A practical, low computing cost method for vibration compensation in positioning mechanisms such as those found in CNCs and 3d printers

Milan Cosnefroy, M.Mus.

August 16, 2018

1 Problem definition

1.1 Ringing artifacts

When using 3d printers at high speeds, an artefact known as ”ringing” becomes a major element in low print quality; it is due to vibration of the print head about its intended position relative to the build plate after abrupt accelerations, and shows up as a wavy surface, mainly after right or acute angles in the tool path.
This document describes a method which intends to avoid such artefacts, and brings other advantages.
Previous techniques have been mathematically complex and computationally intensive, requiring pre-processing of the G-Code on a personal computer; this new approach has the huge advantage of bringing almost no overhead, which will allow implementation even on machines controlled by 8-bit microcontrollers.

1.2 Mechanical considerations

1.2.1 Cartesian-type machines

Each axis is assumed to be independant from the others, as is the case with most ”Cartesian” type machines. The algorithm described here is intended for such machines and may or may not help with similar problems in coreXY, delta, SCARA, and other constructions.

1.2.2 Axis modelisation

Effective motion along each axis will be assumed to be analogous to that of a load attached using a spring to a point following the ideal tool path.

1.3 Correction scope

1.3.1 Fixed-acceleration and per-junction Δv motion control scheme

Most consumer 3d printers use a motion control scheme where the velocities at segment extremities are chosen in order not to exceed a certain amount of instantaneous speed difference - often called "jerk", but not related to the third-order derivative of position - and use a fixed-rate acceleration at the beginning and end of each segment, to try and attain the desired print speed in the middle portion if the segment is long enough.
This paper assumes we are working with machines which use this type of motion control.

1.3.2 "Jerk" compensation only

This paper mostly deals with the correction of vibrations induced by "jerk", although a new approach to acceleration will be described further on.

1.3.3 Correction approach

My early empirical approach suggested a solution exists by replacing each segment junction by an additional path segment (and therefore two junctions) of appropriate length and target velocity, i.e. cutting corners; we will now analytically determine said length and velocity.

2 Mathematical analysis

2.1 Variables and parameters

Fixed parameters will be represented using upper-case characters, variables using lower-case. A subscript i indicates component along axis i.

2.1.1 Machine settings

J : "jerk", maximum Δv allowed at the junction between two segments.
A : maximum acceleration allowed.
V : maximum print speed
J_i, A_i, V_i : maximum "jerk", acceleration and speed along axis i

2.1.2 Spring-attached load model

m : load mass
k : spring constant

2.1.3 Speed vectors

$\vec{V_1}$: target speed vector at end of previous segment.
$\vec{V_c}$: target speed vector for added (corrective) segment.
$\vec{V_2}$: target speed vector at beginning of next segment.
$\Delta\vec{V}$: speed vector difference between previous and next segment.
$\Delta\vec{V_1}$: speed vector difference between previous and corrective segment.
$\Delta\vec{V_2}$: speed vector difference between corrective and next segment.

2.1.4 Motion relative to point along corrective segment

$\vec{x}(t), \vec{v}(t), \vec{a}(t) :$ position, speed and acceleration relative to a point following the planned path.
$t :$ time elapsed since start of corrective segment.
$T :$ time at end of corrective segment.

2.2 Differential equation

2.2.1 General solution form

$$m\vec{a}_i + k\vec{x}_i = 0$$

By defining $K = \frac{k}{m}$, we can rewrite the equation as:

$$\vec{a}_i + K_i\vec{x}_i = 0$$

Which solves to:

$$x_i(t) = c_1 \cdot cos(\sqrt{K_i} \cdot t) + c_2 \cdot sin(\sqrt{K_i} \cdot t)$$

From which we can derive

$$v_i(t) = \sqrt{K_i}(-c_1 \cdot sin(\sqrt{K_i} \cdot t) + c_2 \cdot cos(\sqrt{K_i} \cdot t)$$

2.2.2 Constant determination using initial conditions

$$x_i(0) = c_1 \cdot cos(0) + c_2 \cdot sin(0) = c1$$

At $t = 0$, there is by definition no deviation from the path yet; therefore:

$$c_1 = 0$$

Let's move on to c_2, using $v_i(t)$:

$$v_i(0) = \sqrt{K_i} \cdot c_2 \cdot cos(\sqrt{K_i} \cdot 0)$$

$$v_i(0) = \sqrt{K_i} \cdot c_2$$

$$c_2 = \frac{v_i(0)}{\sqrt{K_i}}$$

$$v_i(0) = -\Delta\vec{V}_{1,i}$$

$$c_2 = -\frac{\Delta\vec{V}_{1,i}}{\sqrt{K_i}}$$

Therefore:

$$x_i(t) = -\frac{\Delta\vec{V}_{1,i}}{\sqrt{K_i}} \cdot sin(\sqrt{K_i} \cdot t)$$

$$v_i(t) = -\Delta\vec{V}_{1,i} \cdot cos(\sqrt{K_i} \cdot t)$$

2.2.3 Full solution using conditions at $t = T$

We want the effective path to join with the ideal path where the corrective segment joins the next tool path segment; therefore:

$$x_i(T_i) = 0$$

$$sin(\sqrt{K_i} \cdot T_i) = 0$$

$$\sqrt{K_i} \cdot T_i = 0 \mod \pi$$

We also want the effective speed to be equal to the target speed when beginning the next segment.

$$v_i(T_i) = \Delta V_{2,i}$$

$$v_i(T_i) = \Delta V_i - \Delta V_{1,i}$$

$$-\Delta V_{1,i} \cdot cos(\sqrt{K_i} \cdot T_i) = \Delta V_i - \Delta V_{1,i}$$

$$\Delta V_{1,i} \cdot (1 - cos(\sqrt{K_i} \cdot T_i)) = \Delta V_i$$

$$\Delta V_{1,i} = \frac{\Delta V_i}{1 - cos(\sqrt{K_i} \cdot T_i)}$$

We already know that $\sqrt{K_i} \cdot T_i = 0 \mod \pi$;
However $\Delta V_{1,i}$ is not defined for $\sqrt{K_i} \cdot T_i = 0 \mod 2\pi$, which leaves us with:

$$\sqrt{K_i} \cdot T_i = \pi \mod 2\pi$$

We are not interested in letting the tool head oscillate before settling down; therefore, we will use the first possible value:

$$\sqrt{K_i} \cdot T_i = \pi$$

$$(T_i = \frac{\pi}{\sqrt{K_i}})$$

Which leads us to:

$$\Delta V_{1,i} = \frac{\Delta V_i}{1 - cos(\sqrt{K_i} \cdot T_i)} = \frac{\Delta V_i}{2}$$

$$\Delta V_{2,i} = \Delta V_i - \Delta V_{1,i} = \frac{\Delta V_i}{2}$$

$$\Delta V_{1,i} = \Delta V_{2,i} = \frac{\Delta V_i}{2}$$

$$V_{c,i} = \frac{V_{1,i} + V_{2,i}}{2}$$

We

3 "Jerk" compensation algorithm

4 An alternative to traditional firmware acceleration