Layer to Layer Registration of a Slurry-Based 3D Printing Machine

By

Kristopher J. Seluga

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

SCIENCE BACHELORS IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

MAY 2000
(June 2000)

Signature of Author: _____

Department of Mechanical Engineering
May 4, 2000

Certified by: _____

Emanuel Sachs
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by: _____

Ernest Cravalho
Professor of Mechanical Engineering
Chairman, Undergraduate Thesis Committee

Layer to Layer Registration of a Slurry-Based 3D Printing Machine

By

Kristopher J. Seluga

Submitted to the Department of Mechanical Engineering
on May 4, 2000 in Partial Fulfillment of the
Requirements for the Degree of Science Bachelors in
Mechanical Engineering

ABSRACT

Slurry-based 3D printing is a new technology designed to further increase the feature resolution capabilities of 3D printing. This technology deposits an unprinted wet slurry layer by rapidly depositing overlapping parallel lines of slurry to form each layer. Line registration may affect the final properties of the part and it is therefore desirable to control the registration of slurry lines within each layer and from one layer to the next. A galvanometer was used to adjust nozzle position to compensate for line position errors. The galvo correction was calculated and actuated using a PMAC controller running programmable logic controller programs.

The line spacing controller is successful in improving line spacing and registration over open loop performance. However, the degree of line position improvement is not certain due to measurement errors. Because of the large measurement variability, the line spacing may actually be better than measured. Developing more accurate diagnostics is important to determine the accuracy of the line position controller. More importantly, the effect of different line registration patterns on layer and part characteristics should be investigated.

Thesis Supervisor: Emanuel Sachs

Title: Professor of Mechanical Engineering

Acknowledgements

The author would like to thank Ely Sachs, Ben Polito, David Ables, Christopher Stratton and the rest of the 3D printing staff for their assistance in the development of the control software and the writing of this thesis.

TABLE OF CONTENTS

Abstract 2
Acknowledgements 3
Table of Contents 4
Variable Definitions 5
1.0 INTRODUCTION 7
1.1 Process Background 7
1.2 Machine Description 7
1.3 Line Registration Problem 10
2.0 PROGRAMMING IMPLEMENTATION OF MACHINE CONTROL 13
2.1 Slow Axis 14
2.2 Fast Axis 14
2.2.1 Counter Mass Motor 14
2.2.2 Carriage Motor 16
2.3 Galvanometer 17
3.0 PREDICTED MACHINE PERFORMANCE 23
3.1 Slow Axis 23
3.2 Fast Axis 23
3.3 Line Spacing Predictions 26
4.0 MEASURED MACHINE PERFORMANCE 30
4.1 Slow Axis 30
4.2 Fast Axis 30
4.3 Line Spacing Results 33
4.3.1 Line Position Measurement Technique 34
4.3.2 Results 34
5.0 CONCLUSIONS AND FURTHER WORK 40
Appendix A PLC Program Code 41
Appendix B Variable Initialization Program 50
Appendix C Slow Axis Motion Programs 51

For the following figures, equations and programs refer to this variable definition list.

Variable name ;variable description

Machine geometry

B	distance before beginning of line where galvo correction calc is made (m)
b	bed width (m)
d	distance from spring to edge of bed (m)
L	galvo arm length (m)
noz_off	nozzle offset from carriage center

Slow axis:

Vss	commanded instantaneous slow axis velocity (m/sec)
Vsa	actual instantaneous slow axis velocity (m/sec)
Vsa_ave	actual slow axis velocity averaged over the period of carriage motion (m/sec)
Xs	actual instantaneous position of slow axis (m)
s	commanded line spacing (m)
n0	positive carriage direction pass line counting variable
n1	negative carriage direction pass line counting variable
Xs_i	initial specified slow axis position where first line deposition begins
Xs_f	final commanded slow axis position where final line deposition ends
offset	commanded layer to layer line registration offset (0.5 = 180° phase)
delta_x	calculated line position error for galvo correction calculation (m)
Xs_first_0	recorded position of first positive line for line registration calculation (m)
Xs_first_1	recorded position of first negative line for line registration calculation (m)
Xs_exp	expected line position based on first position and line number (m)
Xs_last_0	position of slow axis at start of last positive line (m)
Xs_last_1	position of slow axis at start of last negative line (m)

Fast axis:

f	measured or calculated (from lookup table) frequency of carriage
Yfa	actual instantaneous fast axis position (m)
Vfa	actual instantaneous fast axis velocity (m/sec)
ta_f_0	recorded time elapsed for last positive pass over bed (sec)
ta_f_1	recorded time elapsed for last negative pass over bed (sec)
Vf_ave_0	calculated average velocity over bed on last positive pass (m/sec)
Vf_ave_1	calculated average velocity over bed on last negative pass (m/sec)
Vf_ave_s	commanded average velocity over bed (m/sec)
n	counting variable for the frequency measurement
ta_f	timing variable for frequency measurement
percent	velocity tolerance for open loop voltage searching algorithm (percentage)

counter mass motor:

K1	controller gain for fast axis counter mass closed loop control
out_pos	positive fast axis counter mass open loop driving voltage (program units)
out_neg	negative fast axis counter mass open loop driving voltage (program units)
out_pos_set	counter mass open loop driving voltage calculated from empirical formula

carriage motor:

K2	controller gain for fast axis carriage motor closed loop control
out_car	carriage motor driving voltage (program units)
Kp	carriage position controller proportional gain
Kd	carriage position controller derivitive gain
Yfs	commanded carriage position (m)
Vf_run	commanded carriage velocity while over bed (m/sec)
N	counting variable for bed being laid
pos_limit	position controller output limit (prog units)

Galvo:

theta	uncorrected galvo angular position (rad)
theta_deg	uncorrected galvo angular position (deg)
delta_theta	total calculated angular galvo correction (rad)

Other:

x	determines direction of carriage pass (0 for + carriage velocity, 1 for negative)
mark	position where galvo correction is made (dist. B from edge of slurry bed)
Ynoz	nozzle position (m)
te	expected travel time between lines (sec)
ta	measured travel time between lines (sec)
theta_conversion	PMAC conversion between radians, program units and output voltage
counter_0_elapsed	elapsed time on counter 0 (sec)
counter_1_elapsed	elapsed time on counter 1 (sec)
counter_2_elapsed	elapsed time on counter 2 (sec)
counter_3_elapsed	elapsed time on counter 3 (sec)

DAC3	galvo output DAC3
DAC4	fast axis counter mass output DAC4
DAC2	fast axis carriage motor output DAC2

INTRODUCTION & PROCESS BACKGROUND

Slurry-based 3D printing is a new technology designed to further increase the feature resolution capabilities of 3D printing. To accomplish this, the particle size of the powder material must be reduced to approximately 1.0 micron. At this particle size, the conventional method of using a roller to provide the flat unprinted layer of powder fails. Electrostatic forces between powder particles and small currents in the air make it difficult to maintain a smooth powder layer. The most promising solution to this problem is to deposit the unprinted layer in the form of wet slurry that will not be subject to the same problems.

The slurry-based 3D printing machine is designed to provide just such a layer of unprinted slurry. It accomplishes this by depositing many parallel lines of slurry rapidly to form each layer. After the layer is completed, it is moved to another machine to be printed and then returns to the slurry machine so the next slurry layer can be deposited.

Machine description

The slurry-based 3D printing machine is composed of two perpendicular axes. The fast axis holds a carriage that passes a nozzle over a bed and deposits lines of slurry. The bed is the slow axis that moves perpendicular to the nozzle motion advancing the lines of deposited material. Together, the motion of these two axes determines the characteristics of the final layer. The following figure is a picture of the machine from above. It shows the layout of the fast axis with linear motors and the slow axis.

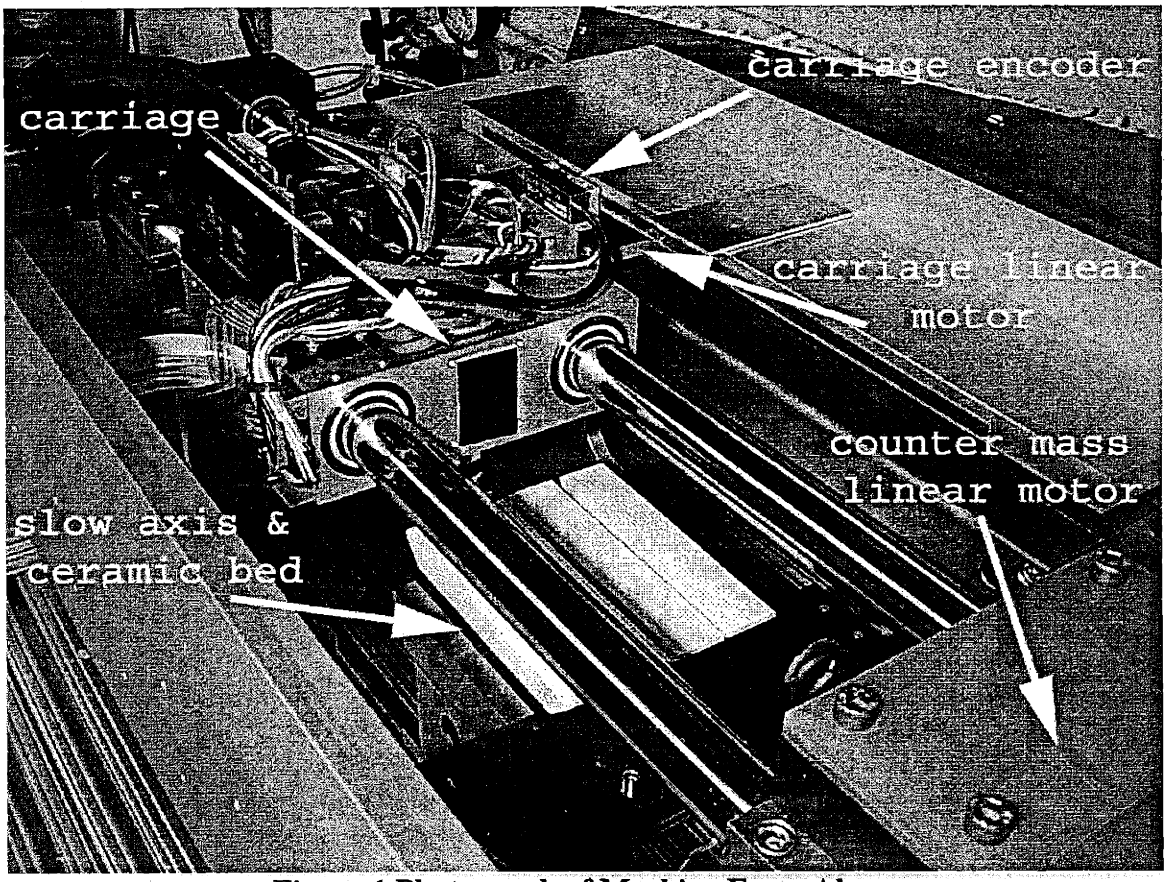


Figure 1 Photograph of Machine From Above

The slow axis is a standard ball screw linear actuator purchased off the shelf. The fast axis actuation is more complicated. The carriage (which contains some fluid system elements, the galvo electronics, galvo and nozzle) rides on air bearings on two rails. The rails themselves ride on air bearings that are attached to the base of the machine. These railings and the attached metal blocks make up the counter mass. The counter mass is several times more massive than the carriage and has a small range of motion. The carriage and counter mass move in a reciprocal motion and during each collision energy is transferred between the counter mass and the carriage to maintain the desired carriage velocity. A schematic of the carriage and counter mass arrangement is shown in figure 2.

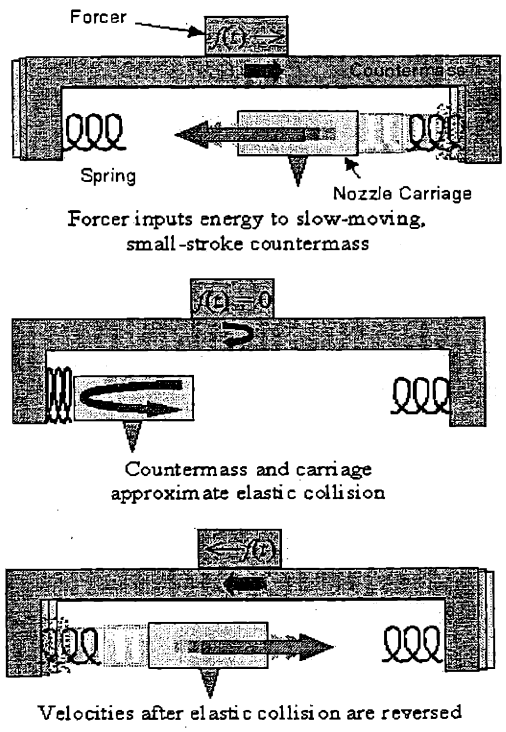
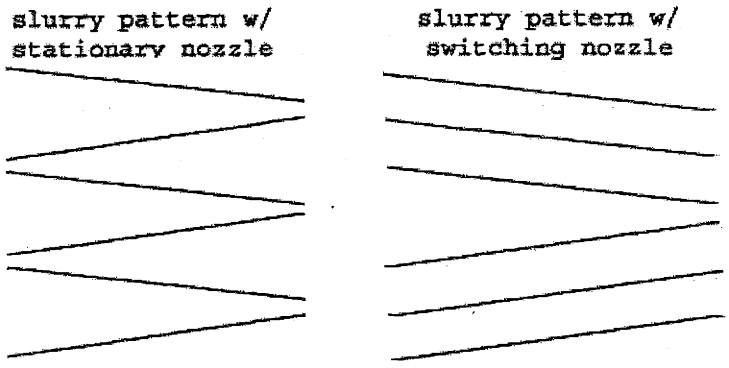


Figure 2 Fast Axis Schematic of Relative Motion Provided by David Ables of MIT 3DP Lab

This configuration serves to eliminate machine vibrations that would otherwise occur at high carriage frequencies. The counter mass and carriage are each driven by a linear motor. The counter mass linear motor serves as the main power source while the carriage motor is used to control the carriage velocity during each traverse.

Because the slurry is deposited continuously and the carriage travels in two directions, the pattern created by the nozzle passing over the bed would result in a series of zigzag lines. This is not desirable because crossing lines would result in a rough layer. To overcome this problem, the machine deposits a set of two beds simultaneously by switching the nozzle position. All the lines in each bed are parallel to each other but the different beds' lines are at an angle to one another as shown in figure 3.



(not to scale- angles are exaggerated)
Figure 3

The switching of the nozzle position is accomplished by mounting the nozzle on a small (5.3 cm) arm actuated by a galvanometer as shown in figure 4. The galvo used has a range of motion of ±15° about its neutral position, which is parallel to the carriage motion.

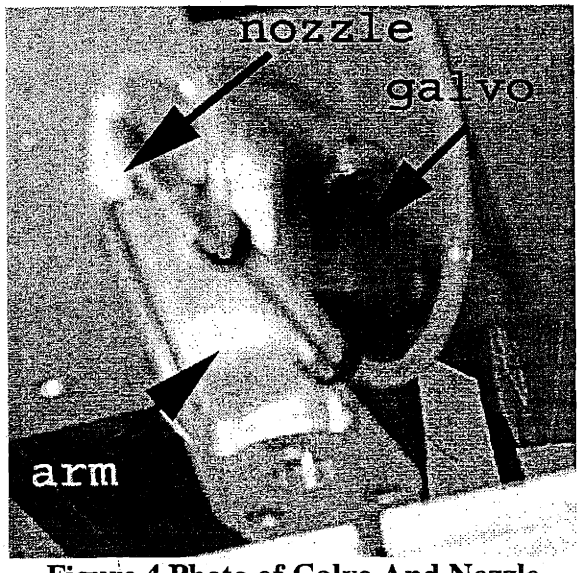


Figure 4 Photo of Galvo And Nozzle

Line registration problem

The first concept proof model of the ceramic slurry-based 3D printing machine did not have the ability to control the registration of slurry lines. Line registration is the relative position, or phase between the line positions from one layer to the next. This can be measured relative to

another layer or to the absolute position scale of the slow axis. Line registration may affect the properties of the final part including density, surface finish, and local thickness. It is therefore desirable to control the registration of slurry lines within each layer and from one layer to the next. Figure 5 shows two possible registration patterns as seen from the side of the part.

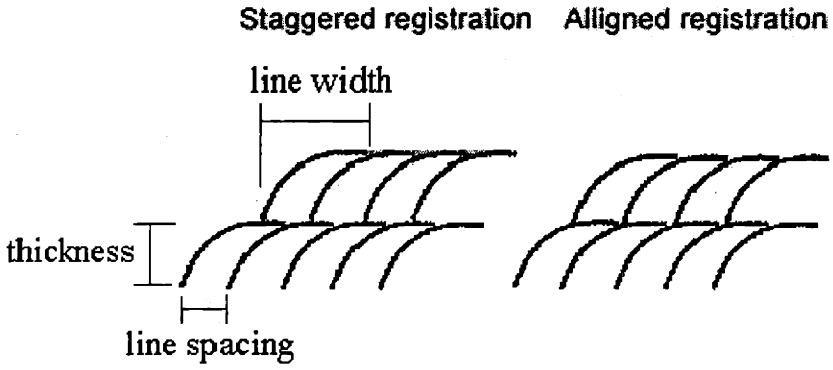


Figure 5 Two Distinct Line Registration Patterns

My task was to assist in the acquisition of motion control hardware and to develop the control algorithm that coordinates these two axes to enable the machine to control line registration.

Carriage speed and frequency, cross speed (bed speed) and nozzle position relative to carriage are the three machine variables that must be coordinated to control line registration. Carriage speed and position and bed speed and position are known through the use of linear encoders and are available as inputs to the Delta Tau PMAC controller. With this information, determining the position error is a matter of geometry. A schematic of the situation is shown in figure 6.

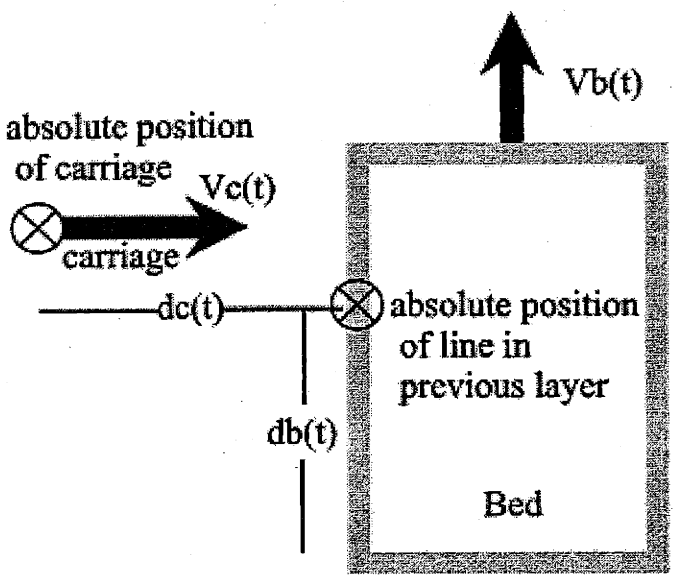


Figure 6 Schematic of Bed and Carriage Positions and Motions

$V_c(t)$ and $d_c(t)$ are the carriage velocity and position respectively and are known from the carriage encoder. $V_b(t)$ and $d_b(t)$ are the bed velocity and position respectively which are to be measured and controlled by the proposed system. $V_c(t)$ is on the order of 1-3.5 m/sec and $V_b(t)$ is no more than 5 mm/sec. Because line spacing is approximately 100μm, the accuracy of $d_b(t)$ (line position) must be on the order of 10μm to ensure good control of line registration.

It was first proposed to vary the speed of the slurry bed to compensate for errors in line position. The controller requirements for this method proved too difficult for conventional controllers and actuators. Instead, it was decided that the galvo could be used to adjust the nozzle position slightly before each pass to compensate for any error in line position. This control scheme was used for all subsequent analysis and implementation.

PROGRAMMING IMPLEMENTATION OF MACHINE CONTROL

The PMAC controller supports two main types of programs, motion programs and programmable logic controllers (plc programs). PMAC motion programs are written in a combination of G-code and Basic. When called, a motion program runs through all the lines of code once and then closes itself. These programs are not useful to controlling the repetitive action of the fast axis because they only run once. Programmable logic controllers are better suited to this application because they run on a continuous loop while they are active. The following table lists each PLC program and its function.

Prog	Function
1	Measure carriage frequency based on carriage encoder signal. Averaged frequency over last 25 passes to reduce variability.
2	Convert encoder position and velocity variables to SI units.
3	Timer 3 (to be called by any programs that require timing)
4	Timer 0 (to be called by any programs that require timing)
5	Timer 1 (to be called by any programs that require timing)
6	Assign a value to variable x based on the direction of carriage velocity. For Vfa > 0, x=0 and when Vfa < 0, x=1.
7	Actuate galvo to home position before galvo correction. Trigger the output on carriage turn around (represented by a change in the value of x).
8	Calculate required galvo correction based on carriage and bed position. Actuate galvo to correction position slightly before nozzle passes edge of bed.
9	Actuate fast axis counter mass in both directions using a proportional negative feedback control scheme. Output is based on error between commanded and measured average carriage velocity.
10	Measure and store carriage average velocity over bed in each direction.
11	Timer 2 (to be called by any programs that require timing)
12	Seek open loop counter mass driving voltage that yields specified velocity. This is no longer required once an empirical formula for required voltage is found and used.
13	Close control loop on counter mass controller by setting a non-zero gain once open loop velocity is satisfactory for a specified number of passes. This prevents crashing during carriage startup and velocity changes.
14	Activate PLC 12 to find a new open loop driving voltage when the commanded velocity changes.
15	Actuate carriage motor while the nozzle is over the bed. Controller action is proportional to error in instantaneous velocity. This controller counteracts drag forces on the carriage.
16	Actuate carriage motor with open loop driver for starting and breaking (negative gain for breaking). A simple switching output forces the carriage to cross the bed at low frequency until the reciprocating motion is developed and the counter mass can begin driving the carriage.

17	Carriage proportional, derivative position controller. Simple PD control used to move carriage to rest position between layers.
18	Carriage centering program for machine startup. Uses carriage motor to traverse the fast axis and calculate the position centered between the springs.
19	Trigger galvo switching on slow axis position. When bed is in position, move galvo from rest position to switching positions with line spacing controller on. Also trigger first line to prevent switching over bed causing stray lines.
20	Automated fast axis startup by initializing certain variables and calling on other PLC's.
21	Automated fast axis stop by initializing certain variables and calling on other PLC's.

The PMAC code to these plc programs is listed in appendix A

Slow axis

The slow axis supports the moving slurry bed and is not used for line position control. Instead, its velocity is set to an appropriate value for the build and commanded to that constant value for the duration of the build. The commanded bed velocity is calculated by the controller, based on fast axis frequency and desired line spacing using the following equation.

$$V_{ss} = s \cdot f \qquad \text{Equation 1}$$

In the implementation, the frequency value is calculated as a function of carriage commanded average velocity from an empirically determined equation. This calculation sets a slow axis velocity that will result in an open loop line spacing close to the desired spacing. This assures that the line spacing error is small enough to be eliminated by the galvo correction.

Fast axis

Counter mass motor

The counter mass motor provides the main power for the carriage motion. The control algorithm for this motor must maintain a specified carriage velocity as the nozzle passes over the bed. The implemented controller is a proportional controller operating about a set point. Because of minor asymmetries in the machine, the control feedback signal is divided into two signals, one for each

direction. The variable "x" identifies the direction of the carriage and is defined as 0 for positive velocity and 1 for negative velocity.

An empirical formula is used to calculate the value of nominal counter mass motor power output that will yield a stable open loop carriage velocity close to the specified carriage velocity. This set output value is then applied to counter mass motor in the opposite direction to the carriage velocity. The controller then adds a correction to the set output that is proportional to the difference between the set carriage velocity and the average carriage velocity measured on the last pass in the same direction. This is the main mechanism for the control of the nozzle velocity.

This algorithm is also described in the following block diagram, figure 7.

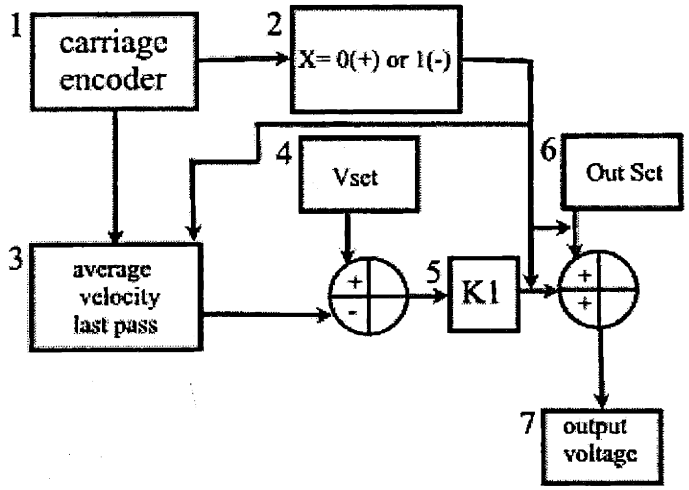


Figure 7 Block Diagram of Counter Mass Motor Control Algorithm

1. The carriage encoder provides position information, which is used to interpolate velocity by the PMAC controller.
2. PLC 6 uses the carriage velocity variable to determine the value of x for use by other calculations.

3. PLC 10 uses the encoder signal to calculate the average nozzle velocity over the bed for each direction corresponding to x=0 and x=1.
4. The commanded nozzle/carriage velocity is compared to the measured value to determine velocity error
5. K1 is the proportional gain of the velocity controller.
6. The predetermined (PLC 12) open loop counter mass output voltage is then added to the proportional correction.
7. The sum is then calculated once for each pass (corrected using x to assure the proper sign) and is output to the counter mass motor until the value of x changes.

Carriage motor

Because the counter mass is only in contact with the carriage during collisions, a smaller separate linear motor is attached directly to the carriage. This motor is used to start and stop the fast axis motion. It is also used to compensate for friction during the carriage's traverse over the bed. A simple proportional controller is employed while the nozzle is over the bed. During layer formation, the carriage motor output is commanded to be proportional to the difference between instantaneous velocity and desired velocity in the same direction as carriage velocity.

This algorithm is also described in the following block diagram, figure 8.

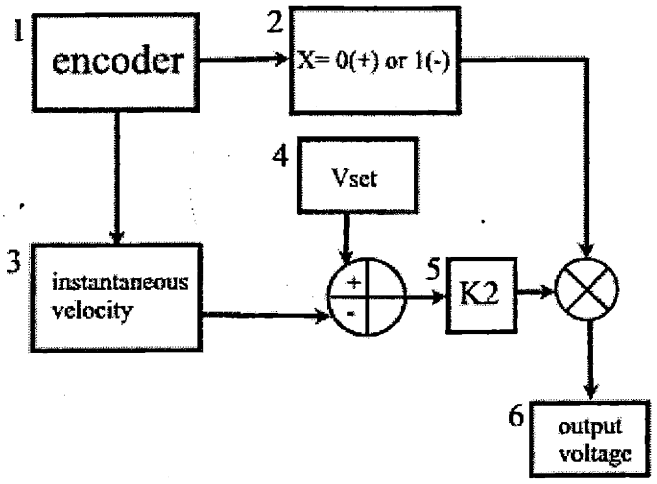


Figure 8 Block Diagram of Carriage Motor Control Algorithm

1. The carriage encoder provides position information, which is used to interpolate velocity by the PMAC controller.
2. PLC 6 uses the carriage velocity variable to determine the value of x for use by other calculations.
3. The PMAC controller calculates the carriage instantaneous velocity based on the encoder signal.
4. While the nozzle is over the bed, the measured velocity is continuously compared to the commanded velocity.
5. This velocity error is multiplied by the gain K2.
6. This product is given the correct sign based on x and then output to the carriage motor.

Galvo

The galvo control algorithm controls two actions. First, it must switch the nozzle position between the two beds before the beginning of each line. This only requires a simple algorithm that switches the galvo position between two positions based on the direction of the carriage velocity. The galvo must also maintain line registration by correcting the nozzle position before the beginning of each line. The program must take as inputs: desired line spacing, nozzle velocity, slow axis position and desired offset/relative registration between layers.

The line registration is based on the absolute position of the bed at the start of each line. At the start of the first layer, the position of the first line is recorded to program memory. This location is then used to calculate the required position of each individual line. Before each line begins, its actual position is compared to the commanded position. This position error is then converted into a small angular correction for the galvo. The details of these calculations are explained below.

The program must first take the desired line spacing and nozzle velocity as inputs. These variables will determine the characteristics of the layer and must be specified by the user. The

carriage frequency associated with this velocity is then calculated based on an empirical polynomial that is fit to the measured data as shown in figure 9.

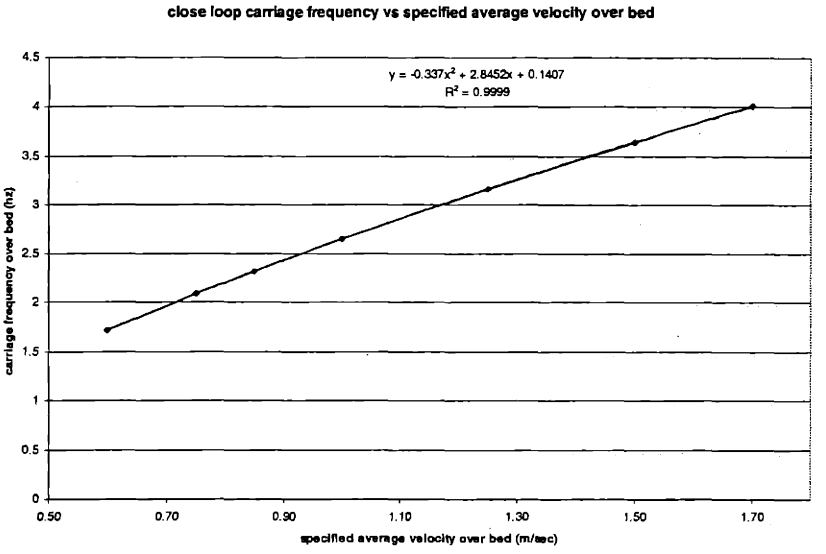


Figure 9 Measured Carriage Frequency As A Function of Average Carriage Velocity

For the current machine configuration, the velocity frequency relationship was found to fit the following equation.

$$f = \left(-0.337 \cdot V_{f\,aves}^2 + 2.8452 \cdot V_{f\,aves} + 0.1407\right) \qquad \text{Equation 2}$$

In theory, this frequency velocity relationship should an equation of the following form.

$$F = \frac{1}{C_1 + \frac{C_2}{V}} \qquad \text{Equation 3}$$

When this equation is fit to the same data as in figure 9 the constants C1 and C2 are found to be C1 = 0.0034 sec and C2 = 0.4 m. This yields a fairly good curve fit (figure 10) showing that this model is close to the observed behavior but is not as good a fit as the polynomial fit. For this reason, the carriage program uses the polynomial equation to predict carriage frequency.

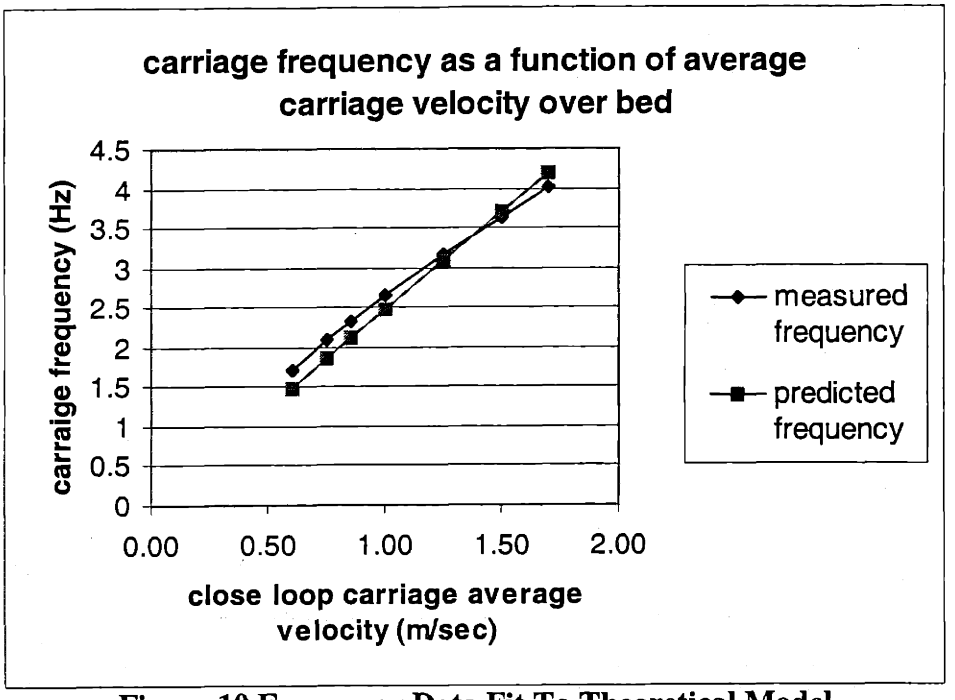


Figure 10 Frequency Data Fit To Theoretical Model

Based on the known line spacing and carriage frequency, the required bed (slow axis) velocity can be represented as the product of frequency and spacing.

$$V_{ss} = s \cdot f \qquad \text{Equation 4}$$

Maintaining this bed velocity will result in an open loop spacing close enough to the desired value that the galvo controller can compensate for the remaining error. During each pass, the galvo controller performs its calculations and actuates the galvo at some small distance B from the edge of the bed referred to in the code as the "mark." The small distance B is necessary to assure that the controller has time to perform the necessary calculation and actuate the galvo before the nozzle begins passing over the bed. Based on an expected nozzle velocity of 1-2 m/sec and the measured galvo settling time, B was chosen to be 5.0 mm. This translates to an available time of 2.5 msec, which is long enough for the calculation and actuation to take place.

When the nozzle passes the mark the following calculations are made:

The expected bed position is calculated based on the position of the first line, the desired spacing and the current line number using equation 5. This calculation is made separately for each carriage direction.

$$X_{s\exp} = X_{s\,first\,0} - n_0 \cdot s \qquad \text{Equation 5}$$

Equation 6 then calculates the predicted line position error. The additional offset term determines the commanded layer-to-layer line registration offset where the offset is represented as a percentage 0-100%.

$$\Delta x = X_s - X_{s\exp} - offset \cdot s \qquad \text{Equation 6}$$

This line position error can then be converted to a galvo angular displacement through a geometric relationship as shown in figure 11 and equation 7.

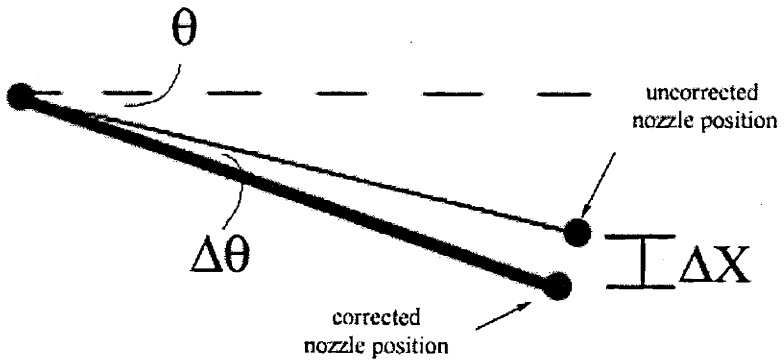


Figure 11 Geometric Relationship Between Galvo Angle and Line Position

$$\Delta\Theta = a\sin\left(\sin\Theta - \frac{\Delta x}{L}\right) - \Theta \qquad \text{Equation 7}$$

This equation assumes that the cosine term in the nozzle displacement is negligible which is true as long as delta theta << theta. The line position error due to the cosine term is represented by equation 8. Based on machine geometry and a given line position error, this relationship yields a cosine term galvo correction three to four orders of magnitude smaller than the sine term correction. This is why the cosine term can be neglected.

$$\Delta\Theta_{\cos} = a\cos\left(\cos\Theta - \frac{\Delta x \cdot V_{fa}}{L \cdot V_{sa}}\right) - \Theta \qquad \text{Equation 8}$$

The full form of this relationship could have been used, but debugging issues made the simpler form more attractive. The current line position controller could easily be rewritten to include the full form though the resulting error reduction may not be significant.

The line position control algorithm is also described in the following block diagram.

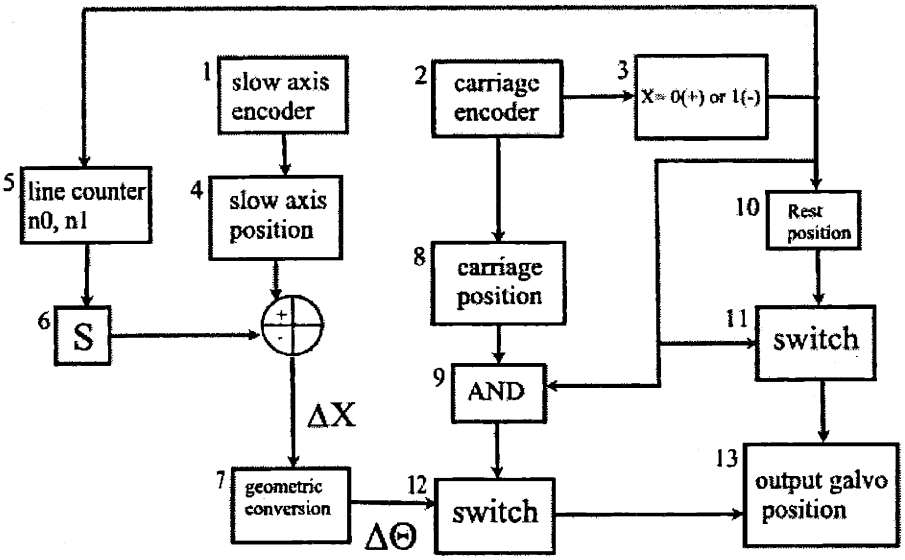


Figure 12 Block Diagram of Line Position Controller

1. The slow axis encoder provides slow axis position data and can be used by the PMAC to determine slow axis velocity.
2. The carriage encoder provides carriage position and velocity data as before.
3. This controller also uses the variable x.
4. Average nozzle velocity over bed is also required.
5. A line counting function determines the variables n0 and n1 based on the switching of the x variable.
6. The specified line spacing is then multiplied by the line number to determine the commanded line position in an absolute position scale. When this is subtracted from the actual slow axis position, it yields the line position error ΔX.
7. Based on figure 11 and equation 7 the position error ΔX is converted into an angular correction $\Delta\Theta$.

8. The carriage position is compared to the position of the edge of the bed to determine galvo position.
9. When the absolute value of the nozzle position and x are in a certain condition (x=0 and Yn>mark; or x=1 and Yn<mark) the galvo is actuated to its corrected angular position.
10. The galvo rest position Θ is determined by the geometric constraints of the bed.
11. When the x variable changes values at carriage turn around, the galvo is commanded to the rest position. This brings the galvo close to the corrected position so that the correction can later be actuated in a short time.
12. When the nozzle approaches the edge of the bed (as determined by block 9) the corrected angular position is output to the galvo until the carriage position switches causing the galvo to return to its rest position.

PREDICTED MACHINE PERFORMANCE

In order to better understand the machine system and to effectively control it, analytical models of the machine components were developed. The slow axis performance data was provided by the manufacturer. The carriage and line position controllers were modeled based on machine geometry and expected velocity errors using spreadsheets. The methods and results of these simulations are explained below.

Slow axis

The expected slow axis velocity performance provided by the manufacturer is ±10% instantaneous velocity (at 20 Hz bandwidth) with a position repeatability of ±5.0 μm.

Fast axis

The discrete nature of the fast axis design makes the system difficult to model with time dependent transfer functions. In order to understand the stability problem observed with the fast axis controller, a table of sample carriage passes was calculated for varying gains. This is a quantitative model of the carriage dynamics ignoring the effects of the carriage motor (which is assumed smaller than the counter mass controller effects). The simulation takes as inputs, specified carriage velocity, counter mass controller gain, collision distance constant found from Figure 10 and carriage mass. The counter mass motor output is then calculated just as in the actual algorithm using equation 9.

$$F_{motor} = 0.0106 \cdot output_{program\_units} + 0.0981 \qquad \text{Equation 9}$$

Once the motor force output is known, the conservation of total system energy before and after each collision is used to predict the carriage velocity as shown in following equations.

$$E_{out} = E_{in} - E_{loss} + E_{controller} + E_{nominal} \qquad \text{Equation 10}$$

The terms E_{in} and E_{out} represent the kinetic energy of the system before and after the collision. Because the counter mass's velocity is unknown, is smaller than the carriage velocity and should be approximately equal and opposite before and after the collision, it is neglected. E_{in} and E_{out} then represent the carriage kinetic energy as in equation 11.

$$E_{carraige} = \frac{1}{2} \cdot m \cdot v^2_{carriage} \qquad \text{Equation 11}$$

Since the nominal (open loop) energy is sufficient to compensate for drag and collision losses, the average energy loss can be found with equation 12.

$$E_{loss} = E_{nominal} = F_{nominal} \cdot d_{collosion} \qquad \text{Equation 12}$$

When these equations are combined they yield equation 13, an expression for carriage velocity after each collision.

$$v_{out} = \sqrt{v^2_{in} - \frac{2*d}{m} \cdot F_{nominal} + \frac{2*d}{m} \cdot \left(F_{controller} + F_{nominal}\right)} \qquad \text{Equation 13}$$

In order for the simulation to model stability, an element of random variation is added to the loss term in the model as shown in equation 14.

$$v_{out} = \sqrt{\left(1 - \%_{loss}\right) \cdot v^2_{in} + \frac{2*d}{m} \cdot \left(F_{controller} + F_{nominal}\right)} \qquad \text{Equation 14}$$

To determine the proper amount of energy loss variation ($\%^{loss}$) to add to the model, the gain is set to zero and energy loss term is adjusted until the simulation resembled the observed open loop response. Using this procedure, the energy loss variation term was found to be ± 10%. With this term defined, the gain was varied to study the stability of the system.

The result is that low gains yield a high standard deviation and error in average velocity. For zero gain, the standard deviation is less than 1.0% of specified carriage velocity, the average

velocity has an error of up to ± 2.5% and the velocity error for an individual pass is less than ± 5.0%. As the gain is increased, the standard deviation decreases slightly (to less than 0.7%), the average velocity error decreases (to less than ±0.3%) and the maximum velocity error for an individual pass is less than ±1.5% as shown in figure 13.

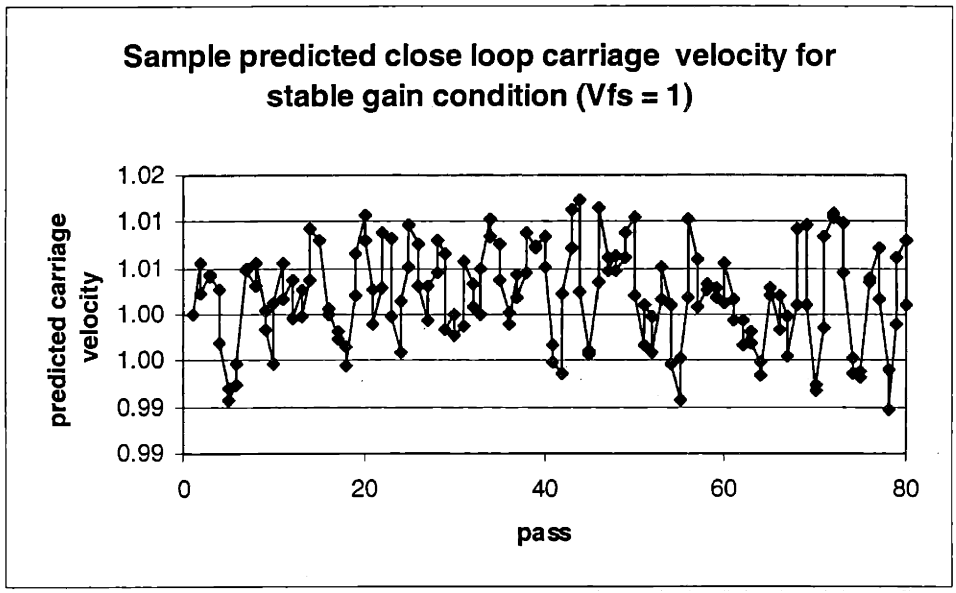


Figure 13 Predicted Stable Close Loop Carriage Performance

Figure 14 shows the condition where gain is increased past the critical value where the carriage velocity suddenly blows up.

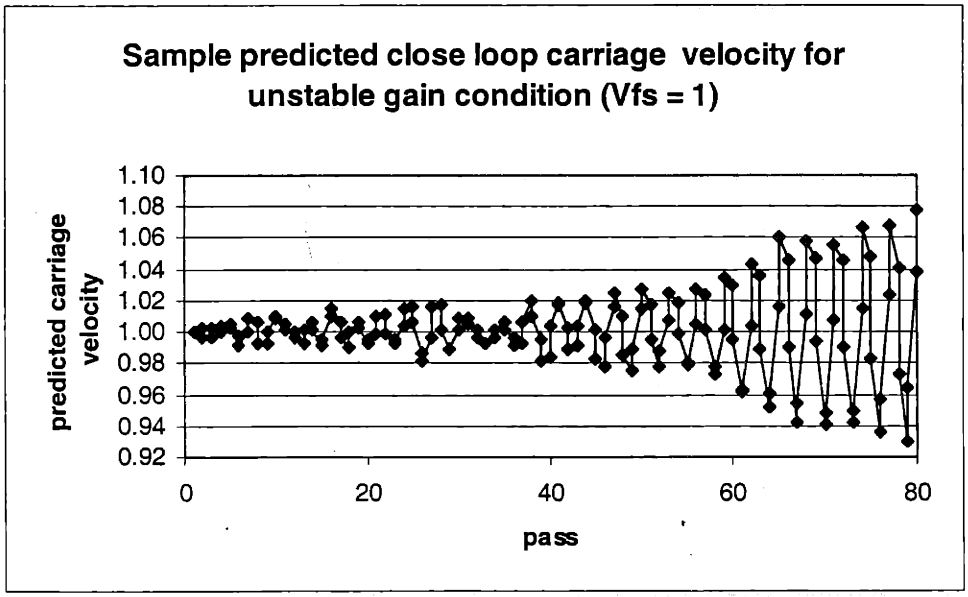


Figure 14 Predicted Unstable Close Loop Carriage Performance

The gain for which the system goes unstable is found to be virtually independent of power loss during collisions, open loop counter mass restoring force or even specified carriage velocity. The instability condition instead results from the discrete nature of the control action and is difficult to change. This model predicts an optimum gain of approximately 5000 and a maximum stable gain of approximately 8400. This model is qualitatively similarly to the real system, but its prediction of optimum and stable gain values are too hi by a factor of 5. This is probably due to the assumption made about counter mass velocity and the estimated fast axis geometry and mass values.

Line spacing predictions

The line spacing controller was simulated early in the control design process to assure that the galvo would be capable of making the necessary actuation. The galvo used has a range of motion of ±15°. Based on the geometry of the slurry bed, the uncorrected nozzle position was chosen to be ±0.0125 m from center. This translates to an angular position of ±0.238 radians (13.6°). Based on the range of motion, this means that the maximum allowable galvo correction is 0.047 radians (1.4°).

The line position error simulation is based on the machine geometry and a simple geometric model of carriage, bed and nozzle position. Figure 15 shows the geometry of the simulation.

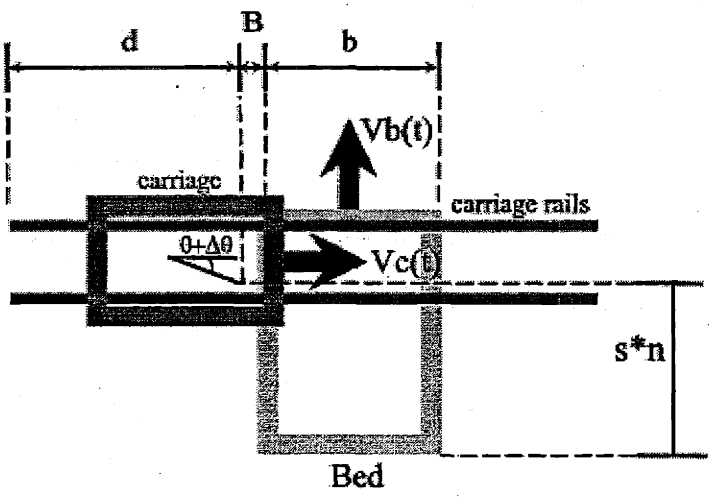


Figure 15 Geometry Used In Analysis

In addition to geometry, carriage frequency, line spacing, and desired carriage velocity are specified. From these variables, commanded carriage and bed velocity are determined by the following relationships.

$$V_{fs} = 2 \cdot (2 \cdot d + b) \cdot f \quad \text{Equation 15}$$

$$V_{bs} = f \cdot s \quad \text{Equation 16}$$

The carriage and the bed velocities are then used to generate an array of velocities that vary randomly by ±10% about the commanded velocities. The spreadsheet then simulates the formation of a layer by calculating the necessary galvo correction based on the random velocity errors and machine geometry.

$$\sin(\Delta\Theta_{fast\_axis} + \Theta) = -\frac{1}{L} \cdot (V_s \cdot \left(\frac{D}{V_f} - \frac{D}{V_f + V_{fa}} \right) - L \sin\Theta) \quad \text{Equation 17}$$

$$\sin(\Delta\Theta_{slow} + \Theta) = \left(\frac{1}{L} \right) \cdot \left(L \cdot \left(\sin(\Theta + \Delta\Theta_{last}) + \frac{V_{sa}}{f} - s \right) \right) \quad \text{Equation 18}$$

$$\Delta\Theta_{total} = \Delta\Theta_{fast\_axis} + \Delta\Theta_{slow\_axis} \qquad \text{Equation 19}$$

The following figures are two results from the same simulation algorithm. The variation between the two figures results from the variation in the spreadsheet's random value generator used in the simulation.

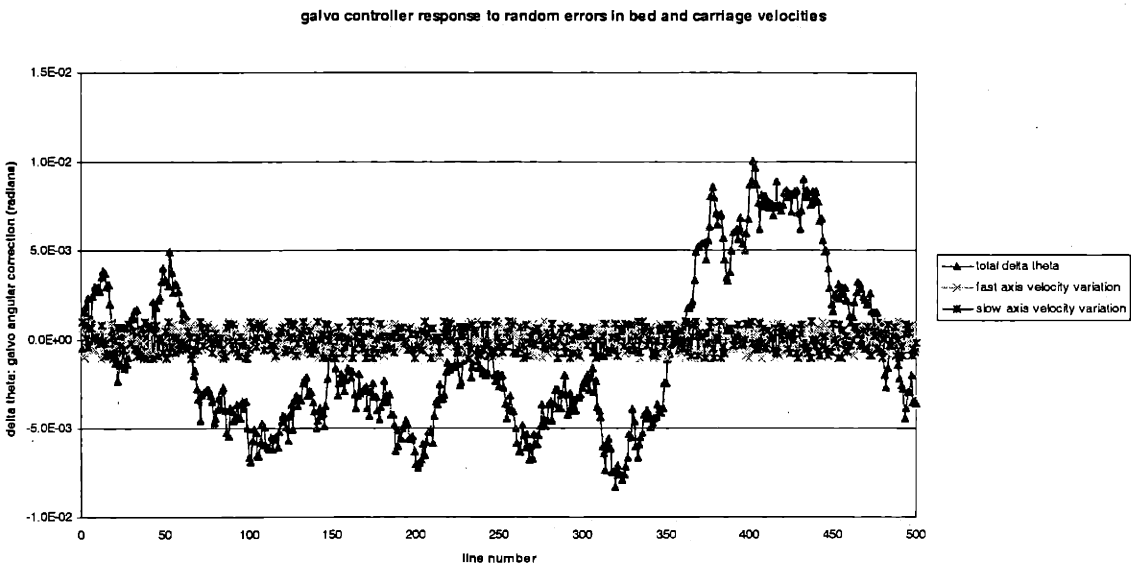


Figure 16 Predicted Required Galvo Adjustment For Random Velocity Errors

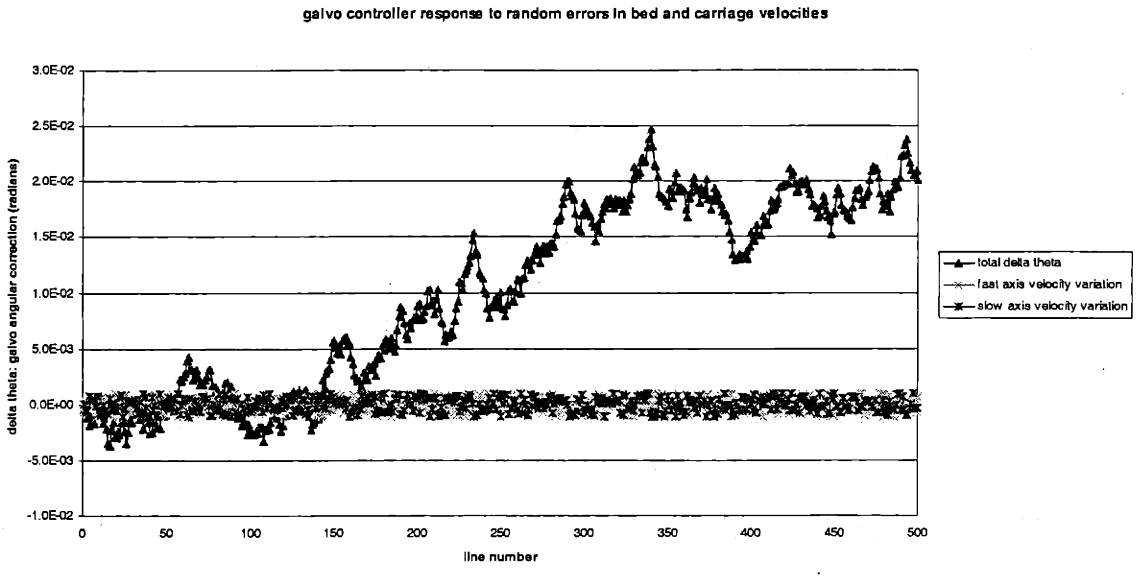


Figure 17 Predicted Required Galvo Adjustment With Different Random Velocity Errors

In each case, the predicted galvo correction never exceeds 0.05 radians (1.43°) after 500 lines. Because the slurry bed is only 21 cm wide and the expected line spacing is about 0.25mm, a more realistic number of expected lines is 100. When the simulation is stopped at 100 lines, the maximum predicted correction is only 0.015 radians (0.430°). Compared to a maximum range of motion of 0.047 radians, this simulation suggests the galvo line position controller is feasible.

MEASURED MACHINE PERFORMANCE

Slow axis

The slow axis velocity shows an instantaneous velocity error of ±10% but approximately ±0.1% variation when measured over the period of the carriage motion. This averaged velocity is most important because it determines the open loop position of the bed. No plots are available because the data gathering program could not acquire reliable data for the slow axis velocity.

Fast axis

The fast axis velocity controller's performance is characterized by its stability, step response, steady state error, and disturbance rejection capabilities. The stability was qualitatively evaluated for varying counter mass and carriage motor gains. At low gain, the carriage motion was stable and would even accept some random disturbances to the counter mass without crashing. Crashing occurs when the close loop system becomes unstable and the large motor outputs cause the counter mass to exceed its range of motion and cease carriage motion. As the gain of each motor was varied both separately and together to find the maximum acceptable gains that maintained carriage stability. Once these values were determined both gains were set to slightly lower values (to ensure stability).

The step response of the system was then measured for varying values of stable gain. Figures 18 and 19 show the step response for several of these values. As expected, the system time constant is reduced as gain is increased (from about 1 sec to 0.5 sec). The response only exhibits an overshoot when the gain is almost at an unstable value.

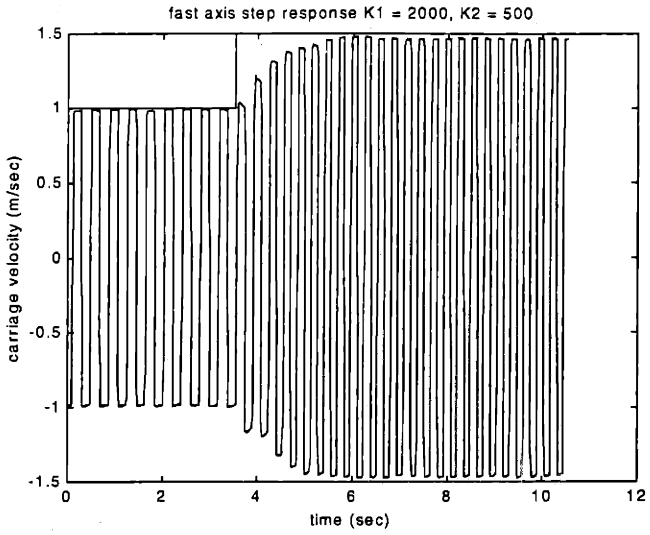


Figure 18 Carriage Motor Step Response For High Gain

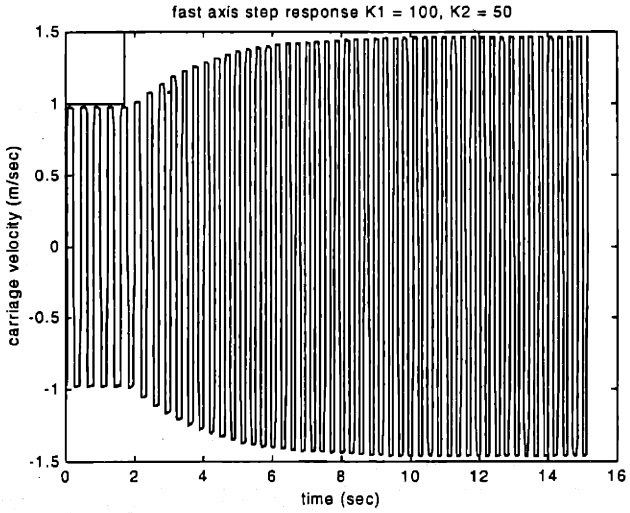


Figure 19 Carriage Motor Step Response For Low Gain

Steady state carriage velocity error is of great interest because the nozzle velocity determines the quality of the deposited lines as well as the expected line position errors that must be corrected. For a set velocity of 1.5 m/sec, the measured steady state performance was recorded for four conditions; no gain, no carriage motor gain, no counter mass motor gain, both gains turned on. With both gains turned off, the counter mass output voltage is set to ±Vout which is calculated once from an empirical formula relating average carriage velocity to output voltage. As the gains are turned on, the active portion of the counter mass and carriage motor outputs becomes

proportional to the measured velocity error. The steady state response for the four possible conditions are shown in figure 20.

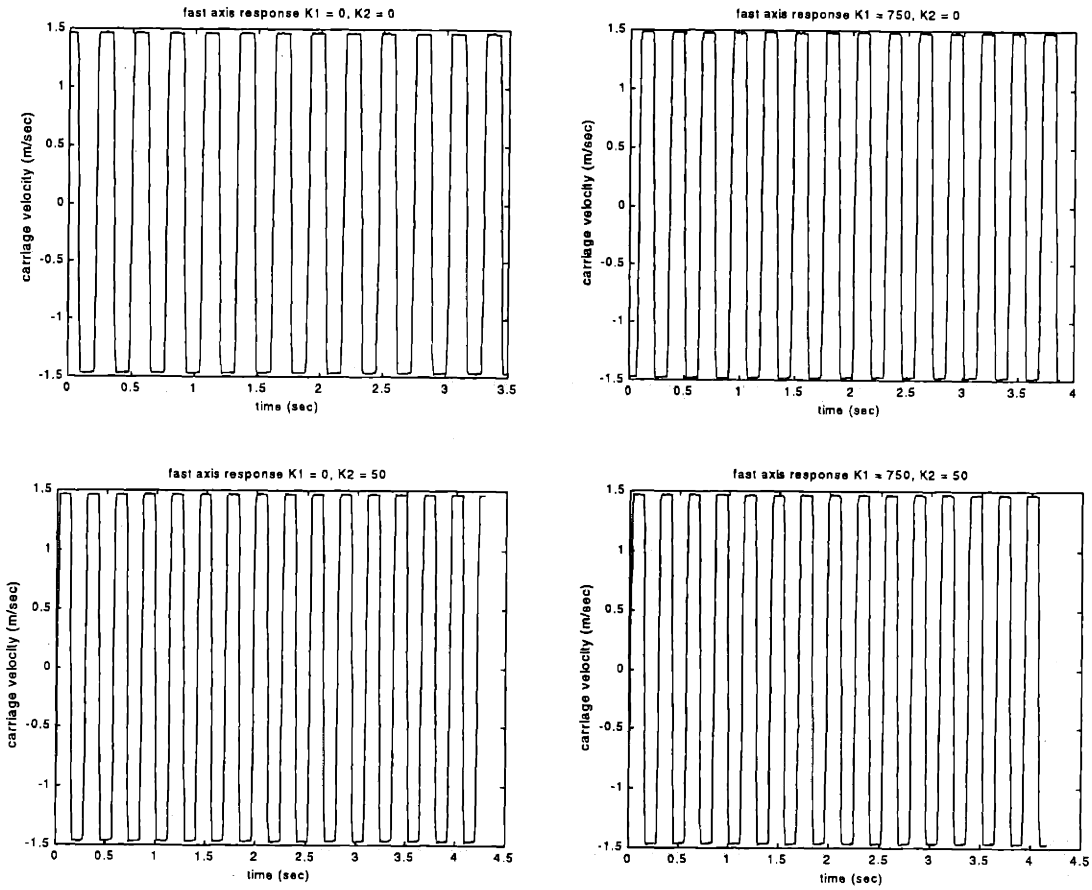


Figure 20 Steady State Carriage Velocity Response For Varying Controller Gain

The data calculated from these plots is shown in the following table. It shows the percent error of the averaged carriage velocity as well as the maximum percent error in instantaneous carriage velocity.

Recorded Carriage Velocity Errors		
	average % error	max % error
Open loop control	2.17	4.02
Counter mass motor close loop control	2.18	4.16
Carriage motor close loop control	2.20	3.39
Counter mass & carriage close loop	1.40	2.14

This indicates that, for the employed gains, the counter mass and carriage controllers cause no or slight improvement in error when run individually, but yield a substantial improvement in steady state performance when run simultaneously. The reasons for this behavior are not currently understood and more detailed data is required to better understand this behavior.

Line spacing results

Line spacing performance is diagnosed by measuring ink lines printed on white paper. The resulting pattern is shown in figure 21.

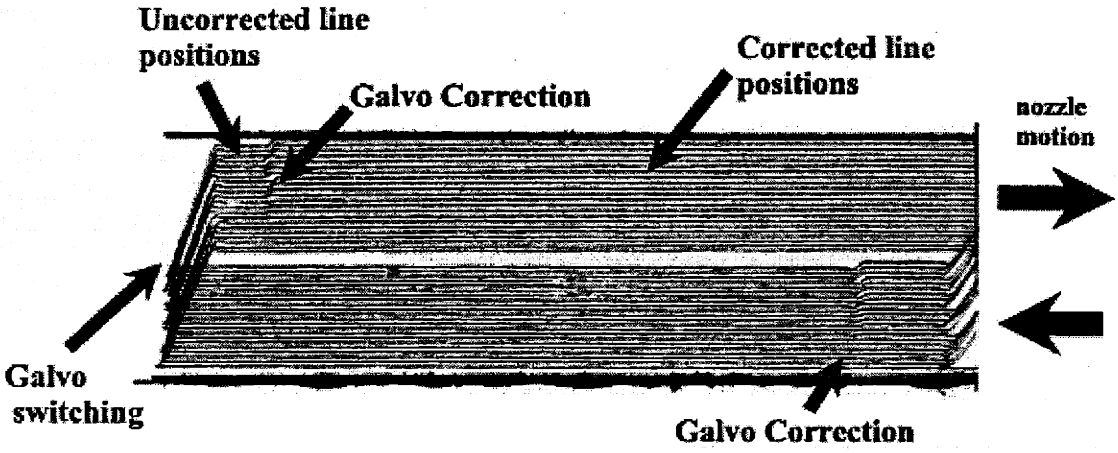


Figure 21 Typical Ink Pattern (shown smaller than actual size)

There are several important features of this pattern. Primarily, there are two 22mm wide ink beds. The upper bed is printed with the nozzle moving from left to right and the lower bed with nozzle moving from right to left. The two dark lines above and below the pattern result from the rest position of the galvo during the fast axis startup and shutdown. These would deposit slurry into a catch pan during normal operation and not appear on the slurry bed. Another feature is the small step change in line position approximately 2cm from the beginning of each line. This is

the galvo position correction. This pattern allows the measurement of the open loop line position (before correction) and the corrected position for each ink test.

Line Position Measurement Technique

All of the line spacing measurements were made with a scope equipped with a crosshair and digital calipers that control object position in two axes. The line spacing was measured by aligning the crosshair with the estimated center of each line, moving the calipers so the crosshair was centered with the next line, and reading the distance measurement. Figure 22 shows a magnified view of the ink lines as seen through a low magnification scope.

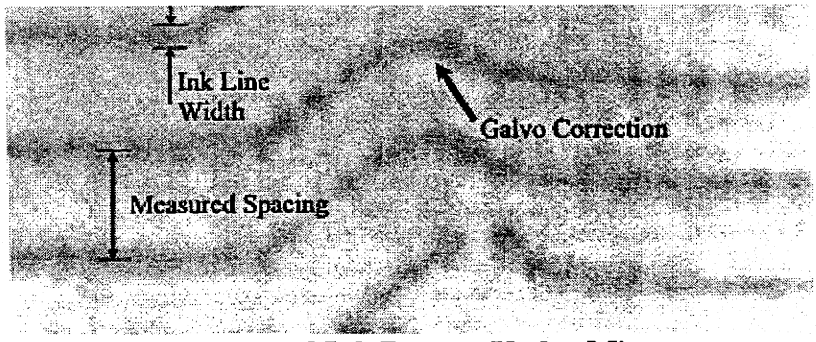


Figure 22 Typical Ink Pattern Under Microscope

The lines have rough edges making accurate measurement difficult. The measured ink line widths range from 0.20 mm to 0.48 mm. This uneven line thickness leads to measurement repeatability errors on the order of 0.1 mm. This error is the same order of magnitude as the measured line position error. This means that the true line position performance may be better than estimated by this measurement method.

Results

The first line position controller implemented was based on the relative line position. This algorithm calculates line position based on specified line spacing relative to the recorded position

of the last line. The measured line position and spacing for a commanded spacing of 1.0 mm and nozzle velocity of 1 m/sec is shown in figures 23 and 24. Figure 23 shows the absolute position of each line before the galvo correction, after the correction and the ideal line positions. This plot shows that the relative algorithm does not bring the corrected line positions closer to the ideal position.

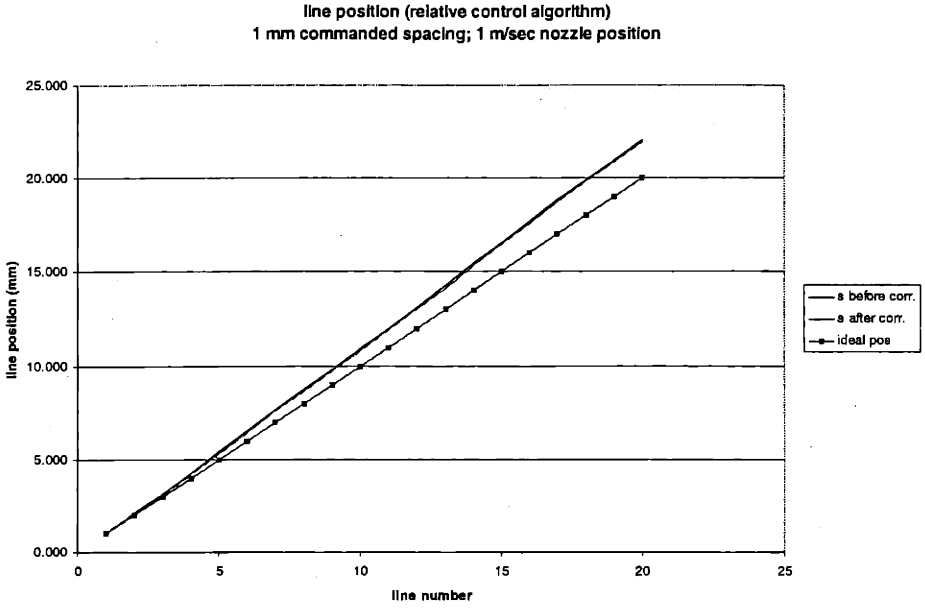


Figure 23 Relative Line Position Controller Performance

Figure 24 shows the line spacing measured from this test. As with the line position, the corrected line spacing is not much improved over the open loop results.

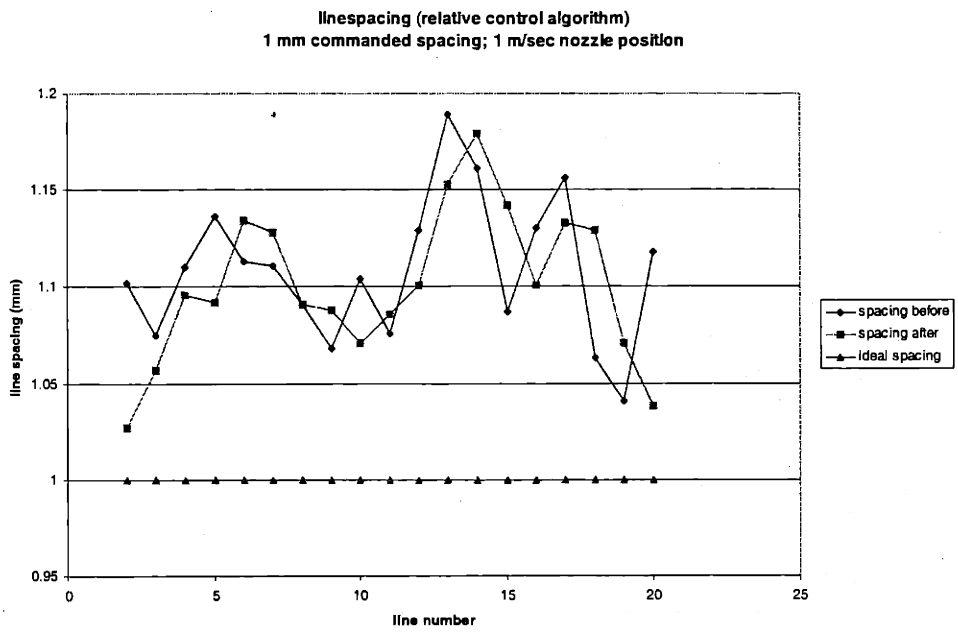


Figure 24 Relative Spacing Position Controller Performance

These tests demonstrate that the relative line spacing controller is not effective in controlling line position and spacing. It is however, a useful development exercise as the algorithm is similar to the absolute line position controller but somewhat simpler.

After the relative control program functioned properly it was modified to calculate line position error based on an absolute position scale as described previously. The results of this algorithm are shown below. Figures 25 and 26 show the closed loop line position and spacing for a specified spacing of 0.5 mm and a nozzle velocity of 1.5 m/sec. Figure 25 shows the corrected line position data falling along the desired line while the uncorrected spacing is clearly in error.

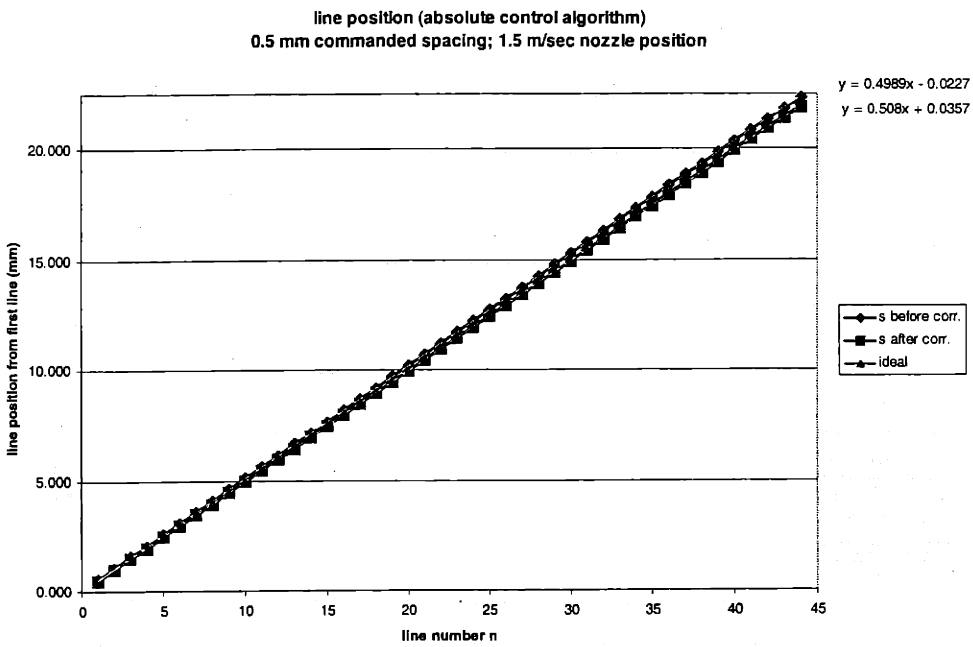


Figure 25 Absolute Line Position Controller Performance (small errors)

Figure 26 shows the corrected and uncorrected line spacing for this test. For the uncorrected lines, the average spacing is 0.508 mm with a standard deviation of 0.017 mm. For the corrected positions, the average line spacing is 0.499 mm with a standard deviation of 0.020 mm.

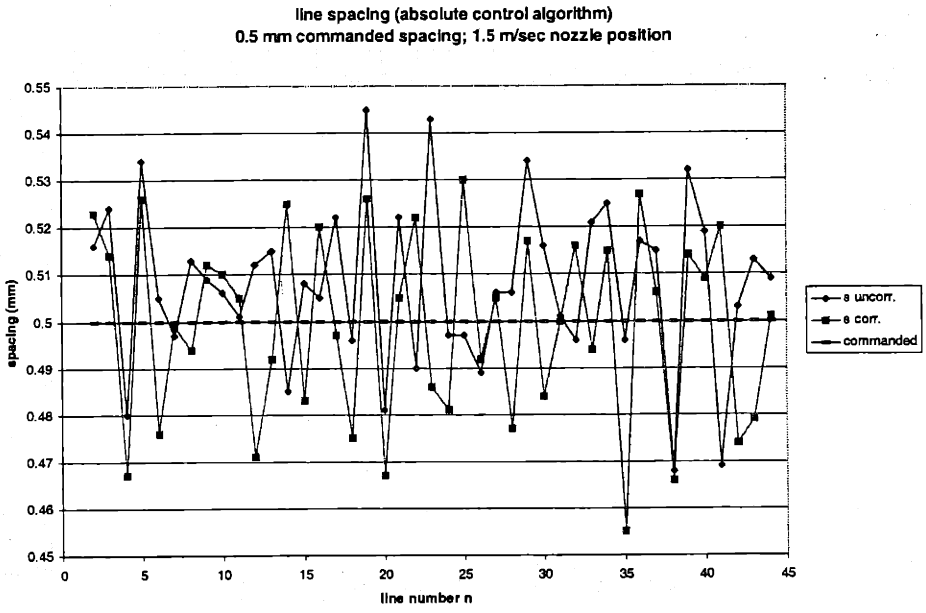


Figure 26 Absolute Line Position Controller Performance (small errors)

Figures 27 and 28 show the performance of the same controller for a commanded spacing of 1.5 mm and a nozzle velocity of 1 m/sec. In this case, the larger commanded line spacing exaggerates the uncorrected position error. The line position shown in figure 27 once again shows the corrected line data falling closely on the ideal position.

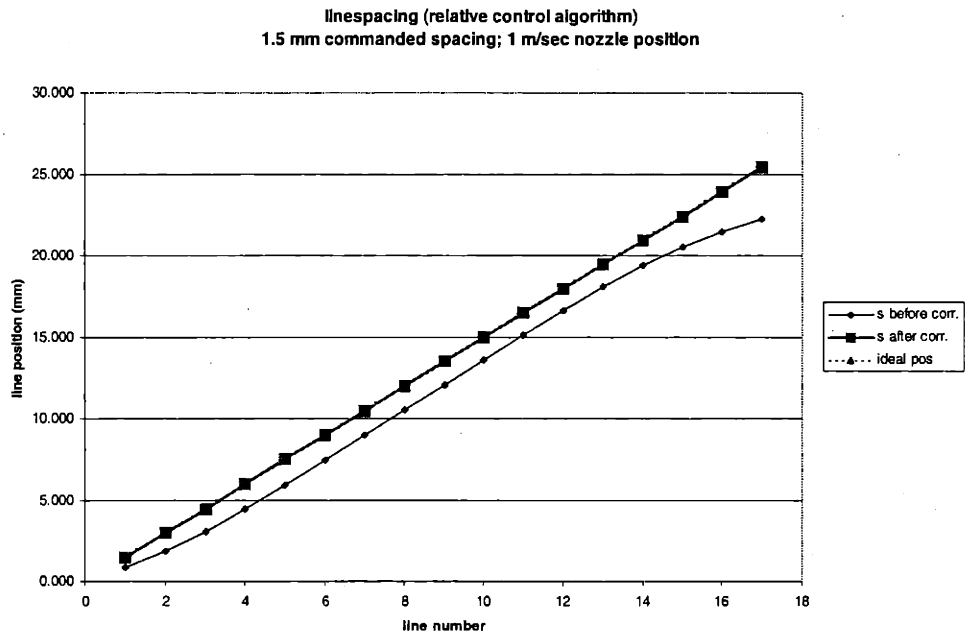


Figure 27 Absolute Line Position Controller Performance (large errors)

Figure 28 of line spacing also shows the amplified effects of the line position controller. The large errors in uncorrected line spacing result from the acceleration of the slow axis. Despite this large disturbance, the line position controller is able to compensate resulting in an average corrected spacing of 1.499 mm even with an average uncorrected spacing of 1.341 mm.

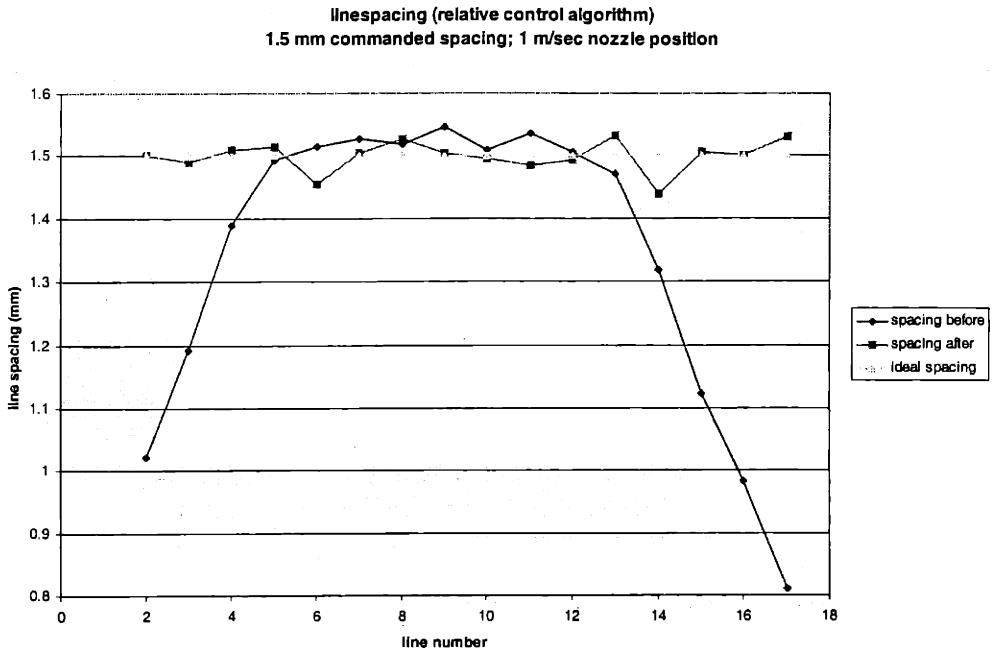


Figure 28 Absolute Line Position Controller Performance (large errors)

In all cases, the measured line position errors fall within ± 0.06 mm (± 60 μm) of the commanded positions. Because the expected slurry line widths are approximately 0.5 mm, this accuracy should be adequate to control the effects of line registration on layer and part characteristics. These, and other data samples show that the absolute line position algorithm and hardware effectively controls line position and enables line registration between layers.

CONCLUSIONS AND FURTHER WORK

The control algorithms developed for the 3D printing machine fast axis, slow axis and line position controller are capable of creating a slurry layer with adequately controlled line spacing and registration. There is however, the opportunity for further work. Further modeling of the fast axis dynamics is required to fully understand the controller stability problem. The dependence of the steady state performance on the interaction of the counter mass controller gain and the carriage controller gain is also not understood. This must be explored if better nozzle velocity control is to be achieved. If the line position controller's performance is to be measured better, more accurate line position and registration diagnostics are required. Perhaps most importantly, the effects of line registration on final slurry layer and part characteristics should be explored.

Appendix A: PLC controller code

```
#include "macros"

OPEN PLC 1              ;measure frequency
clear
;measure frequency based on last 25 passes
if (Q52!=x)                                     ;if carriage changes direction
        disable plc 3                           ;stop counter 3
        ta_f=counter_3_elapsed                  ;store time for pass
        enable plc 3                            ;reset counter
        n=n+1                                   ;increment n
        if (n<25)                               ;for the last n passes
                Q(1000+n)=1/(2*ta_f)            ;store the calculated frequency
        else                                    ;when n reaches a maximum
                n=0                             ;reset n to 0
                Q(1000+n)=1/(2*ta_f)            ;store calculated frequency
        endif
endif
Q52=x                                           ;set dummy variable = x
                                                ;calculate the average frequency over the last n passes
f=(Q1000+Q1001+Q1002+Q1003+Q1004+Q1005+Q1006+Q1007+Q1008+Q1009+Q1010+Q1011+Q1012+Q1013+
Q1014+Q1015+Q1016+Q1017+Q1018+Q1019+Q1020+Q1021+Q1022+Q1023+Q1024)/25
close


OPEN PLC 2              ;output positions and velocities in m and m/sec for watch window
clear
P166=M166*((.0000001)/(I0109*32))/((I10/8388608)/1000)          ;slow axis velocity (m/s)
P162=M162*(.0000001)/(I0108*32)                                 ;slow axis position (m)
P174=M174*((.0000001)/(I0109*32))/((I10/8388608)/1000)          ;slow axis average velocity (m/s)
P262=M262*(.000005)/(I0208*32)                                  ;fast axis position (m)
P266=M266*((.000005)/(I0209*32))/((I10/8388608)/1000)           ;fast axis velocity (m/s)
P274=M274*((.000005)/(I0209*32))/((I10/8388608)/1000)           ;fast axis average velocity (m/s)
close


OPEN PLC 3              ;timer 3
clear
P90=16777216                            ;counter rollover value (2^24)
P97=M0                                  ;store starting value of counter M0
P98=0                                   ;initialize timer 3
While (P98 > -1)
        P98=((M0-P97)%P90)*(I10/8388608)/1000   ;time 3 elapsed in seconds
Endwhile
close

OPEN PLC 4              ;timer 0
clear
P90=16777216                            ;counter rollover value (2^24)
P91=M0                                  ;store starting value of counter M0
P92=0                                   ;initialize timer 0
While (P92 > -1)
        P92=((M0-P91)%P90)*(I10/8388608)/1000   ;time 0 elapsed in seconds
Endwhile
close

OPEN PLC 5              ;timer 1
clear
P90=16777216                            ;counter rollover value (2^24)
```

```
P93=M0                                  ;store starting value of counter M0
P94=0                                   ;initialize timer 1
While (P94 > -1)
        P94=((M0-P93)%P90)*(I10/8388608)/1000   ;time 1 elapsed in seconds
Endwhile
close


OPEN PLC 6                      ;define direction of carriage (x) and mark
clear
IF (Vfa > 0)                    ;if Vfa is positive
x = 0                           ;designate pos direction
mark = 0 -(b)/2 -B              ;set mark 0 for positive direction (center -b/2 -B)
ELSE                            ;if Vfa is negative
x = 1                           ;designate neg direction
mark = 0 + (b)/2 + B            ;set mark 0 for positive direction (center +b/2 +B)
ENDIF
close

OPEN PLC 7              ;galvo home actuation before correction

clear

if (P24=1)                      ; dummy variable so other plc's can trigger galvo switching
if (x=0)                        ; going in positive direction
        if (P12=0)              ;dummy variable (do only once each pass)
                if (abs(theta_conversion*theta) < 5000)
                        DAC4=theta_conversion*theta             ;actuate galvo to positive home position on
velocity sign change
                else
                        DAC4=5000               ;default max output
                endif
                P12=1                           ;reset dummy variable
        endif
endif

if (x=1)                        ; going in negative direction
        if (P12=1)              ;dummy variable (do only once each pass)
                if (abs(theta_conversion*theta) < 5000)
                        DAC4=theta_conversion*(-theta)  ;actuate galvo to negative home position on velocity
sign change
                else
                        DAC4=-5000              ;default min output
                endif
                P12=0                           ;reset dummy variable
        endif
endif
endif

close

OPEN PLC 8              ;galvo correction and actuation
clear

Vss = s*(-0.337*Vf_ave_s*Vf_ave_s + 2.8452*Vf_ave_s + 0.1407)           ;convert to slow axis specified
speed based on lookup frequency
;Vss = s*f                              ;convert to slow axis specified speed based on measured frequency
Ynoz = Yfa + noz_off                    ;calculate nozzle position based on offset

if (abs(Ynoz) < abs(mark))              ;when position of nozzle passes mark

        ;(fast axis correction)
```

```
        if (x=0)                                        ;for positive direction
                if (P11=0)                              ;trigger on dummy variable
                        ta = counter_0_elapsed          ;recorded time between pos marks
                        disable PLC 4                   ;stop counter 0
                        enable PLC 4                    ;restart counter 0 at 0 sec

                ;(slow axis correction)


                        Vsa_ave = (Xs-Xs_last_0)/ta             ;calc Vsa actual average slow axis velocity
since last line
                        Xs_exp = (Xs_first_0 - n0*s)            ;calc expected slow axis position
                        delta_x = Xs - Xs_exp - offset*s        ;calc slow axis position error plus offset
correction
                        delta_theta = (asin(sin(theta)-(delta_x/L))-theta)      ;convert to galvo correction (rad)
                        Xs_last_0 = Xs                          ;record last 0 position of slow axis
                        n0 = n0 + 1                             ;increment 0 pass #
                        P11=1                                   ;reset dummy variable
                endif
        else                                            ;for negative direction
                if (P11=1)                              ;trigger on dummy variable
                        ta = counter_1_elapsed          ;recorded time between neg marks
                        disable PLC 5                   ;stop counter 1
                        enable PLC 5                    ;restart counter 1 at 0 sec

                ;(slow axis correction)
                        Vsa_ave = (Xs-Xs_last_1)/ta             ;calc Vsa actual average slow axis velocity
since last line
                        Xs_exp = (Xs_first_1 - n1*s)            ;calc expected slow axis position
                        delta_x = Xs - Xs_exp - offset*s        ;calc slow axis position error plus offset
correction
                        delta_theta = (asin(sin(theta)-(delta_x/L))-theta)      ;convert to galvo correction (rad)
                        Xs_last_1 = Xs                          ;record last 1 position of slow axis
                        n1 = n1 + 1                             ;increment 1 pass #
                        P11=0                                   ;reset dummy variable
                endif
        endif


        if (abs(M101) > abs(Xs_i))          ;if the slow axis is in position where the bed has started to pass under
the nozzle
        and (abs(M101) < abs(Xs_f-Xs_i))    ;and the slow axis has not compleated its pass past the nozzle

                if (x = 0)                  ;for positive direction
                and (Ynoz > mark)                   ;when carriage hits mark
                and (P24=1)                         ;trigger for other programs to turn on galvo correction
                and (abs(theta_conversion*(delta_theta+theta)) < 5000)      ;set software limit on total galvo
output
                        DAC4=theta_conversion*(delta_theta+theta)           ;actuate galvo based on delta theta
                endif
                if (x = 1)                  ;for negative direction
                and (Ynoz < mark)                   ;when carriage hits mark
                and (n1 > 0)                        ;trigger to prevent stray line
                and (P24=1)                         ;trigger for other programs to turn on galvo correction
                and (abs(theta_conversion*(delta_theta+theta)) < 5000)      ;set software limit on total galvo
output
                        DAC4=-theta_conversion*(delta_theta+theta)          ;actuate galvo based on delta theta
total
                endif
        endif
endif
```

```
close

OPEN PLC 9              ;fast axis counter mass closed loop driver
clear

;drive fast axis control (proportional control velocity loop)
;compare average actual velocity to specified velocity
;use difference*constant feedback to adjust driving current

out_neg = -out_pos              ;set voltage should be equal and opposite for each direction
if (x=1)                ;for positive passes
        Q50 = out_pos + K1*(Vf_ave_s-Vf_ave_0)   ;Q50 = optimum open loop voltage + correction based on delta
V_ave
        if (Q50 < 2*3276.8)                      ;limit Q50 voltage
                DAC3 = Q50                       ;output signal to DAC4
        else
                DAC3 = 1*3276.8                  ;output maximum voltage
        endif
else
        Q50 = out_neg - K1*(Vf_ave_s-Vf_ave_1)   ;Q50 = optimum open loop voltage + correction based
on delta V_ave
        if (Q50 > -2*3276.8)                     ;limit Q50 voltage
                DAC3 = Q50                       ;output signal to DAC4
        else
                DAC3 = -1*3276.8                 ;output maximum voltage
        endif
endif
close


OPEN PLC 10             ;measure fast axis average velocity over bed
clear
if (x=0)                                         ;for positive passes
        if (Yfa > -b/2)                          ;when the carriage begins passing over the bed
                if (P13=0)                       ;dummy variable to make sure this only happens once each
pass
                        enable PLC 11            ;start counter 2
                        P13=1                    ;dummy variable
                endif
        endif

        if (Yfa > b/2)                           ;when carriage begins passing over the opposite bed edge
                if (P13=1)                       ;dummy variable
                        disable PLC 11           ;stop counter 2
                        ta_f_0 = counter_2_elapsed          ;store elapsed time
                        Vf_ave_0=b/(ta_f_0)                 ;calculate average velocity over bed
                        P13=0                               ;dummy variable
                endif
        endif
else                                             ;for negative passes
        if (Yfa < b/2)                           ;when the carriage begins passing over the bed
                if (P14=0)                       ;dummy variable to make sure this only happens once
                        enable PLC 11            ;start counter 2
                        P14=1                    ;dummy variable
                endif
        endif

        if (Yfa < -b/2)                          ;when carriage begins passing over the opposite bed edge
                if (P14=1)                       ;dummy variable
                        disable PLC 11           ;stop counter 2
                        ta_f_1 = counter_2_elapsed          ;store elapsed time
                        Vf_ave_1=b/(ta_f_1)                 ;calculate average velocity over bed
```

```
                P14=0                                   ;dummy variable
            endif
        endif
endif

close


OPEN PLC 11         ;timer 2
clear                                       ;define timer 2
P90=16777216                                ;counter rollover value (2^24)
P95=M0                                      ;store starting value of counter M0
P96=0                                       ;initialize timer 2
While (P96 > -1)
        P96=((M0-P95)%P90)*(I10/8388608)/1000   ;time 2 elapsed in seconds
Endwhile
close


OPEN PLC 12         ;seek open loop counter mass driving voltage that gives specified velocity
clear

out_pos_set = 148.36*Vf_ave_s*Vf_ave_s + 230.3*Vf_ave_s + 45.123        ;new lookup formula

if (Vf_ave_0 > ((1-(percent/100))*Vf_ave_s))            ;if the actual average velocity is equal to the specified
velocity
and (Vf_ave_0 < ((1+(percent/100))*Vf_ave_s))           ;+- a small percent
            P15 = P15 + .005                            ;increment variable P15
endif

if (Vf_ave_0 < ((1-(percent/100))*Vf_ave_s))        ;if the average velocity of the last pass was too slow
;and (out_pos < 10000)                              ;set upper limit for voltage output
and (out_pos < out_pos_set)                                 ;For use with lookup formula
            ;out_pos = out_pos + 0.10*(Vf_ave_s-Vf_ave_0)   ;adjust voltage to find V_ave_s
            out_pos = out_pos + 1                           ;For use with lookup formula
endif

if (Vf_ave_0 > ((1+(percent/100))*Vf_ave_s))        ;if the average velocity of the last pass was too fast
;and (out_pos > 150)                                ;set lower limit for voltage output
and (out_pos > out_pos_set)                                 ;For use with lookup formula
            ;out_pos = out_pos + 0.10*(Vf_ave_s-Vf_ave_0)   ;adjust voltage to find V_ave_s
            out_pos = out_pos - 1                           ;For use with lookup formula
endif
close

OPEN PLC 13         ;close loop on counter mass once open loop velocity/voltage is satisfactory
clear
if (P15 > 10)               ;if the actual velocity has been close to the specified velocity for a sufficient time
        K1=1000             ;set counter mass gain to close the loop
        K2=1000             ;set the carriage gain
        disable plc 12      ;stop changing out_pos
        enable plc 15       ;turn on carriage motor
        DAC2 = 0            ;set carriage motor to zero output
        disable plc 13      ;disable this plc
endif
close

OPEN PLC 14         ;activate PLC 12 when commanded velocity changes
clear
if (Q51!=Vf_ave_s)          ;if the V_ave_s is changed from what is was
        K1=500              ;reduce counter mass gain transient
        K2=50               ;reduce carriage gain for transient
```

```
        P15=0                   ;reset P15
        enable plc 12           ;start searching for optimum out_pos
        enable plc 13           ;reset plc 13
        disable plc 15          ;turn off carriage motor
endif
Q51 = Vf_ave_s          ;set dummy variable = to new Vf_ave_s
close


OPEN PLC 15             ;close loop control of carriage motor
clear
out_car = K2*(abs(Vf_ave_s) - abs(Vfa))         ;proportional velocity controller
if (Yfa > (-b/2 - B))                           ;if the carriage is over the bed
and (Yfa < (b/2 + B))
        if (x=0)                        ;for positive direction
                DAC2 = -out_car         ;output commanded voltage
        else
                DAC2 = out_car          ;output commanded voltage
        endif
endif
if (Yfa < (-b/2 - B))                   ;if the carriage is not over the bed
and (Yfa > (b/2 + B))
        out_car = 0                     ;output is zero
endif
close

OPEN PLC 16             ;fast axis carriage open loop driver for start and break
clear
if (x=0)                ;for positive passes
        if (out_car < .5*3276.8)                ;limit Q70 voltage
                DAC2 = out_car                  ;output signal to DAC3
        else
                DAC2 = .25*3276.8               ;output maximum voltage
        endif
else
        if (out_car < .5*3276.8)                ;limit Q70 voltage
                DAC2 = -out_car                 ;output signal to DAC3
        else
                DAC2 = -.25*3276.8              ;output maximum voltage
        endif
endif
close

OPEN PLC 17             ;fast axis PD position controller
clear
out_car = Kp*(Yfs - Yfa) - Kd*(Vfa)     ;proportional position control with added damping to stabilize sys
if (abs(out_car) < pos_limit)           ;if command < pos_limit
DAC2 = out_car                          ;output command voltage
else
DAC2 = ((abs(out_car))/(out_car))*pos_limit ;output upper limit voltage to carriage motor
endif
close

OPEN PLC 18             ;fast axis centering program
clear

Kp = 10000                              ;set carriage position proportional gain
Kd = 7500                               ;set carriage position damping gain
if (counter_0_elapsed < 10)             ;for some time
        pos_limit = 650                 ;limit the max carriage output
        enable plc 4, 17                ;enable timer and position controller
        P101=0.175                      ;set first commanded position
```

```
endif
if (counter_0_elapsed > 10)                ;after first time period
and (counter_0_elapsed < 20)
and (abs(Vfa) < .001)                      ;when carriage velocity is close to zero
and (P22=0)                                ;trigger on dummy variable
        Q200 = Yfa                         ;record first position
        P101 = -0.175                      ;set second commanded position
        P22=1                              ;reset dummy variable
endif
if (counter_0_elapsed > 20)                ;after second time period
and (counter_0_elapsed < 21)
and (abs(Vfa) < .001)                      ;when carriage velocity is close to zero
and (P22=1)                                ;trigger on dummy variable
        Q201 = Yfa                         ;record second carriage position
        P22=0                              ;reset dummy variable
endif
if (counter_0_elapsed > 21)                ;for next time period
and (counter_0_elapsed < 22)
and (abs(Vfa) < .001)
        P101 = (Q200+Q201)/2               ;set new commanded position (half way between 1st & 2nd recorded
positions)
        d = (abs(Q201-Q200)-b)/2           ;calculate dimension d for use elsewhere
endif
if (counter_0_elapsed > 40)                ;after the total specified time has elapsed
        disable plc 4,17                   ;turn off timer and position controller
        M210=1                             ;enable writing to encoder 2 register
        M203=0                             ;write zero value to encoder 2 register
        M210=0                             ;disable writing to encoder 2 register
        M202=0                             ;make sure carriage output is zero
        pos_limit = 3200                   ;reset carriage output limit for other programs
        counter_0_elapsed = 0              ;reset counter elapsed time
        disable plc 18                     ;disable this plc
endif
close

OPEN PLC 19             ;galvo running schedule
clear

if (abs(M101) < abs(Xs_i))                 ;if slow axis position is before the edge of the bed
        DAC4 = -5000                       ;output negative default position
endif


if (abs(M101) > abs(Xs_i))                 ;if slow axis has reached the starting position
and (x = 1)                                ;trigger on neg velocity to prevent stray lines
and (P23 = 0)                              ;dummy variable, only run this routine once
        if (offset = 0)                    ;only change Xs_first if there is no offset specified
                Xs_first_1 = Xs            ;record location slow axis at start of first line for absolute mode line
registration
        endif
        Xs_first_0 = Xs_first_1 -(Vss/(2*(-0.337*Vf_ave_s*Vf_ave_s + 2.8452*Vf_ave_s + 0.1407)))        ;
calculate Xs_first_0 based on lookup frequency
        ;Xs_first_0 = Xs_first_1 - (Vss/(2*f))                  ;based on measured frequency
        n0=0            ; reset line counter for positive direction
        n1=0            ; reset line counter for negative direction
        P24=1           ; reset dummy variable
        enable plc 7    ; begin galvo switching routine
        P23=1           ; reset dummy variable
endif

if (abs(M101) > abs(Xs_f-Xs_i+100))                ;if slow axis position is past end of bed
and (x = 0)                                        ;trigger on positive direction to prevent stray lines
```

```
        disable plc 7               ;turn off galvo switching routine
        DAC4 = 5000                 ;set galvo to positive default position
endif

close


OPEN PLC 20             ;fast axis startup
clear
if (P20=0)                                      ;dummy variable to trigger action
        n0=1                                    ;reset positive line count
        n1=1                                    ;reset negative line count
        K1=0                                    ;set counter mass gain for startup
        K2=0                                    ;set carriage gain for startup
        pos_limit = 3276                        ;set position controller output limit
        out_pos = 150                           ;set counter mass output voltage
        out_car = 500                           ;set a starting carriage output in prog units
        DAC4 = -5000                            ;put galvo in starting position
        disable plc 17                          ;turn off carriage position controller
        enable plc 1,2,6,8,9,10,16              ;enable measuring plc's and carriage motor driver
        P101=0.08                               ;set rest fast axis position
        P21=0                                   ;reset dummy variable
        P23=0                                   ;reset dummy variable
        P24=0                                   ;reset dummy variable
        P25=0                                   ;reset dummy variable
        P26=0                                   ;reset dummy variable
        P20=1                                   ;reset dummy variable
endif

if (out_car > 200)                      ;if carriage output is greater than a minimum value
        out_car = out_car - 0.005*n0    ;slowly decrease carriage driving voltage as it becomes stable
else
        out_car = 200                   ;otherwise output minimum voltage
endif

if (n0 > 15)                            ;after 15 positive passes
        disable plc 16                  ;turn off carriage motor
        out_car = 0                     ;reset carriage output to zero
        out_pos = 350                   ;reset counter mass output to a positive starting value
        enable plc 12,13,14,15,19       ;turn on cose loop counter mass, carriage and galvo drivers
        Vf_ave_s = Vf_run               ;set initial desired velocity (this will cause counter mass to
                                        ;begin seeking new open loop voltage)
        disable plc 20                  ;turn off this start program
endif

close


OPEN PLC 21             ;fast axis stop
clear
Vf_ave_s = 1                            ;set desired speed (to slow down axis before breaking)
if (Vf_ave_0 < 2.0)                     ;when desired speed is reached
        if (P21=0)                      ;dummy variable to do this action only once
                enable plc 3            ;use timer 3 to time stop
                disable plc 1,9,12,15   ;turn off close loop counter mass and carriage drivers
                DAC3 = 0                ;set counter mass output voltage to zero
                DAC2 = 0                ;set carriage motor output voltage to zero
                enable plc 16           ;turn on open loop carriage motor driver
                out_car = -1200         ;output negative voltage to break carriage
                P21=1                   ;reset dummy variable
                P20=0                   ;reset dummy variable
        endif
```

```
        if (counter_3_elapsed > 2)          ;after a set amount of time
                out_car = 0                 ;set carriage velocity command to zero
                DAC2 = 0                    ;set carriage output to zero
                K1=0                        ;set counter mass gain to zero
                disable plc 1,3,4,5,7,8,9,10,11,12,13,14,15,16,20,21   ;disable all plc's
        endif
endif
close



; 1 measure frequency
; 2 watch positions and velocities (m and m/sec)
; 3 timer 3
; 4 timer 0
; 5 timer 1
; 6 define direction of carriage (x) and mark
; 7 galvo home actuation before correction
; 8 galvo correction and actuation
; 9 fast axis counter mass closed loop driver
; 10 measure fast axis average velocity over bed
; 11 timer 2
; 12 seek open loop counter mass driving voltage that gives specified velocity
; 13 close loop on counter mass once open loop velocity/voltage is satisfactory
; 14 activate PLC 12 when commanded velocity changes
; 15 close loop control of carriage motor
; 16 fast axis carriage open loop driver for start and break
; 17 carriage PD position controller
; 18 carriage centering program
; 19 galvo spacing controller/line counter

; 20 fast axis startup
; 21 fast axis stop
```

Appendix B: Variable initialization program

```
;this program is to be run once to load some machine variables

#include "macros"

I5 = 2                                  ;enable plc's
I15 = 1                                 ;calculate trig functions in radians

s = 0.0005                              ;set desired line spacing s (m)

n0 = 0                                  ;initialize counting variable to first positive pass
n1 = 0                                  ;initialize counting variable to first negative pass

L = 0.053                               ;galvo arm length (m)
theta = asin(0.0125/L)                  ;galvo neutral angle (rad) (make sure I15 = 1)
theta_conversion = 18774                ;conversion factor (program units per radian)
noz_off = L*cos(theta)+.025-.065        ;calculate nozzle offset
b = 0.120                               ;slurry bed dimension (m)
d = 0.105                               ;distance between spring and edge of slurry bed (m)
B = 0.005                               ;"sensor" distance B from edge of bed (m)
te = distance_trav/Vfs                  ;expected time = (distance traveled between lines)/(Vfs)
counter_0_elapsed = te                  ;actual time = expected time for first 0 pass
counter_1_elapsed = te                  ;actual time = expected time for first 1 pass

out_pos = 150                           ;initial DAC4 output (Volts*3276.8)
K1 = 0                                  ;gain for fast axis counter mass proportional control
K2 = 0                                  ;gain for fast axis carriage motor proportional control
percent = 2.0                           ;tolerance for fast axis voltage seeking plc (units=percents)

Vf_ave_s = 0.75                         ;initial commanded average velocity
Vf_run = 1                              ;initial running commanded average velocity
Kp = 10000                              ;set fast axis position proportional gain
Kd = 5000                               ;set fast axis position derivative gain
P101 = 0.09                             ;timing variable

Xs_i = -15000                           ;initial slow axis destination
Xs_f = -260000                          ;final slow axis destination
```

Appendix C: Slow Axis Motion Programs

```
#include "macros"

;********** Set-up and Definitions **********
&1 ; Coordinate System 1
CLOSE ; Make sure all buffers are closed
;#1->X ; Assign motor 1 to the X-axis - 1 program unit
; of X is 10^7 encoder counts of motor #1
;********** Motion Program Text **********
OPEN PROG 1   ; Open buffer for program entry, Program #1
CLEAR         ; Erase existing contents of buffer
#1J/          ; close servo loop
LINEAR ; Blended linear interpolation move mode
ABS           ; Absolute mode - moves specified by position
TA100         ; Set specified acceleration time in msec
TS0           ; Set no S-curve acceleration time
FRAX(X)
F(Vss*(10000000)) ; set feedrate (speed) in m/sec
P100=Xs_f
X(P100) ; Move X-axis to position 0
CLOSE         ; Close buffer - end of program
              ; To run this program:
              ; &1 B1 R ; Coordinate System 1, point to Beginning of
              ; Program 1, Run


#include "macros"

;********** Set-up and Definitions **********
&1 ; Coordinate System 1
CLOSE ; Make sure all buffers are closed
;********** Motion Program Text **********
OPEN PROG 2
clear
;enable plc 20                          ;begin startup plc
While (K1 = 0) wait                          ;if fast axis is in closed loop mode
        call1                                ;begin slow axis motion
while (abs(M101) < abs(Xs_f+100)) wait       ;when layer is done
        enable plc 21                        ;activate stopping plc
        Kp = 20000
        Kd = 7500
        enable plc 17
        homez1
close
```