this maximum velocity to be v_{max}. Since $v(0) = v_{max}$, it must be true that $v_0 = v_{max}$. We solve for v_2 and v_4 by setting the final velocities and accelerations to be 0.

$$\dot{v}(t_c) = 4\, v_4\, {t_c}^3 + 2\, v_2\, t_c, \text{ at time } T/2 \text{ the acceleration is 0, so} \tag{2}$$

$$0 = 4\, v_4\, (T/2)^3 + 2\, v_2\, T/2, \text{ so} \tag{3}$$

$$0 = v_4\, T^2 + 2\, v_2, \text{ or} \tag{4}$$

$$v_2 = -\frac{v_4\, T^2}{2} \tag{5}$$

Now we look at equation 1 and evaluate it at the end of the ramping time so we can solve for v_2 and v_4.

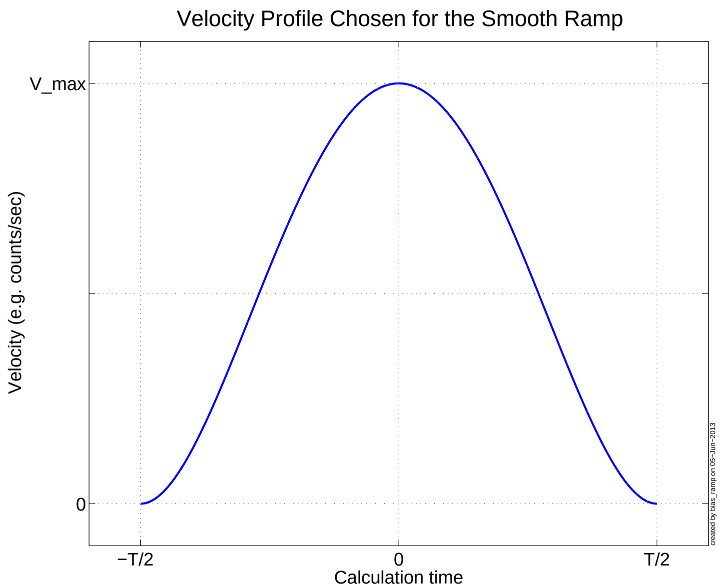

Figure 2: Chosen velocity profile for the smooth ramp.

$$v_4 \, {t_f}^4 + v_2 \, {t_f}^2 + v_{max} = 0 \tag{6}$$

We substitute in our expressions for v_2 and t_f to get

$$v_4 \, \frac{T^4}{16} - \frac{v_4 \, T^2}{2} \cdot \frac{T^2}{4} + v_{max} = 0 \tag{7}$$

which simplifies to

$$v_4 \, \frac{T^4}{16} = v_{max} \tag{8}$$

so

$$v_4 = v_{max} \, \frac{16}{T^4}, \text{ so} \tag{9}$$

$$v_4 = \frac{16}{T^4} \, v_{max}, \text{ and, from equation 5} \tag{10}$$

$$v_2 = -\frac{8}{T^2} \, v_{max}. \tag{11}$$

2.2 Velocity Profile Summary

The velocity is profile is

$$
\begin{gathered}
v(t_c) = v_4\, {t_c}^4 + v_2\, {t_c}^2 + v_0, \text{ where} \\
v_4 = \frac{16}{T^4}\, v_{max},\; v_2 = -\frac{8}{T^2}\, v_{max}, \text{ and } v_0 = v_{max} \\
\text{for } -{}^{T}\!/_{2} < t_c < {}^{T}\!/_{2}.
\end{gathered}
\tag{12}
$$

2.3 Displacement Profile

Once we have a general description for the velocity, we integrate the velocity to get the displacement profile. Integration of equation 12 yields a displacement of:

$$
\begin{gathered}
x(t_c) = \frac{v_4}{5}\, {t_c}^5 + \frac{v_2}{3}\, {t_c}^3 + v_0\, t_c + x_0 \\
\text{for } -{}^{T}\!/_{2} < t_c < {}^{T}\!/_{2}.
\end{gathered}
\tag{13}
$$

We define $\Delta x \equiv x_f - x_i$ and we can evaluate equation 13 as the definite integral

$$
\begin{aligned}
\Delta x &= x(t_f) - x(t_i) \\
&= \frac{v_4}{5}\left(\frac{T^5}{32} - \frac{-T^5}{32}\right) + \frac{v_2}{3}\left(\frac{T^3}{8} - \frac{-T^3}{8}\right) + v_0\left(\frac{T}{2} - \frac{-T}{2}\right) \\
&= \frac{v_4}{5}\,\frac{T^5}{16} + \frac{v_2}{3}\,\frac{T^3}{4} + v_0\, T
\end{aligned}
\tag{14}
$$

substituting in the values of v_4, v_2, and v_0 from equation 12, this becomes

$$
\Delta x = \frac{1}{5}\, v_{max}\, T - \frac{2}{3}\, v_{max}\, T + v_{max}\, T, \text{ or} \tag{15}
$$

$$
v_{max} = \frac{15}{8}\,\frac{\Delta x}{T} \tag{16}
$$

To calculate x_0, we recall that half way through the ramp time, i.e. $t_c = 0$, we are at the midpoint between the initial and final locations, so

$$
x_0 = \frac{x_f + x_i}{2} \tag{17}
$$

2.4 Displacement Profile Summary

We can now express the displace curve, or the ramp, as

$$x(t_c) = x_5\, {t_c}^5 + x_3\, {t_c}^3 + x_1\, t_c + x_0, \text{ where}$$
$$x_5 = \frac{16}{5\,T^4}\, v_{max},\ x_3 = -\frac{8}{3\,T^2}\, v_{max},\ x_1 = v_{max}, \text{ and } x_0 = \frac{x_f + x_i}{2} \tag{18}$$
$$\text{for } -{}^{T}\!/_{2} < t_c < {}^{T}\!/_{2}, \text{ and } v_{max} = \frac{15}{8}\,\frac{\Delta x}{T}.$$

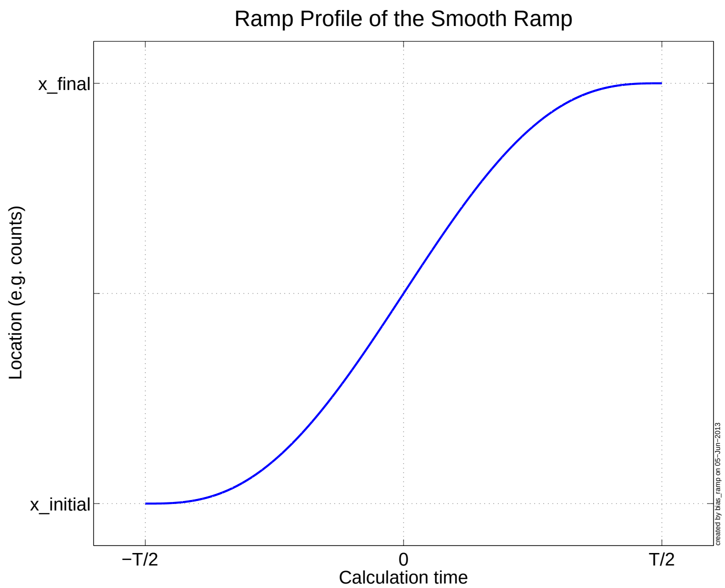

Figure 3: Profile of the P5 smooth ramp.

2.5 Acceleration Profile

It is also useful to consider the acceleration implied by the ramp. The derivative of the velocity given in equation 12 is

$$\dot{v}(t_c) = 4\, v_4\, {t_c}^3 + 2\, v_2\, t_c \tag{19}$$

and for times $-{}^{T}\!/_{2} < t_c < {}^{T}\!/_{2}$ the 'jerk' is

$$\ddot{v}(t_c) = 12\, v_4\, {t_c}^2 + 2\, v_2. \tag{20}$$

There is a discontinuity in the jerk at the beginning and end of the ramp, but this does not seem to matter.

The maximum acceleration occurs at the time $t_{\max}$

$$\begin{aligned} \ddot{v}(t_{\max}) = 12\, v_4\, {t_{\max}}^2 + 2\, v_2 = 0 \\ {t_{\max}}^2 = -\frac{v_2}{6\, v_4} \end{aligned} \tag{21}$$

Solving this with the values of t_2 and t_4 from equation 12 yields

$$t_{\max} = \pm \frac{T}{\sqrt{12}} \tag{22}$$

If we put this time back into the equation 19, we see that the peak acceleration, $a_{\max}$, is

$$\begin{aligned} \dot{v}(t_{\max}) &= 4\, v_4\, \big(\frac{\pm T}{\sqrt{12}}\big)^3 + 2\, v_2\, \big(\frac{\pm T}{\sqrt{12}}\big) \\ a_{\max} &= 4\, \frac{16}{T^4}\, \big(\frac{\pm T}{\sqrt{12}}\big)^3\, v_{max} - 2\, \frac{8}{T^2}\, \big(\frac{\pm T}{\sqrt{12}}\big)\, v_{max} \\ a_{\max} &= \pm \frac{16}{3\,\sqrt{3}}\, \frac{1}{T}\, v_{max}, \text{ or} \\ a_{\max} &= \pm \frac{10}{\sqrt{3}}\, \frac{\Delta x}{T^2} \end{aligned} \tag{23}$$

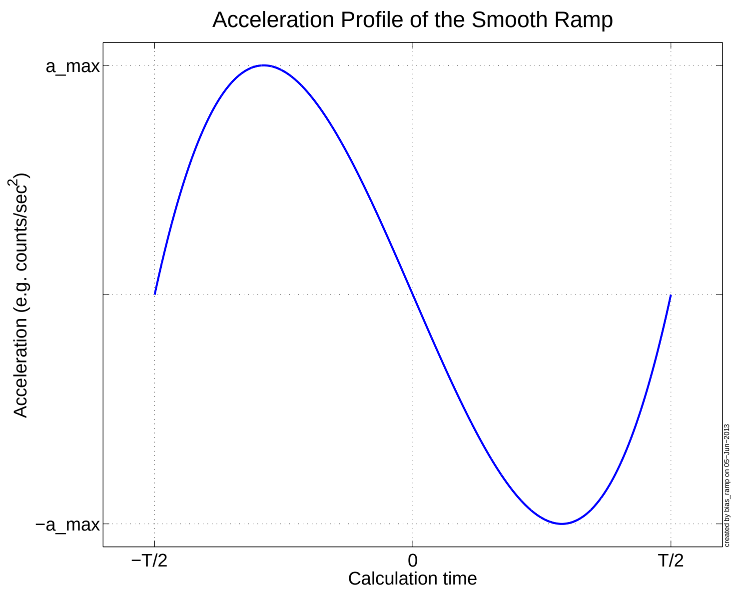

Figure 4: Acceleration Profile of the P5 smooth ramp.

3 C-code

The c-code is located in the {userapps} directory at:

{userapps}/release/isi/common/src/RAMP_BIAS.c

The function which the FE code calls is named RAMP_P5

The code listing is below:

```c
/* RAMP_BIAS.c  Function: RAMP_P5.c
 *
 * This function applies a smooth ramp for bias changes
 * It is a 5th order polynomial.
 *
 * Inputs:
 *
 * (1) double desired_val: value we want to ramp to, or hold
 * (2) double T_ramp:  ramp time in seconds
 *
 * Outputs:
 *
 * (1) double output_val: the current output value
 * (2) double state: 0 = ramping, 1= holding
 *
 * Authors: BTL
 * April 30 - May 2013
 * see T1300510 for a derivation of the ramp - BTL June 12, 2013
 */

#define MODEL_RATE FE_RATE

typedef enum {RAMPING, HOLDING} RampStates;


void RAMP_P5(double *argin, int nargin, double *argout, int nargout){
        static int RampTimer = 0;  // How far along the ramp are we, in cycles
        static int TotalRampCycles = 0;  // Number of cycles in the ramp
        static RampStates CurrentState = HOLDING;
        static int FirstCycle = 1;  // 1 will reinitialize things
        static double PreviousInput;
        static double PreviousOutput;
        static double FinalOutput; // end value for the ramp
        double ThisOutput;
        double Tramp;  // ramptime (sec) read only on new ramp start;
        static double RpC[6];  // these are the polynomial Ramp Coefs.
        double Xdiff;  // Total change for the ramp
        double Vmax;   // max velocity, computed from dX and dT
        double tt;     // time from ramp start, but scaled as -T/2 -> T/2.

        // Start by reading the inputs, we only read the ramp time on input changes.
        double ThisInput = argin[0];

        // on the first cycle, set the output = the input, and end
        if(FirstCycle == 1){
                ThisOutput      = ThisInput;
                FinalOutput     = ThisInput;
                // PreviousInput   = ThisInput;
                // PreviousOutput = ThisOutput;
```

```
            FirstCycle      = 0;
            CurrentState    = HOLDING;
    } else {
            if(ThisInput != PreviousInput){
                    // start a new ramp
                    Tramp       = (double) argin[1];
                    if (Tramp < 0)    {Tramp = 0.0;}
                    if (Tramp > 100) {Tramp = 100.0;}
                    RampTimer = 0;
                    TotalRampCycles = (int) (MODEL_RATE * Tramp);
                    FinalOutput    = ThisInput;
                    PreviousInput = ThisInput;
                    CurrentState   = RAMPING;
                    Xdiff = (double) FinalOutput - PreviousOutput;
                    Vmax = (1.875) * Xdiff/Tramp;
                    // RC are the Ramp Coefficients
                    RpC[0] = PreviousOutput + (0.5 * Xdiff);
                    RpC[1] = Vmax;
                    RpC[2] = 0.0;
                    RpC[3] = (-2.6666666667/(Tramp* Tramp)) * Vmax;
                    RpC[4] = 0.0;
                    RpC[5] = (3.20/(Tramp*Tramp*Tramp*Tramp)) * Vmax;
            }
            switch(CurrentState){
            case RAMPING:
                    RampTimer++;
                    // make this back into a time which goes from -T/2 to +T/2;
                    tt = (double) 2*RampTimer - TotalRampCycles;
                    tt = (0.5 * tt)/(1.0 * MODEL_RATE);  // cast to double before the divide
                    // RC[5]*tt^5 + RC[4]*tt^4 + ... RC[0]
                    ThisOutput = ((((RpC[5]*tt + RpC[4])*tt + RpC[3])*tt + RpC[2])*tt + RpC[1])*tt + RpC[0];
                    if(RampTimer >= TotalRampCycles){
                            ThisOutput    = FinalOutput;
                            CurrentState = HOLDING;
                    } else {
                            CurrentState = RAMPING;
                    }
            break;
            case HOLDING:
                    ThisOutput = FinalOutput;
            break;
            }
    }
    // setup for the next cycle;
    PreviousInput  = ThisInput;
    PreviousOutput = ThisOutput;

    // Set the outputs: Ramp value and current state
    argout[0] = ThisOutput;
    // Output the int value of the current state
    argout[1] = CurrentState;

    // Testing ouputs
    // argout[2] = RpC[0];
    // argout[3] = RpC[1];
    // argout[4] = RpC[2];
```

```
        // argout[5] = RpC[3];
        // argout[6] = RpC[4];
        // argout[7] = RpC[5];
        // argout[8] = tt;
}
```