DESIGN, ANALYSIS, AND CONTROL OF A BILL DESKEWING DEVICE FOR AUTOMATED TELLER MACHINES

by

Ross Bennett Levinsky

S.B., Aeronautical/Astronautical Engineering
Massachusetts Institute of Technology
(1989)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

January 1992

Signature of Author _____
Department of Mechanical Engineering
January 17, 1992

Certified by _____
Harry West
Thesis Supervisor

Accepted by _____
Ain A. Sonin
Chairman, Departmental Graduate Committee

Design, Analysis, and Control of a Bill Deskewing Device for Automated Teller Machines

by

Ross Bennett Levinsky

Submitted to the Department of Mechanical Engineering
on January 17, 1992 in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering

Abstract

A prototype device has been constructed to straighten banknotes that are skewed with respect to their direction of travel in an automated teller machine. The design requirements were high reliability, low hardware cost, low control complexity, and infrequent need for adjustment; they were met with a simple system consisting of four optical sensor pairs, two solenoids, and a tunable open-loop controller. To demonstrate the modeling of such a system, a simplified analysis of note handling by frictional contact is presented and numerically applied to the deskewing process. This modeling technique can be used to gain a qualitative understanding of how changing design parameters affect machine performance. A conceptually simple tunable open-loop controller, based on a lookup table, has been implemented to improve robustness of the deskewer by compensating for changing environmental conditions and note quality. It is shown that experimentally-observed constraints on the behavior of the deskewer allow a stable tuning algorithm to be developed; in addition, the simplicity of the controller concept allows a low-cost control circuit to be used. Experiments indicate that the machine reduces note skew to within design limits.

Thesis Supervisor: Dr. Harry West
Title: Assistant Professor of Mechanical Engineering

To My Parents

Acknowledgements

I am indebted to Harry West for the opportunities to work on this project and to travel to Japan. His counsel on matters technical and not-so-technical was invaluable. I am also grateful to the Omron Corporation for funding what was initially an open-ended, speculative project through to its current level of maturity.

Many thanks go to my friends and fellow engineers at Omron: Hiroyuki Nishimura (the motivating force for this project at Omron), Ryuichi Onomoto (who taught us about note handling and collaborated in the design of the deskewer), Ichiro Kubo (who explained the sensor circuits used in Omron's money-handling equipment), and Yoshimasa Sugitate (who designed and helped in the analysis of a new note feeder). They made both my work on this project and my stay in Japan tremendously positive experiences. I look forward to working (and playing) with them for many years to come.

Jack Kotovsky, who was a member of the design team and who built the prototype for his bachelor's thesis, did a fine job. His work has withstood many hours of experimentation and abuse. Henry Dotterer followed Jack on this project and, on his own, did the first experiments on learning algorithms (wholly separate from those described in this thesis). Matthew Selick, provided assistance in coding, guidance through the maze of Turbo C, and ceaseless help in divining the mysteries of interrupts on the IBM PC. Without him, the deskewer control would never have reached its current level of sophistication. Nathan Delson also made his experience with interrupts available to me at the beginning of this work.

Thanks to the many people who made easier what was occasionally a difficult experience; to my friends, for listening to me complain about note-handling and for accompanying me on forays into the world of late-night food in Boston; to Peter Robeck, for his time, patience, inspiration, and friendship; and most of all, to my parents, for their love, support, and continual admonishments to "Finish that thesis!"

Table of Contents

List of Figures

Chapter 1
Introduction

1.1 Historical Background

Since 1969, the Omron Corporation of Japan has designed and produced various types of cash-handling equipment, from change dispensers to automated teller machines (ATM). The technologies used have been experimentally tuned to a very high level of sophistication over time; by 1990, Omron's most advanced model, the HX-ATM, reliably operated at a transfer rate of ten notes per second.

During the late 1980's, though, engineers at Omron became convinced that they needed to expand their analytical knowledge of the note-handling process in order to increase transfer speeds and improve reliability. To reduce the need for expensive human service, they also wanted to begin incorporating self-adjusting mechanisms and mechanical error-correction systems into their designs.[1] With those goals in mind they entered into a research project with the Mechatronics Design Lab of the Massachusetts Institute of Technology.

1.2 Basic Technology of Bill Counting

The bill-handling technology used by Omron is similar to that employed by other major Japanese manufacturers of such products (Hitachi, Fujitsu, Oki, and Toshiba). In contrast to ATM in America, where cash to be deposited is first placed in an envelope, ATM in Japan directly input bills from the customer. The notes are slid from the top of the deposit stack by a rotating feed roller (see Figure 1.1) and carried through the machine by rubber transport belts that "sandwich" the notes. Through a complicated series of belts and sensors the bills are validated, counted, and stacked in cartridges during a deposit cycle, and fed, again using rotating-feed-roller technology, out of the cartridges during cash withdrawal. While the machines examined during this project were built for the Japanese market (and were thus sized

[1]Information about Japan's ATM market, Omron's machines, and Omron's perceived need for analytical models came from a series of discussions with key Omron personnel over a two-year period. These people included, but were not limited to, Hiroyuki Nishimura, Ryuichi Onomoto, Ichiro Kubo, and Yoshimasa Sugitate.

to handle yen), Omron uses near-identical technology to produce cash-handling equipment for other Asian countries.

This principal failure mode in Omron's machines is jamming. Typically, a note becomes stuck at a fork in the travel path while the notes behind it continue to move, causing one or more bills to crumple before the machine's optical sensors detect the problem and stop the transport mechanism. The exact causes of failure are difficult to pinpoint because the failure rate is so low - the only statistics Omron has are taken from in-service operation. It is has not been possible to observe enough failures in the controlled factory environment to adequately determine all causes of jamming.

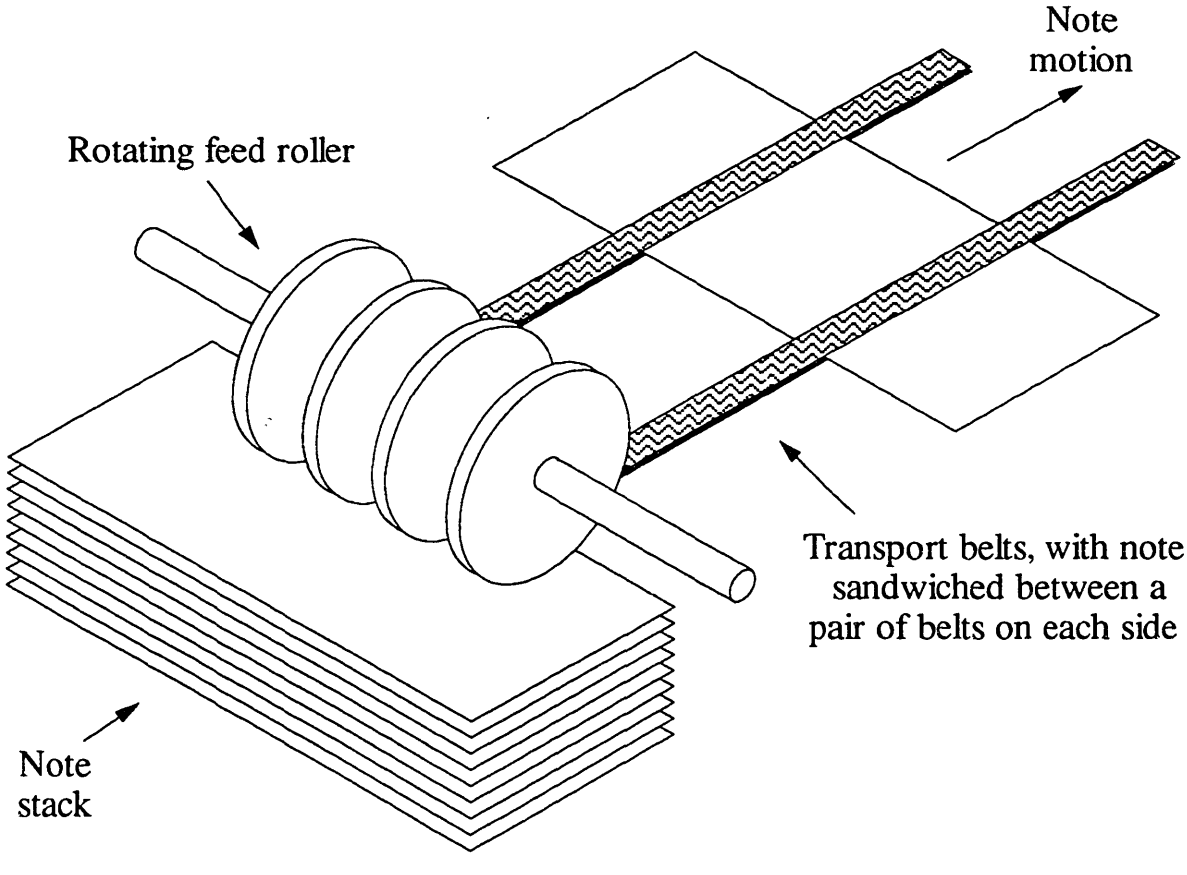


Figure 1.1: Simplified Sketch of Feed Roller

1.3 Skew

Omron did possess enough statistical information about jamming, however, to indicate that a phenomena known as "skew" was the most common cause of all jamming incidents. It was

therefore decided that the initial approach to improved machine reliability would be to correct skewed notes in transit. Skew (see Figure 1.2) is defined as a note's angular deviation from the correct travel position in which its short side is parallel to the direction of belt travel. The spacing between a skewed bill and its two surrounding notes (leading and trailing) is incorrect; this can cause jamming at forks in the travel path.

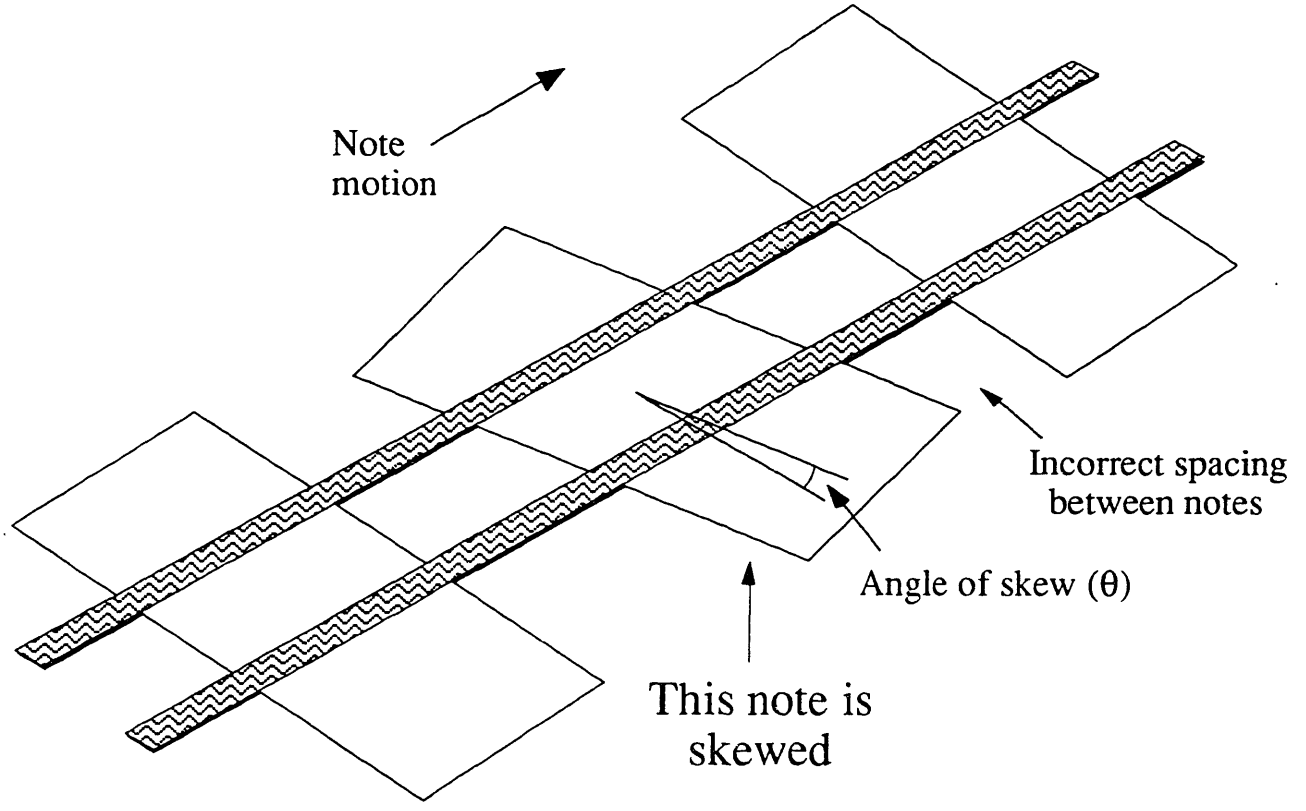


Figure 1.2: Definition of Skew

Earlier research in this project investigated the origins of skew. It has been found that skew occurs at the very beginning of the feeding process, when the note is accelerated by the slipping frictional contact of the feed roller. If a note enters the roller with one corner leading (slightly skewed), the frictional forces are initially applied at that corner, causing a torque that rotates the bill and increases its skew angle. This effect has been predicted by numerical simulation (see Appendix C) and confirmed with high-speed video. In addition, if the feed roller does not exert even pressure at all points along its axis, the note skews. This occurs because the frictional feeding force on a given area of the note's surface is proportional to the pressure that the roller applies at that area; when the pressure is not uniformly distributed along the roller's axis, a net moment is exerted on the note. This phenomena is exploited during

machine maintenance when technicians perform feed-roller pressure adjustments based on a machine-accumulated record of each deposited note's skew as it exited the feeder. An asymmetrical distribution of skew indicates that the roller pressure is uneven.

1.4 Design Considerations

During the course of discussions with Omron, four major design requirements were formulated for the deskewer (the working name for the skew-correcting mechanism). They were:

1. High reliability. The current generation of Omron machines is approaching a failure rate (where failure is defined as any problem requiring human intervention) of 1 in 5000 notes; addition of the deskewer should improve, not degrade, this figure. The machines are designed to accommodate skews of up to 8° without jamming, so the deskewer must be consistently capable of reducing skew to less than 8°.

2. Low cost. Although ATM are complex their profit margin is small; inexpensive sensors and actuators must be used.

3. Computationally-simple control algorithm. The CPU is already severely taxed in present-generation machines, so it is desirable to design a deskewer that requires little computation to properly straighten notes.

4. Low need for adjustment. ATM are serviced infrequently by expensive trained personnel, so a long-wearing or self-adjusting mechanism is necessary.

Possible deskewer designs were classified into three broad types:

1. Geometric-constraint. A mechanism is incorporated that firmly grasps the note and deskews it, possibly while also continuing the note's travel through the machine. An example of this method, shown in Figure 1.3, is a scheme in which there is a section of the ATM that transports the note by motor-driven rollers (no rubber belts). By giving the rollers slightly different angular velocities, the note simultaneously deskews and continues its movement through the machine. This method is likely to produce excellent deskewing performance and low note wear but has the drawbacks of high inertia in the transport rollers and the need for active control of the motors.

2. Force application. In these methods the note is deskewed by the application of a force to its surface while it is being transported by the belts. The force can come from sources as

diverse as a directed airstream and a robot finger dragging on the top surface of the bill. The difficulties are that a compressor is required for schemes involving air; some form of force control is required for mechanical systems; and note wear may be high.

3. Hybrid Geometry/Force. These designs have a mechanism that stops a point of the note (geometric constraint) and allows the frictional force applied by the moving transport belts (force application) to deskew the note. Two examples are shown in Figure 1.3; the blocking rods descend into the travel path and straighten the note when its leading side collides with one of the rods, while the solenoids pin a single point on the leading side of the note and allow the rest of the bill to rotate about the pinning point. The advantages of these schemes are the low cost and low inertia of the stopping elements, and the simple (stopper is either on or off) control. Possible problems include note buckling or crumpling and high note wear.

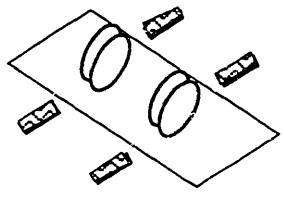
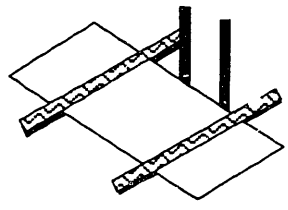
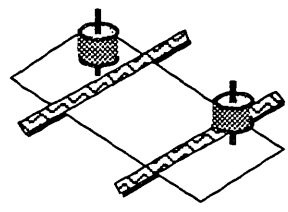
Designs / Characteristics	Active rollers 	Blocking rods 	Solenoids 
Control Complexity	High	Low	Low
Note wear	Lowest	Highest (crumpling)	Moderate (belt sliding)
Actuator inertia	High	Low	Lowest
High need for adjustment?	Yes	No	No
Cost	High (motors & controllers)	Low	Low

Figure 1.3: Selected Design Options

Both the active roller geometric-constraint design, because it showed potential for both excellent performance and low note damage, and the two hybrid geometry/force methods (solenoids and blocking rods), because they were inexpensive and simple to control, were selected for detailed consideration. The hybrid methods were eventually viewed as more viable because of lower component costs than the geometric method and uncertainty as to whether the rollers in the geometric-constraint design could have their angular velocities precisely altered at

frequencies above 10 Hz (This figure is actually an underestimate for ten notes/second operation, because a roller's speed must be adjusted and stabilized in the interval between notes. This implies a minimum response frequency above 10 Hz.).

Discussion with Omron's engineers revealed that they had independently tried an error-correction scheme similar to that of the blocking-rod system and had found that the rods crumpled an unacceptably high percentage of bills; wrinkled notes did not have enough stiffness to withstand the stresses of suddenly impacting a barrier in the travel path. So, the solenoid deskewing technique was finally chosen as the concept to be implemented in the prototype machine.

1.5 The Deskewing Machine

A conceptual sketch of the deskewer prototype is shown in Figure 1.4. Detailed drawings are presented in Appendix B, while additional information on design alternatives and construction has been documented elsewhere [Kotovsky 1989].

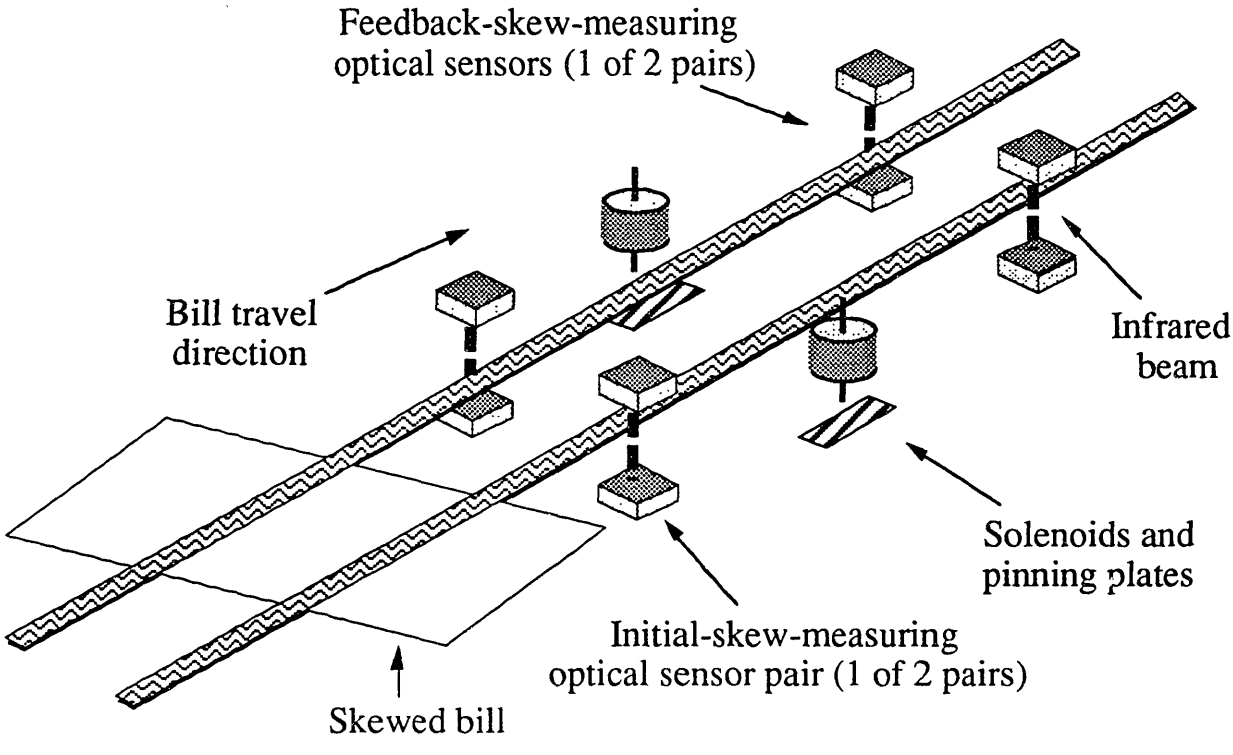
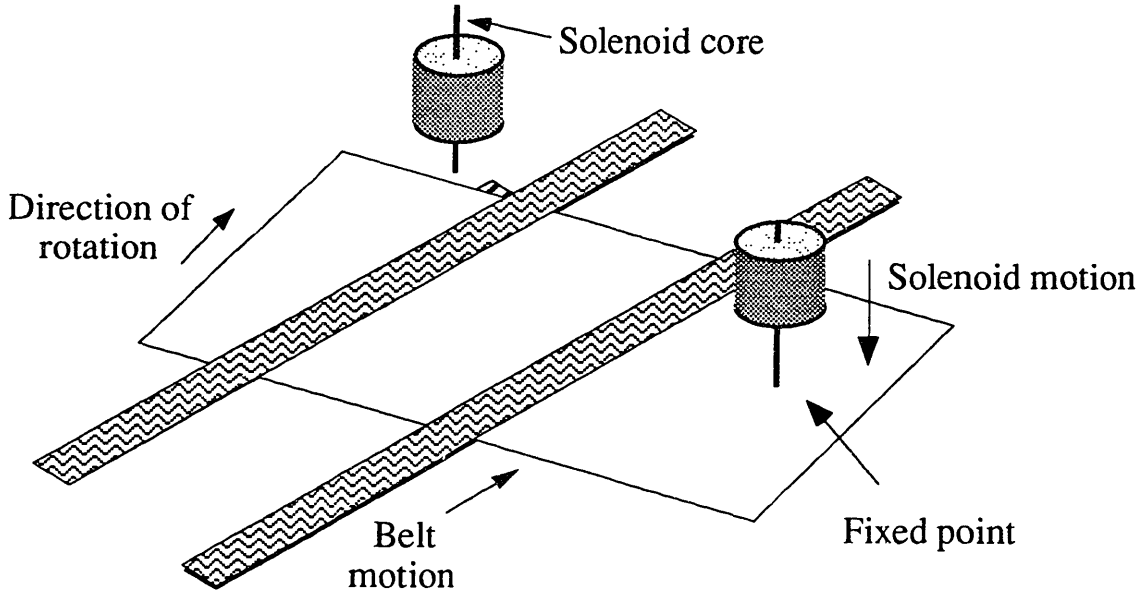


Figure 1.4: Simplified Conceptual Sketch of the Deskewing Machine

The note's skew is determined by measuring the time difference between obscuration of the left and right initial-skew optical sensor pairs, as detailed in Section 4.1. The measured skew, along with belt velocity (measured by a tachometer), is used to select a "wait time" and a "fire time" for the solenoids. The waiting period is programmed into a countdown timer that inhibits control action until the trailing half of the note is under the solenoid core (to reduce problems with buckling); at that time a second countdown timer containing the fire time begins decrementing and activates the solenoid, pinning the bill for the duration of the fire time. A very small area of the note is pinned against a steel plate (below the travel path) by the extended solenoid core so that the note is only free to rotate about the solenoid's contact point (Figure 1.5). The transport belts slip over the note's surface and generate a frictional force which rotates the note about the pinning point, thus reducing the skew. If the fire time has been properly selected the solenoid releases at the exact moment that the note is fully deskewed, and the note resumes its normal travel. Depending on note velocity and skew, fire time ranges from approximately 10 to 50 milliseconds.



Right-side solenoid fires and holds bill to pinning plate.
Belts on both sides continue to move, dragging bill about axis of right-side solenoid.

Figure 1.5: Sketch of Deskewing Action

1.6 Modeling and Control Problems

The detailed modeling of frictional-contact note-handling machines presents three fundamental difficulties. First, the Coulomb friction model (frictional force is proportional to normal force) is empirically derived and does not model the stick-slip behavior that can occur when a portion of the bill touches a surface such as the feed roller or the transport belts.

The second problem is the extreme variability in note quality encountered during actual operation. Notes are nonisotropic because they have usually been folded; at a crease they bend more easily in some directions than in others. This contrasts with the usual simplifying assumption that a note is an isotropic plate, and makes buckling analyses very difficult. This is a significant problem in sophisticated models of the deskewer because poor selection of the solenoid contact point has been experimentally observed to yield various forms of note crumpling; the behavior of each note depends on its overall condition (stiffness decreases with increasing wear) and particular pattern of creases.

Third, even with a considerably simplified system model the resulting note-motion equations involve complex multidimensional integrals with time-varying limits of integration, as shown in Chapter 2. These equations are usually solved numerically, and are thus principally used to examine the qualitative effects of the variation of design parameters on machine performance.

In addition to the basic difficulty of getting a plant model, control of the deskewer is challenging because the required fire time for a given skew can vary according to such factors as humidity, which is believed to change the note's mechanical properties, and belt wear, which affects both the coefficient of friction between belt and yen and the overall belt tension. This performance variability is not a problem if each note's skew is continuously measured and some type of individual-note closed-loop control effected, but such a system is difficult to implement because of the high speed (ten notes/second) of machine operation and the lack of an inexpensive sensor for continuous measurement of skew.

While a closed-loop controller offers good performance, it violates the low-cost and low-control-complexity design requirements; "closing the loop" on a particular note is much more difficult and costly than simple open-loop solenoid programming, in which the fire time is a function of a single measurement of incoming skew. In contrast, control hardware for the open-loop controller is simple and inexpensive, consisting of timers and low-cost optical sensor pairs that measure the incoming-note skew, and timers that hold the solenoid wait and

fire times. The drawback of the open-loop controller is that its fixed mapping between incoming-note skew and solenoid fire timing does not allow for changing deskewing times due to environmental, machine wear, and note wear effects.

Each control scheme has its advantages. An open-loop controller is desirable because of its computational simplicity and inexpensive sensors, while an individual-note closed-loop controller more effectively handles changing machine and note conditions. The ideal controller would combine the simplicity of open-loop with the robustness of closed-loop. The compromise that has been chosen is a tunable open-loop controller in which the open-loop control law is adjusted in response to measured deskewing performance. Feedback is not effected on any single note while it is deskewing, but a measurement of the success of the process, taken as the note is exiting the deskewer, is used to alter the control mapping and thus change the solenoid fire timing for all future notes. The implementation of such a scheme is explained in Chapters 3 and 4.

1.7 Thesis Overview

This thesis describes both the frictional-contact analysis and the self-tuning controller that have been applied to the deskewer. A simple analytical model of the system is presented in Chapter 2, along with sample computer solutions. The concepts underlying the tunable controller and a derivation of the tuning rule are explained in Chapter 3. Results of the implementation of the controller on the prototype deskewing hardware are given in Chapter 4. Chapter 5 summarizes the project and points out directions for further research.

Appendix A gives the dimensions and mass of the ¥10000 note, which is the most commonly handled note in Japanese ATM. Appendix B has more detailed views of the deskewer, along with parts lists. Appendix C applies the analytical methods developed in Chapter 2 to a model of the rotating-roller note feeder. A step-by-step procedure for utilizing interrupts on the IBM-PC and PC/AT is set forth in Appendix D. Appendix E lists the deskewer control program.

Chapter 2
Analysis of the Deskewer

2.1 General Case of Coulomb-Frictional Contact

The fundamental model for many bill-handling tasks is that of a machine surface interacting with the note through frictional contact. This is the case in the feeding process, in which the feed roller slips with respect to the note's surface and accelerates the bill into the travel belts. It is also the reason that the deskewer is able to straighten bills; the belts slide over the yen and exert a moment which rotates the note about the solenoid's pinning point. Because sliding frictional contact is so fundamental to the note-handling process it is useful to consider the general case of Coulomb-frictional contact (in which the frictional force is linearly proportional to the contact force).

Consider a small sliding plate in contact with a moving surface, as in Figure 2.1. The plate can be viewed as a differential element of the yen, while the moving surface can be considered a sliding belt or rotating feed roller. Coulomb's Law of Friction describes the magnitude of the frictional force as proportional to the normal force between plate and surface (with proportionality constant μ, the coefficient of friction) and the direction of the frictional force as that of the relative velocity between plate and surface (Figure 2.2):

$$\text{Magnitude of frictional force} = |\mathbf{F_f}| = \mu\,|\mathbf{F_n}| \tag{2-1}$$

$$\text{Direction of frictional force} = \mathbf{V_s} - \mathbf{V_p} = (V_{sx} - V_{px})\,\hat{\mathbf{i}} + (V_{sy} - V_{py})\,\hat{\mathbf{j}} \tag{2-2}$$

To get the complete expression for frictional force on the plate, the force direction vector is normalized into a unit vector. This leads to:

$$\text{Normalized } \mathbf{F_f} \text{ direction} = \frac{\mathbf{V_s} - \mathbf{V_p}}{|\mathbf{V_s} - \mathbf{V_p}|} = \frac{(V_{sx} - V_{px})\,\hat{\mathbf{i}} + (V_{sy} - V_{py})\,\hat{\mathbf{j}}}{\sqrt{(V_{sx} - V_{px})^2 + (V_{sy} - V_{py})^2}} \tag{2-3}$$

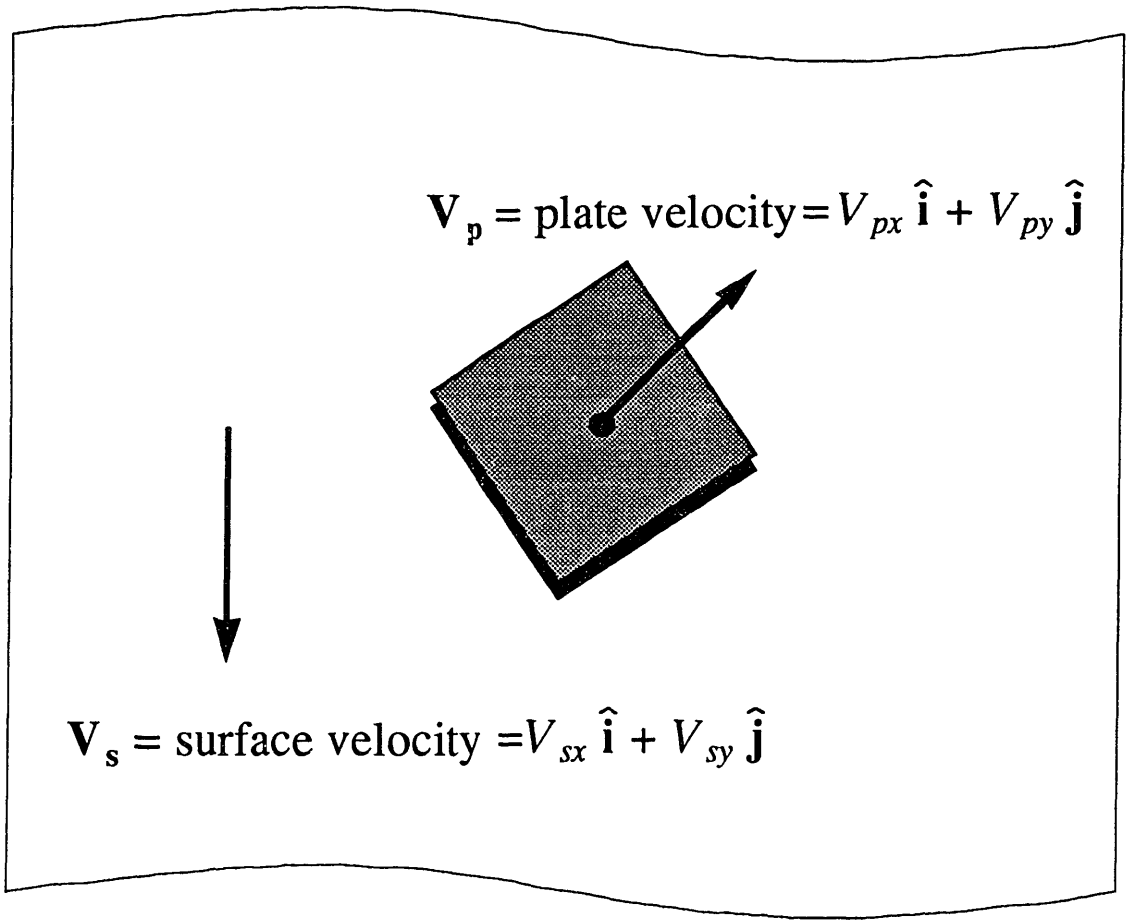


Figure 2.1: Small Plate Element Sliding on Moving Surface

The frictional force vector is simply the frictional force's magnitude multiplied by the unit direction vector:

$$\mathbf{F}_f = |\mathbf{F}_f| \cdot \frac{\mathbf{V}_s - \mathbf{V}_p}{|\mathbf{V}_s - \mathbf{V}_p|} = \mu |\mathbf{F}_n| \frac{(V_{sx} - V_{px})\hat{\mathbf{i}} + (V_{sy} - V_{py})\hat{\mathbf{j}}}{\sqrt{(V_{sx} - V_{px})^2 + (V_{sy} - V_{py})^2}} \tag{2-4}$$

Equation (2-4) is the general expression for sliding-friction force on a non-rotating element. A note can be thought of as many tiny elements connected together, each with a slightly different velocity vector (due to the entire note's rotation). If the entire region of contact between the note and the surface is integrated, the total force acting on the bill can be found, and the note's motion can be predicted. The surface's definition changes according to the specific analysis; for the feeder it is the feed roller area in contact with the note, while for the deskewer it is the belt area in contact with the note (neglecting the area pinned by the solenoid).

Appendix C analyzes a feed roller system using the same Coulomb-friction model, but with a slightly different derivation of the note's equations of motion.

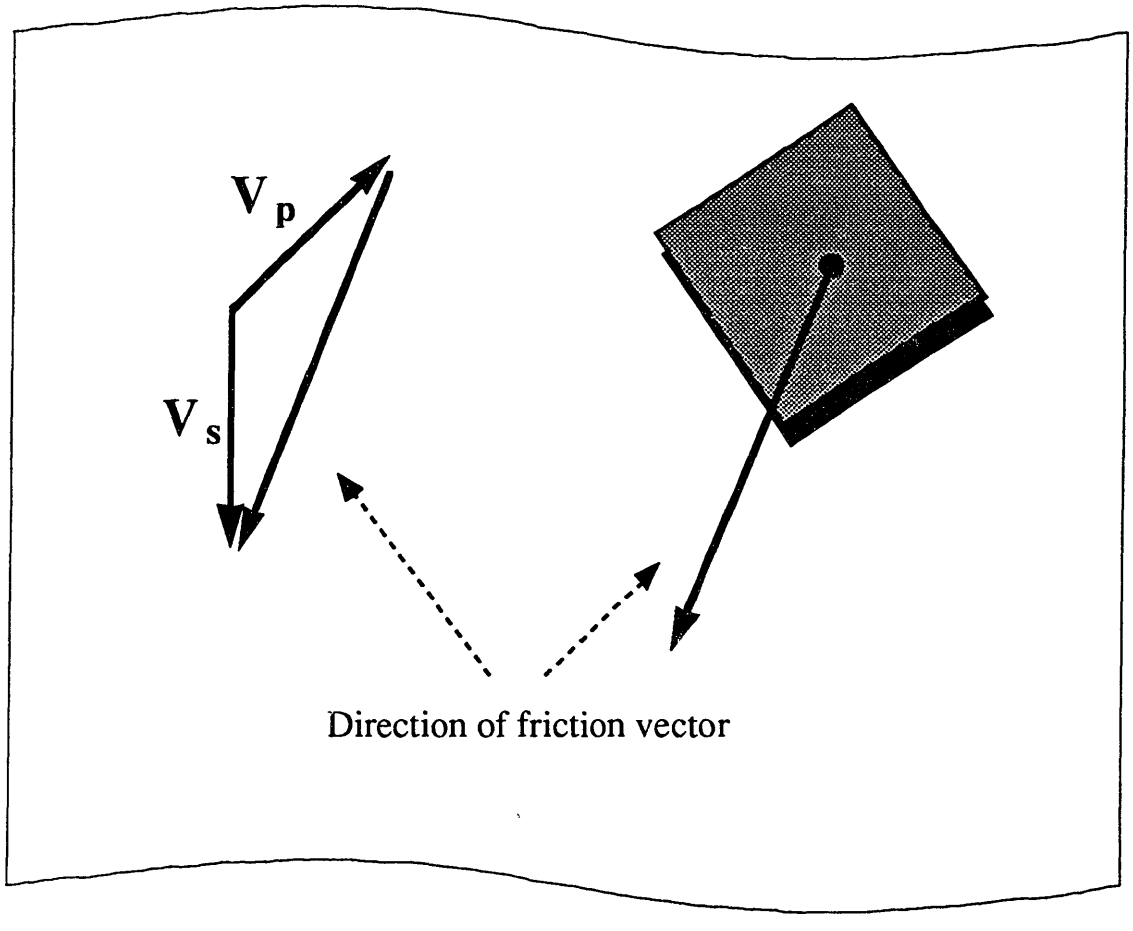


Figure 2.2: Direction of Friction Vector

2.2 Deskewer-Specific Equations

Figure 2.3 is a plan view of the deskewer prototype. Equation (2-4) can be applied by considering the moving belts to be the surface and the yen to be a connected collection of small plates. With the coordinate system oriented as shown, V_{sx}, the x-direction component of the surface velocity, is zero and V_{sy}, the y-direction component of the surface velocity, is set to V_b, the belt velocity:

$$V_{sx} = 0$$
$$V_{sy} = V_b \qquad (2\text{-}5)$$

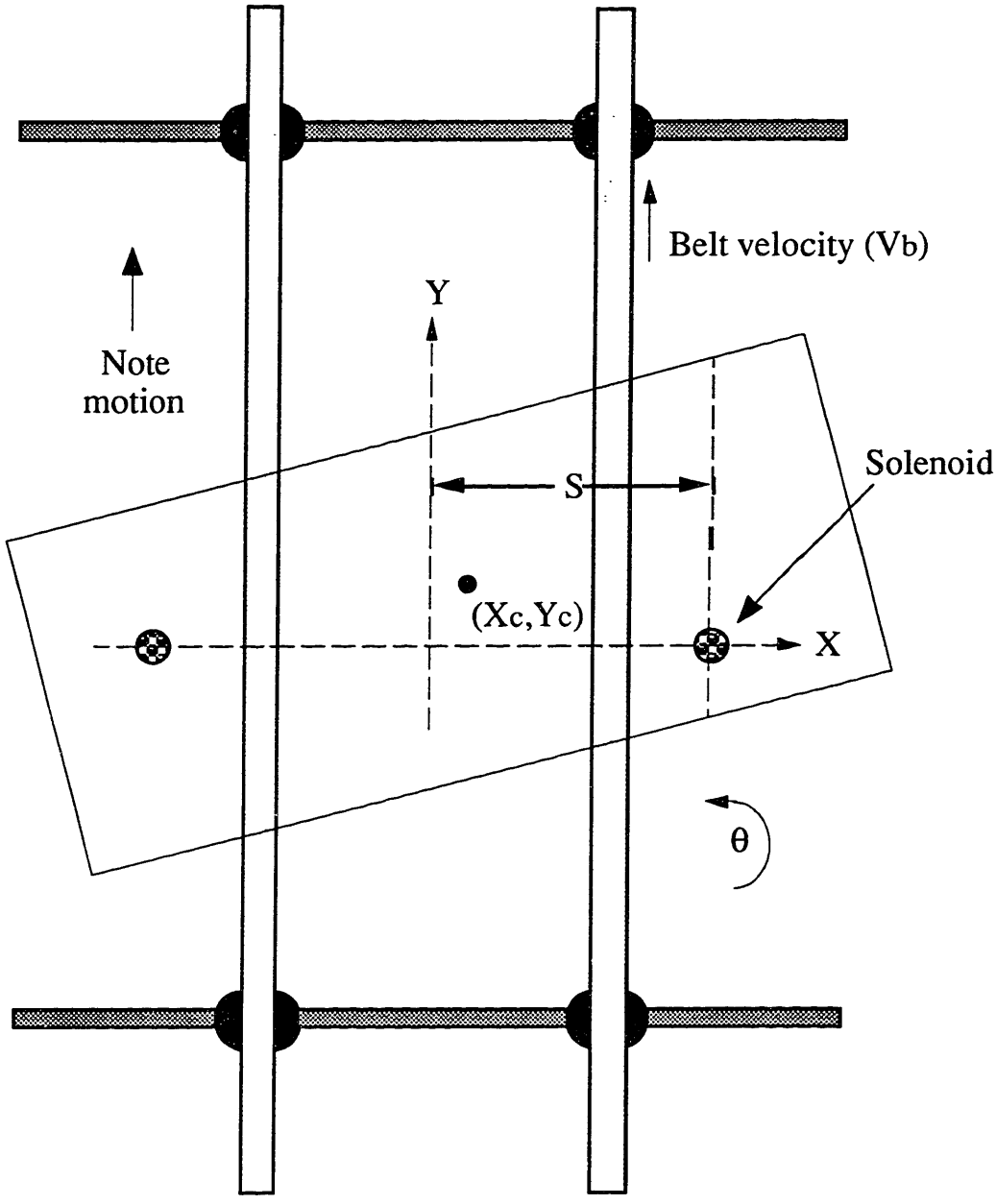


Figure 2.3: Plan View of Deskewing System

Because the note is pinned by the solenoid and rotates around the pinning point, it is convenient to express the note's surface velocity as a function of surface position and rotation rate about the pinning point. If ω (= $\dot{\theta}$) is the angular velocity of the note about the pinning point, (X, Y) is the position of an arbitrary point on the note's surface, and S is the solenoid displacement from the centerline of the travel path (as in Figure 2.3):

$$\text{Velocity of note point (X,Y)} = -\omega Y\hat{\mathbf{i}} - \omega(S-X)\hat{\mathbf{j}} \qquad (2\text{-}6)$$

Thus:

$$V_{px} = -\omega Y$$
$$V_{py} = -\omega(S\text{-}X) \tag{2-7}$$

Substituting into (2-4):

$$\text{Deskewer}\,\mathbf{F_f} = \mu\,|\mathbf{F_n}|\,\frac{\omega Y\,\hat{\mathbf{i}} + (V_b + \omega(S\text{-}X))\,\hat{\mathbf{j}}}{\sqrt{(\omega Y)^2 + (V_b + \omega(S\text{-}X))^2}} \tag{2-8}$$

The note is viewed as a connected group of small plates, so the normal force is more properly treated as a normal stress (assumed constant over a single plate due to its small size) multiplied by the area of a plate.

$$|\mathbf{F_n}| = \sigma\ (X,Y)\ dX\ dY \tag{2-9}$$

The standard differential equation for a rotating rigid body is now applied. I is the moment of inertia of the note when rotating about a given solenoid pivot-point position, and R, the region of integration, is the contact area between the belts and the note. It varies with time.

$$I\ddot{\theta} = \iint_R \mu\ \sigma(X,Y)\,\frac{\omega Y\,\hat{\mathbf{i}} + (V_b + \omega(S\text{-}X))\,\hat{\mathbf{j}}}{\sqrt{(\omega Y)^2 + (V_b + \omega(S\text{-}X))^2}}\ dX\ dY \tag{2-10}$$

The remaining difficulty is the prediction of the normal forces between the note and the belts. This problem is complex due to the high flexibility of the belts and the note, so for numerical investigations the normal stresses are generally assumed to be constant over the contact area.

2.3 Example Solution of Deskewer Equations

It can be seen immediately that (2-10) is difficult to solve analytically even with a simple, constant expression for σ, so a computer simulation has been written to numerically solve the equation. For simplicity, the program ignores buckling of the bill by assuming the note to be a rigid plate. First a spacewise integration is performed over the yen-belt contact area to determine the force on the bill, and then a time-domain integration is performed to find the motion of the bill over a small time interval. The fineness of the integration steps is then adjusted until consistent results are obtained. Predicted performance (in the form of necessary solenoid fire

time vs. input skew) has been obtained for a number of different machine configurations, and a sample result graph is shown below in Figure 2.4.

This graph displays the slight variation in hold times required for different positions, relative to the leading edge of the bill, of the solenoid pinning point. "P" refers to the normalized distance back from the leading edge; P=0 means the pinning point is at the leading edge and P=1 indicates that the pin occurs at the trailing edge. The lateral solenoid position (S) and belt speed (V_b) are fixed in all cases shown, with the solenoid position at 30 mm and the belt speed at 1 m/s. Note properties are those given in Appendix A.

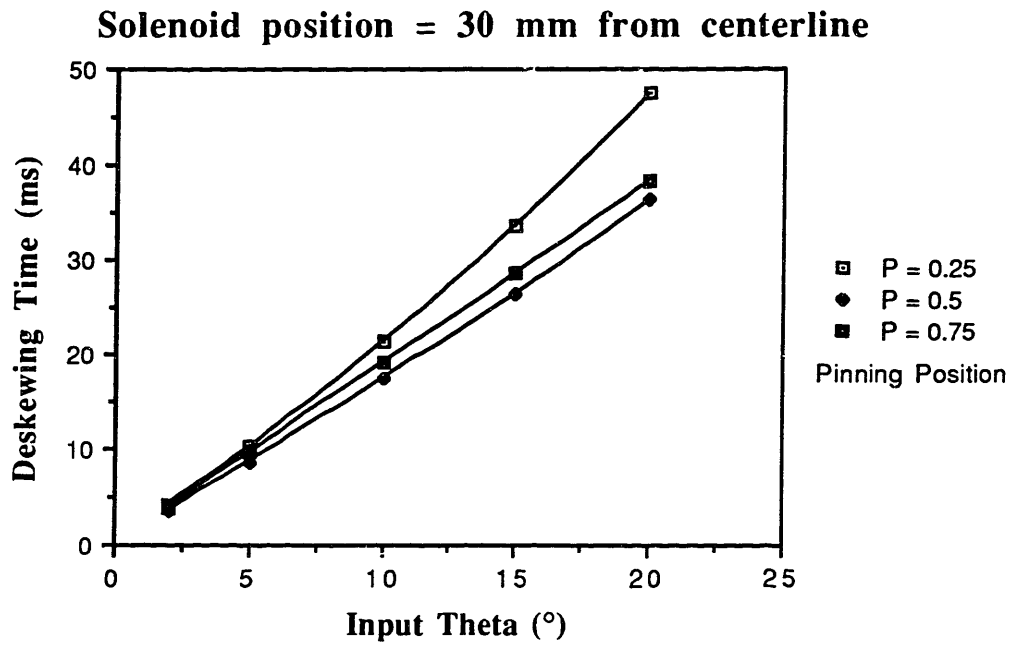


Figure 2.4: Simulation Results for Varying Pin Position (P)

Experimental evidence indicates that this graph is qualitatively correct. Pin positions of P<0.5 tend to require longer hold periods for adequate deskewing, while those towards the trailing edge need less time. As an additional advantage, the trailing-edge positions have been observed to cause less note buckling than those towards the leading edge. When the pin is located towards the rear the deskewing force provided by the belts acts to tension the bill, thus reducing buckling problems.

Figure 2.5 shows a graph similar to that in Figure 2.4, except that the pinning position is now fixed at P=0.75 while the lateral solenoid position (S) is varied from 30 mm to 65 mm. The simulation indicates that the required deskewing time is slightly affected by solenoid

position, with S=65 mm consistently requiring the most time. The biggest variation in timing occurs at an input skew of 20°, and is 4.4 milliseconds (between the 65 mm and the 50 mm positions). This effect is notable because the feeding process normally causes slight (up to approximately 10 mm) variation in the lateral position of notes. Two bills with identical input skews but horizontally-offset positions in the transport belts are entering two slightly different deskewing systems, distinguished by a difference in S. The notes thus require different deskewing times.

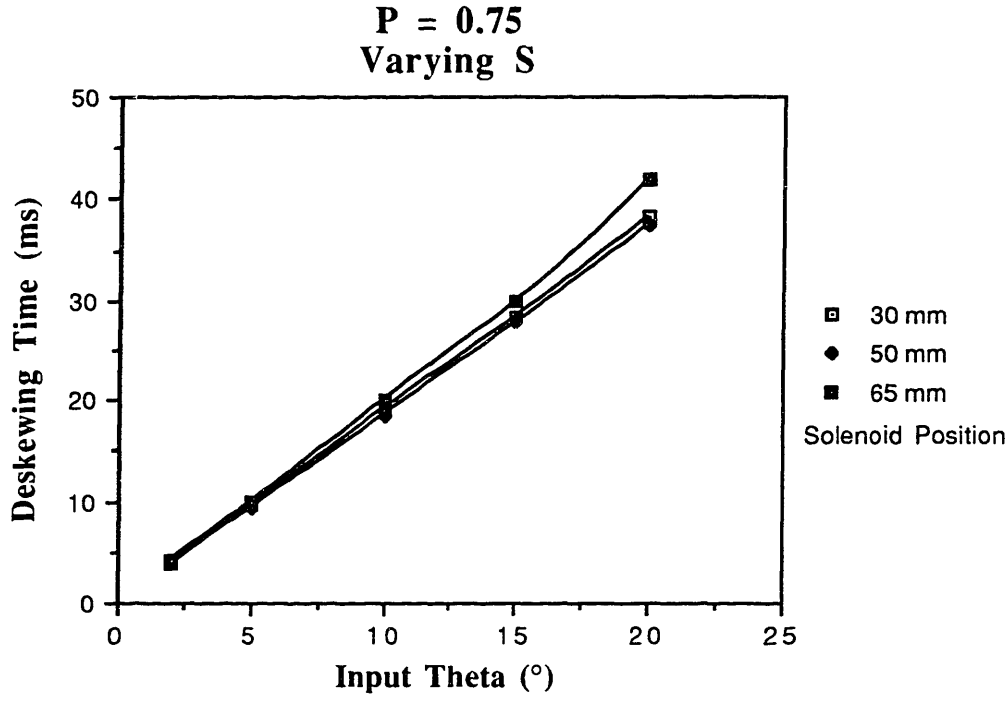


Figure 2.5: Simulation Results for Varying Solenoid Position (S)

Chapter 3
Control of the Deskewer

3.1 Mathematical Formulation of the Deskewer Control Problem

A very large class of systems can be described by equations of the form (3-1), where all terms are vector-valued.

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{x}) + \mathbf{B}(\mathbf{u}) \tag{3-1}$$

In such a system the fundamental control problem is to choose a control action **u** (as a function of time) that is guaranteed to cause the state vector **x** to converge to a desired trajectory. There may be other requirements involving convergence time or cost of control, but the basic necessity is that **u** yields a stable, convergent system.

If (3-1) is rearranged as follows:

$$\dot{\mathbf{x}} - \mathbf{A}(\mathbf{x}) = \mathbf{B}(\mathbf{u}) \tag{3-2}$$

an essential feature of this type of system can be seen - it has only one set of equations to describe its behavior, regardless of the control action. As long as the system's behavior stays within the bounds of the modeling assumptions, the function **u** does not change the overall form of the system equations. Different **u** will lead to different time evolutions of **x**, but the theoretical framework will remain that of (3-2).

The deskewer is fundamentally different from systems described by (3-2) in that the application of the control action (the firing of the solenoids) completely changes the form of the system equations. Again taking θ as the note skew and ω (= $\dot{\theta}$) as the note angular velocity about the pinning point, the system equations can be stated as follows. When no control action is applied (solenoids are inactive):

$$\begin{aligned} \dot{\omega} &= \omega = 0 \\ \theta &= \text{constant} \end{aligned} \tag{3-3}$$

When control action is applied (solenoids are fired), equation (3-4), which is a recasting of (2-10) in the form of (3-2), holds.

$$\dot{\theta} = \omega$$

$$I\dot{\omega} = \iint_R \mu \cdot \sigma(X,Y) \frac{\omega Y\,\hat{\mathbf{i}} + (V_b + \omega(S - X))\,\hat{\mathbf{j}}}{\sqrt{(\omega Y)^2 + (V_b + \omega(S-X))^2}}\, dX\, dY \qquad (3\text{-}4)$$

Figure 3.1 is a qualitatively-correct graph of θ versus time since measurement of skew. The programming of the deskewer is a question of when and how long to switch the system model from (3-3) to (3-4).

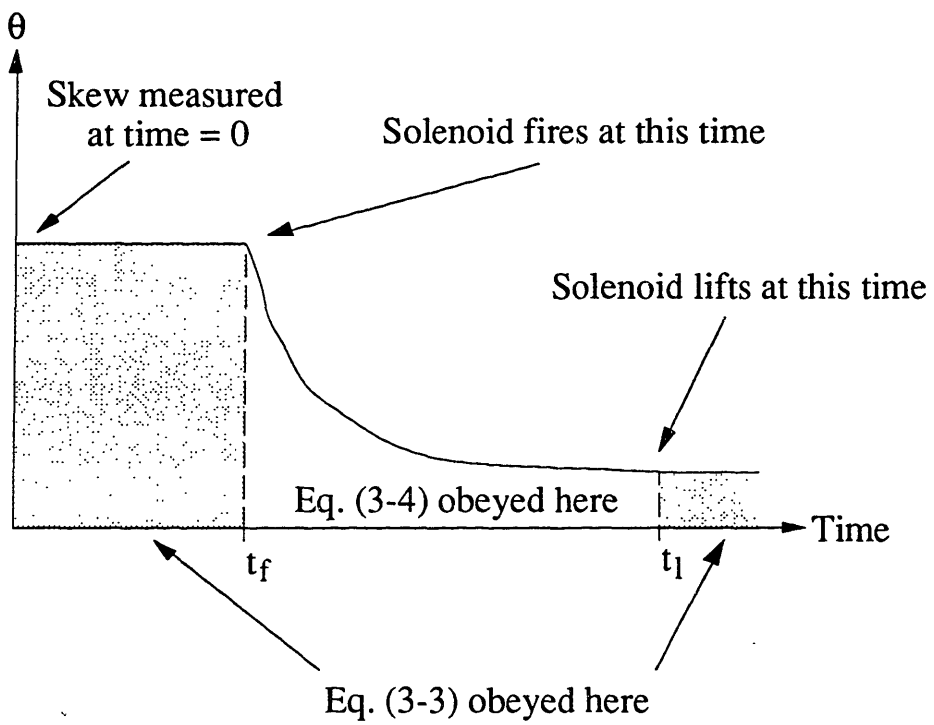


Figure 3.1: Sketch of Skew vs. Time

If a system of the form shown in (3-2) is controlled in a bang-bang fashion, its equations resemble those of the deskewer. The similarity is not exact, though, because a bang-bang controller is typically capable of both positive and a negative control action; if the task is position control of a motor, for example, the bang-bang controller is able to apply both positive and negative voltage (usually of the same magnitude). The deskewer is a "one sided" system in

that the control action is either on or off, with no possible negative action. An analogous system is that of a motor which has only one input voltage and an on-off switch.

3.2 Control Schemes

The time at which to switch to equation (3-4), which is the selection of the solenoid waiting time, can be calculated in a very simple fashion. As noted in Section 2.3, the bill experiences minimal buckling when the solenoid pinning point is located towards the note's trailing edge. For a desired pin location the correct wait time is easily calculated with knowledge of the distance between the skew-detecting sensors and the solenoids, the bill's skew, and the belt velocity (measured from a tachometer attached to one of the deskewer's rotating shafts). In practice the pinning point is generally located on the back 30-50% of the bill, where it causes no buckling problems.

Selection of the total solenoid firing time (t_l - t_f) is a more challenging problem for the reasons discussed in Section 1.6. There are many possible methods for selecting the proper timing, with the most basic distinction being that between closed-loop and open-loop control.

A closed-loop controller continuously measures the skew of the note while it is pinned and retracts the solenoid at the appropriate time. The advantage of this method is that it dynamically produces the correct deskewing times and thus accounts for machine wear, changing note quality, and varying environmental conditions without requiring a detailed understanding of how these factors affect deskewing performance. The disadvantage, as pointed out in Section 1.6, is the difficulty and cost of measuring skew continuously. While arrays of optical sensors could potentially give a continuous report of note skew, they are much more expensive than the simple phototransistor - infrared LED pairs that can be used to take a single measurement.

The open-loop scheme simply programs the solenoids on the basis of measured incoming skew and then ignores the note until after it has left the deskewer, at which point the outgoing skew may be measured for later reference; for example, in the tunable open-loop controller discussed below, the outgoing skew information is used as feedback data to adjust the solenoid fire timing. This method has an appealing simplicity, makes low demands on the skew sensors, and is straightforward to implement in hardware. However, the simplest open-loop controller, in which there is a static mapping between incoming skew and solenoid fire time, has the disadvantage of not adapting to changing note and/or environmental conditions. Exceptionally worn bills are treated exactly the same as fresh, crisp notes, and no distinction is made between humid and dry weather.

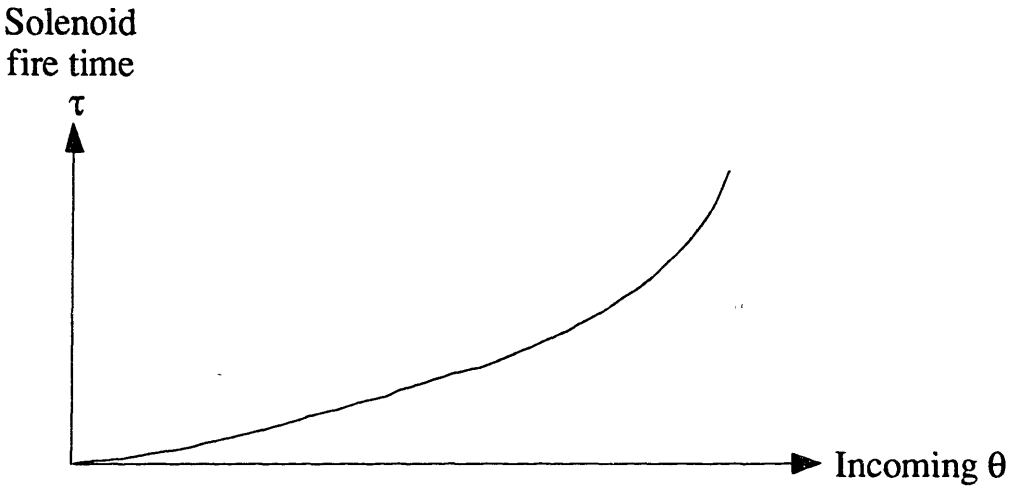


Figure 3.2: General Form of Skew-Timing Function

Thus, this project has focused on the creation of a control scheme that possesses both the hardware simplicity and low cost of an open-loop controller and the robustness of a closed-loop system; it is described as a tunable open-loop controller. It creates a mapping, similar to that depicted in Figure 3.2, which is altered in response to measurements of the deskewer's performance but which is not changed during the straightening of any particular note. The note's outgoing skew is recorded and is used to adjust the mapping after the note has exited. The general form of the controller is shown in Figure 3.3.

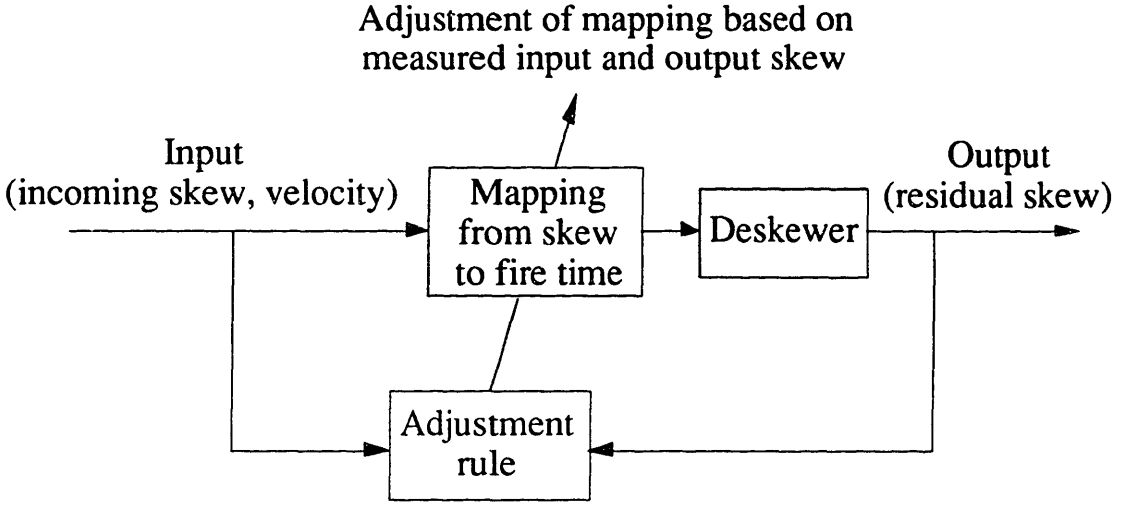


Figure 3.3: General Form of Tunable Open-Loop Controller

3.3 Creation of the Control Mapping

The question of how to implement the mapping from input skew to timing can be viewed as a problem in nonlinear function approximation. There are many methods to perform this task; some common schemes are neural networks, sums of elementary (basis) functions, and binning or partitioning methods, which can be thought of as lookup tables. In a system as simple as the deskewer (with only one variable to be controlled), it is convenient to stay with binning methods for two reasons.

First, they are simple, and thus inexpensive, to program in software and implement in hardware. While performing approximations with neural networks or basis functions may be more memory-efficient if the number of input variables is high (the storage cost of binning algorithms goes up geometrically with the number of system variables being considered), the networks and functions generally require repeated evaluations of transcendental functions. This requires more computational power than a lookup table, which only needs to access memory to find a control value, and leads to greater hardware cost. The choice of a lookup algorithm is complementary to the goal of a low-cost system.

Second, it is shown below that experimentally-observed constraints on the behavior of the deskewer can be used to find a theoretically-convergent algorithm for adjusting the timing values in the lookup table. The deskewer is more complex than is assumed in the derivation of the timing adjustment rule, so its convergence behavior departs from the theoretical prediction, but the tuning method is sufficiently stable to provide acceptable deskewing.

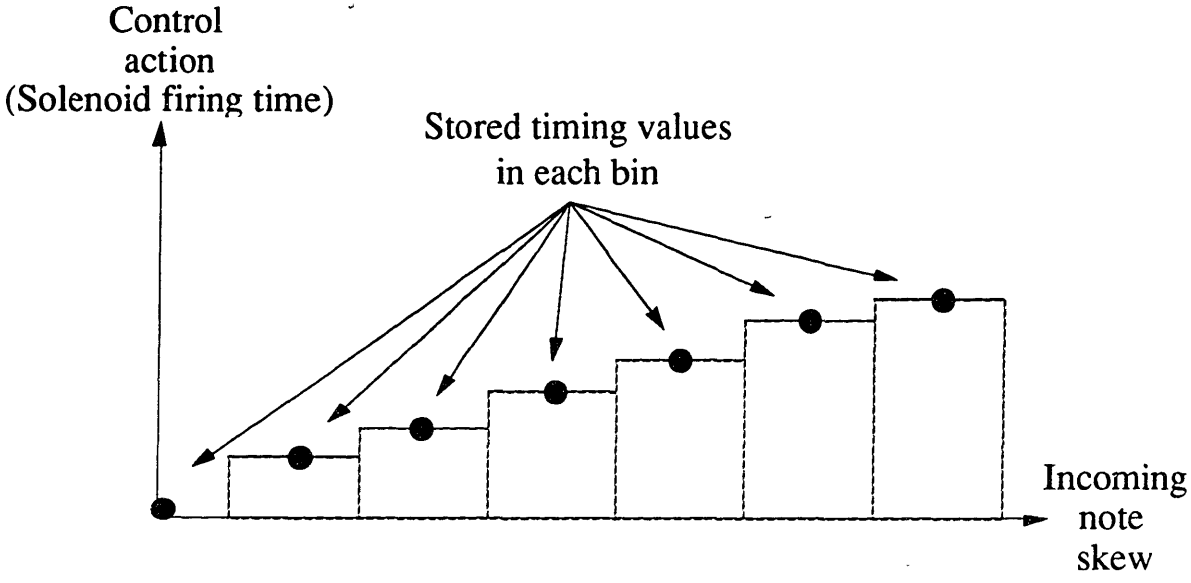


Figure 3.4: Initial Note Skew is Binned

The process of selecting a control action is straightforward. First the state space (which is comprised of incoming skew measurements) is divided into ranges or "bins" as in Figure 3.4, and a solenoid firing time is associated with or "contained" in each bin. The initial value in each bin may be assigned through experiment or the use of results from a numerical simulation of the deskewer. When a new measurement of incoming skew is made, the value of the control action to be applied is taken from the measurement's bin.(Figure 3.5).

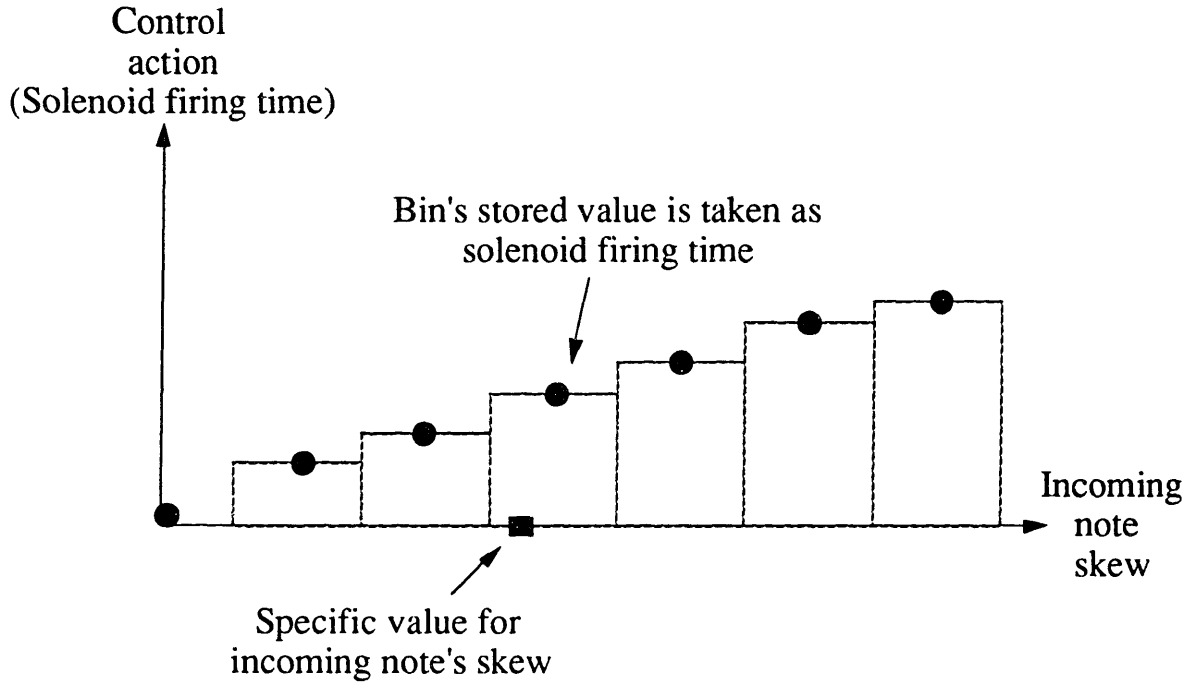


Figure 3.5: Control Action Comes from Bin

After the corresponding output skew value is recorded, the validity of the control action associated with the bin can be judged. The tuning of each bin's control action comes from examination of the measured output skew and application of a suitably chosen update rule, described in Section 3.5

3.4 Qualitative Behavior of the Deskewer

Given the following seven assumptions, all based on observations of the qualitative behavior of the deskewer, convergence of a table-tuning algorithm can be demonstrated. First, though, notation is needed. A *control cycle* consists of a skewed note entering the deskewer, correction of the note, measurement of the residual skew, and adjustment of the solenoid timing. For control cycle j, θ_{in_j} is taken as the incoming skew, τ_j as the solenoid fire time for

that cycle, θ_{out_j} as the outgoing skew, and $\Delta\tau_j$ as the adjustment solenoid fire time. When the "j" subscript is omitted in an equation, the expression is a general description of the deskewer's behavior and does not refer to a particular control cycle.

All of the following analyses assume that the deskewer is operating at a constant note throughput, with no variation in velocity.

1. The output skew for a control cycle is a continuous function of input skew and applied solenoid fire time:

$$\theta_{out_j} = F\left(\theta_{in_j}, \tau_j\right) \tag{3-5}$$

Equation (3-5) is the most problematic of the assumptions used to derive the tuning rule. In actuality, the output skew of a note also depends on such factors as variations in machine velocity and note condition. These effects are addressed in Section 3.8.

2. At any input skew, greater solenoid firing time leads to reduced output skew:

$$\left.\frac{\partial F}{\partial \tau}\right|_{all\ \theta_{in}} < 0 \tag{3-6}$$

3. At all values of θ_{in} and over all values of τ at which the deskewer can operate (there is an upper limit on τ because the note must clear the deskewing section in time to avoid jamming the following note), a finite change in τ leads to a finite change in θ_{out}. Thus:

$$0 < \alpha \leq \left|\frac{\partial F}{\partial \tau}\right| \leq \beta \tag{3-7}$$

4. At any fixed value of τ, a greater input skew leads to a greater output skew. Like (3-7) this phenomenon is bounded:

$$\left.\frac{\partial F}{\partial \theta_{in}}\right|_{all\ \tau} > 0 \tag{3-8}$$

$$0 < \gamma \leq \frac{\partial F}{\partial \theta_{in}} \leq \delta \tag{3-9}$$

5. Solenoid timing is a continuous function of the input skew and the desired final output skew. However, since the desired θ_{out_j} is always 0°, the expression for τ_j does not explicitly contain θ_{out_j}. $G(\theta_{in_j})$ is thus the solenoid fire time that yields $\theta_{out_j} = 0°$:

$$\tau_j = G\left(\theta_{in_j}\right) \tag{3-10}$$

6. Solenoid firing time is always greater than or equal to zero:

$$G \geq 0 \tag{3-11}$$

7. Greater input skews require longer firing times to correct:

$$\frac{dG}{d\theta_{in}} > 0 \tag{3-12}$$

When (3-5) through (3-12) are considered together, graphs for F and G are seen to be qualitatively similar to Figures 3.6 and 3.7.

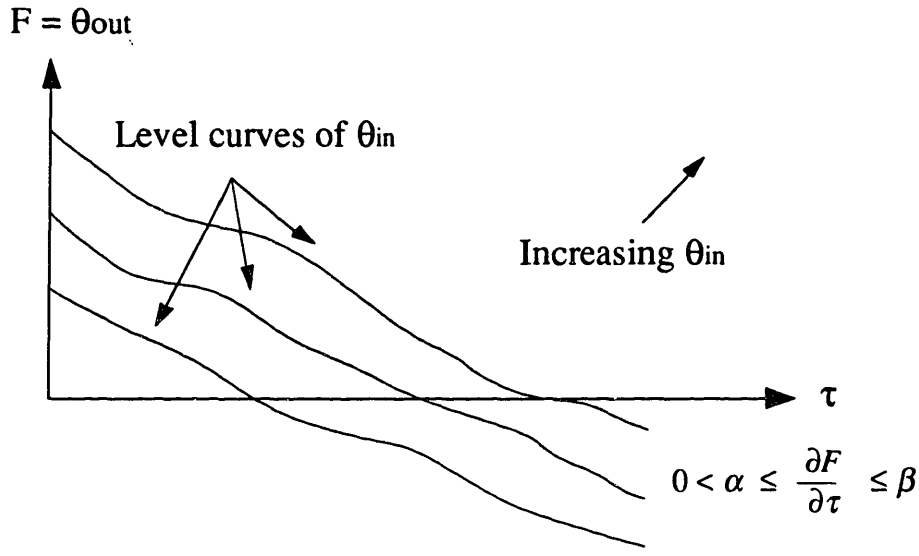


Figure 3.6: General Form of $F(\theta_{in},\tau)$

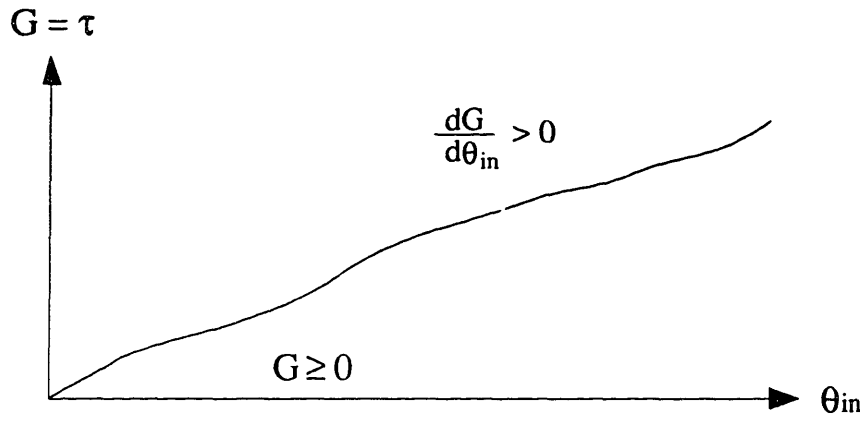


Figure 3.7: General Form of $G(\theta_{in})$

3.5 Convergence at One Input Skew

When the control mapping is stored in a lookup table, the graph of G is discretized by dividing the θ_{in}-axis into bins and associating a single value of G with each bin's range of θ_{in}, as in Figures 3.4 and 3.5. Figure 3.8 displays a typical bin: θ_{in_-} is the smallest value of θ_{in} that falls in the bin; θ_{in_+} is the largest value of θ_{in} in the bin; τ_- is the value of $G(\theta_{in_-})$; τ_+ is the value of $G(\theta_{in_+})$; and τ_{bin} is the control value stored in the bin.

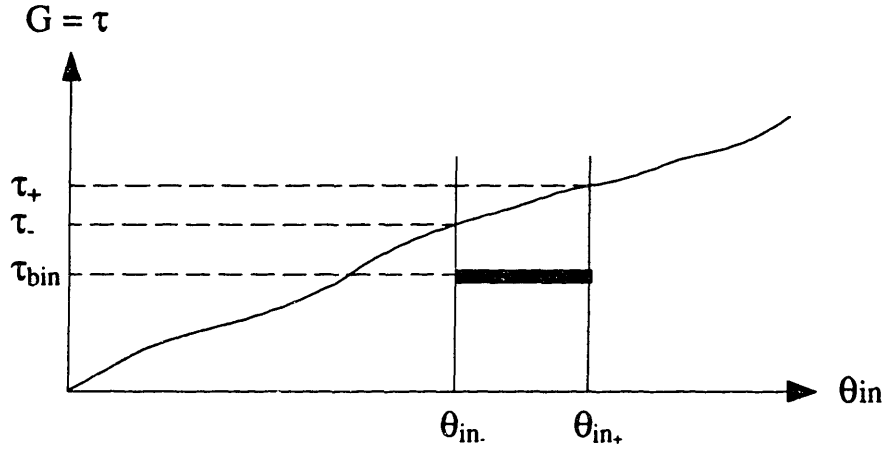


Figure 3.8: Single Bin of $G(\theta_{in})$

If a series of notes with a single value of input skew, $\theta_{in_{repeated}}$, is fed into the deskewer, it is now possible to show that τ_{bin} (for the bin in which $\theta_{in_{repeated}}$ falls) converges to the value of $G(\theta_{in_{repeated}})$ if the following algorithm is used for adjusting τ:

$$\Delta\tau_j = \frac{\theta_{out_j}}{\beta} \tag{3-13}$$

where β is the upper bound on the magnitude of $\left|\frac{\partial F}{\partial \tau}\right|$ as given in (3-7). If θ_{out_j} is greater than 0°, (3-13) indicates that $\Delta\tau_j$ will be positive. Since τ_{j+1} is then larger than τ_j, (3-6) implies that $\theta_{out_{j+1}}$ will be smaller than θ_{out_j}. Combining this information with the bounds on $\left|\frac{\partial F}{\partial \tau}\right|$ given in (3-7) gives:

$$0 < \alpha \le \frac{\theta_{out_j} - \theta_{out_{j+1}}}{\Delta\tau_j} \le \beta \tag{3-14}$$

Equation (3-14) is a consequence of the mean value theorem of calculus; if it is false and $\frac{\theta_{out_j} - \theta_{out_{j+1}}}{\Delta\tau_j}$ is greater than β or less than α, there exists a τ between τ_j and τ_{j+1} at which $\left|\frac{\partial F}{\partial \tau}\right|$ is also greater than β or less than α, respectively. This violates the bounding given in (3-7), so (3-14) must be true.

α and β must be conservatively estimated. For example, if β is larger than the true least-upper-bound (l.u.b.) of $\left|\frac{\partial F}{\partial \tau}\right|$, the tuning scheme will still converge (albeit more slowly than if β were equal to the l.u.b.). If β is smaller than the l.u.b. or α is larger than the true greatest lower bound of $\left|\frac{\partial F}{\partial \tau}\right|$, (3-14) is no longer guaranteed to hold and the convergence relations given below are invalid.

Rearranging (3-14):

$$\theta_{out_j} - \beta\Delta\tau_j \leq \theta_{out_{j+1}} \tag{3-15}$$

$$\theta_{out_j} - \alpha\Delta\tau_j \geq \theta_{out_{j+1}} \tag{3-16}$$

Substituting (3-13) into (3-15) and (3-16):

$$\theta_{out_{j+1}} \geq 0 \tag{3-17}$$

$$\theta_{out_j}\left(1 - \frac{\alpha}{\beta}\right) \geq \theta_{out_{j+1}} \tag{3-18}$$

(3-18) implies that $\theta_{out_{j+1}}$ is always less than θ_{out_j}, because $0 < \frac{\alpha}{\beta} < 1$. If θ_{out_0} is the output skew of the first note in the series of notes with identical input skew, (3-18) yields (by recursion):

$$\theta_{out_0}\left(1 - \frac{\alpha}{\beta}\right)^n \geq \theta_{out_n} \tag{3-19}$$

The τ-tuning formula of (3-13) thus yields bounding values on θ_{out} for a string of notes with a single input skew. Combining (3-17) and (3-19) gives the following inequality:

$$\theta_{out_0}\left(1 - \frac{\alpha}{\beta}\right)^n \geq \theta_{out_n} \geq 0 \tag{3-20}$$

(3-20) shows that θ_{out} can be made arbitrarily close to 0° by increasing the number (n) of notes fed through the machine. The assumption that $\theta_{out_j} > 0°$ is not necessary; if the preceding analysis is repeated assuming $\theta_{out_j} < 0°$, it yields:

$$\theta_{out_{j+1}} \leq 0 \tag{3-21}$$

$$\theta_{out_j}\left(1-\frac{\alpha}{\beta}\right) \leq \theta_{out_{j+1}} \tag{3-22}$$

$$\theta_{out_0}\left(1-\frac{\alpha}{\beta}\right)^n \leq \theta_{out_n} \leq 0 \tag{3-23}$$

Equations (3-20) and (3-23) show that if τ is tuned according to (3-13), θ_{out} converges to 0° for a string of notes with a single input skew. Assuming that $\tau_{bin} < G(\theta_{in_{repeated}})$, the bounds on $\left|\frac{\partial F}{\partial \tau}\right|$ provided by (3-7) and the mean value theorem give:

$$0 < \alpha \leq \frac{\theta_{out_j}}{G(\theta_{in_{repeated}}) - \tau_{bin}} \leq \beta \tag{3-24}$$

Upon rearranging, (3-24) shows that the distance from τ_{bin} to $G(\theta_{in_{repeated}})$ can be made arbitrarily small (although τ_{bin} will always be less than $G(\theta_{in_{repeated}})$):

$$0 < G(\theta_{in_{repeated}}) - \tau_{bin} \leq \frac{\theta_{out_j}}{\alpha} \tag{3-25}$$

Substituting in (3-20):

$$0 < G(\theta_{in_{repeated}}) - \tau_{bin} \leq \frac{\theta_{out_0}}{\alpha}\left(1-\frac{\alpha}{\beta}\right)^n \tag{3-26}$$

If it is now assumed that $\tau_{bin} > G(\theta_{in_{repeated}})$, each θ_{out_j} is negative. This implies:

$$\frac{\theta_{out_j}}{\alpha} \leq G(\theta_{in_{repeated}}) - \tau_{bin} < 0 \tag{3-27}$$

$$\frac{\theta_{out_0}}{\alpha}\left(1-\frac{\alpha}{\beta}\right)^n \leq G(\theta_{in_{repeated}}) - \tau_{bin} < 0 \tag{3-28}$$

Thus τ_{bin} converges to $G(\theta_{in_{repeated}})$ regardless of whether it is initially larger or smaller than $G(\theta_{in_{repeated}})$. Equations (3-26) and (3-28) show that τ_{bin} always stays on the side of $G(\theta_{in_{repeated}})$ from which it came.

3.6 Multiple Input Skews in One Bin

The results of the previous section are valid only for a series of notes with a single input skew. In general this is not the case; the notes may all fall within a specific interval (and perhaps even in a single bin) but they usually have varying input skews. The results of Section 3.5 are here extended to show that τ_{bin} converges to either τ_- or τ_+, or eventually enters the interval $[\tau_-, \tau_+]$ if enough bills with input skew in the open interval $(\theta_{in_-}, \theta_{in_+})$ are run through the deskewer.

The first case is $\tau_- < \tau_{bin} < \tau_+$; the graph of G intersects the constant line τ_{bin} at some θ_{in} in the interval $[\theta_{in_-}, \theta_{in_+}]$. The largest possible increase in τ_{bin} occurs if the next incoming note is skewed at θ_{in_+}, but even in that extreme case equation (3-26) guarantees that $\tau_{bin} + \Delta\tau$ will always be less than or equal to τ_+. If $\tau_{bin} + \Delta\tau$ and τ_+ become equal, by (3-9) any immediately-succeeding incoming note that falls within the bin and is less than θ_{in_+} will yield a negative $\Delta\tau$ and will thus decrease τ_{bin}, and any note with input skew of θ_{in_+} will leave τ_{bin} fixed at τ_+.

Similarly, if $\tau_- < \tau_{bin} < \tau_+$ the smallest possible negative $\Delta\tau$ occurs if the next incoming note is skewed at θ_{in_-}, and (3-28) shows that $\tau_{bin} + \Delta\tau$ ($\Delta\tau$ is negative) will always be greater than or equal to τ_-. If $\tau_{bin} + \Delta\tau$ then equals τ_-, by (3-9) any immediately-succeeding incoming note that falls within the bin and is greater than θ_{in_-} will yield a positive $\Delta\tau$ and thus increase τ_{bin}, and any note with input skew of θ_{in_-} will leave τ_{bin} fixed at τ_-.

Essentially, if τ_{bin} becomes "trapped" between the bounding values of $\tau_+$and τ_+, by (3-26) and (3-28) it cannot break those bounds even in the extreme cases of infinite strings of notes with $\theta_{in_{repeated}}$ at θ_{in_+} or θ_{in_-}. A note with any other input skew value causes a $\Delta\tau$ that is smaller than that of a note at θ_{in_+} and larger than that of a note at θ_{in_-}, and thus cannot cause τ_{bin} to exceed the bounds of τ_+ and τ_-.

All that remains to be shown is that for an arbitrary string of notes (all with input skews in the open interval $(\theta_{in_-}, \theta_{in_+})$) and with τ_{bin} starting at a similarly arbitrary value, τ_{bin} can be driven to enter the "trap" of the τ_+ and τ_- bounds. Assume that a string of notes enters the deskewer, and that the solenoid time applied to the beginning note in the string, τ_{bin_0}, is less than τ_-. For $\theta_{in_-} < \theta_{in_0} \leq \theta_{in_+}$, equation (3-9) and the mean value theorem give:

$$\frac{F(\theta_{in_0}, \tau_{bin_0}) - F(\theta_{in_-}, \tau_{bin_0})}{\theta_{in_0} - \theta_{in_-}} \geq \gamma \qquad (3\text{-}29)$$

Rearranging:

$$\theta_{out_0} = F(\theta_{in_0}, \tau_{bin_0}) \geq F(\theta_{in_-}, \tau_{bin_0}) + \gamma \left(\theta_{in_0} - \theta_{in_-}\right) \tag{3-30}$$

Because $F(\theta_{in_-}, \tau_{bin_0}) > 0$, (3-9) and (3-30) yield:

$$\tau_{bin_1} > \tau_{bin_0} + \frac{\gamma}{\beta}\left(\theta_{in_0} - \theta_{in_-}\right) \tag{3-31}$$

Thus, if the first note's input skew is greater than θ_{in_-}, τ_{bin_1} is guaranteed to be larger than τ_{bin_0}. If τ_{bin_1} is larger than τ_-, the comments at the beginning of this section hold; τ_{bin} is trapped between τ_- and τ_+ and can never cross those bounding values. If τ_{bin_1} is still less than τ_-, the above reasoning leads to:

$$\tau_{bin_2} > \tau_{bin_0} + \frac{\gamma}{\beta}\sum_{k=0}^{1}\left(\theta_{in_k} - \theta_{in_-}\right) \tag{3-32}$$

In general, if $\tau_{bin_j} < \tau_-$, $\tau_{bin_{j+1}}$ is lower-bounded:

$$\tau_{bin_{j+1}} > \tau_{bin_0} + \frac{\gamma}{\beta}\sum_{k=0}^{j}\left(\theta_{in_k} - \theta_{in_-}\right) \tag{3-33}$$

If τ_{bin_j} becomes larger than τ_-, (3-33) is no longer valid because there then exist θ_{in} that produce negative $\Delta\tau$; it can no longer be guaranteed that $\tau_{bin_{j+1}} > \tau_{bin_j}$. But $\tau_{bin_j} > \tau_-$ also means that τ_{bin} is trapped in the interval $[\tau_-, \tau_+]$, implying $\tau_{bin_{j+1}} > \tau_{bin_j}$. Equation (3-33) principally indicates that the tuning algorithm can drive τ_{bin} into $[\tau_-, \tau_+]$ with a finite number of input notes.

It is possible for the series $\sum_{k=0}^{\infty}\left(\theta_{in_k} - \theta_{in_-}\right)$ to converge to a positive value that is insufficiently large to guarantee that the lower bound of τ_{bin} is larger than τ_-; this situation is distinct from the case of all notes being skewed at θ_{in_-}. If such a "degenerate" series of notes occurs, (3-9) guarantees that the $\Delta\tau$ for any of the notes in that series is at least as large as the $\Delta\tau$ would have been had that note been skewed at θ_{in_-}. Since the series of $\Delta\tau$ for notes all skewed at θ_{in_-} causes τ_{bin} to converge to τ_- by (3-26), the degenerate series must converge to at least τ_-, and may move into $[\tau_-, \tau_+]$.

An analysis identical to that which led to (3-33), but for the case of $\tau_{bin_j} > \tau_+$, gives the following result:

$$\tau_{bin_{j+1}} > \tau_{bin_0} - \frac{\gamma}{\beta} \sum_{k=0}^{j} \left(\theta_{in_+} - \theta_{in_k}\right) \tag{3-34}$$

Like (3-33), (3-34) does not guarantee that τ_{bin} will eventually be bracketed by τ_- and τ_+, but it does place an upper bound on τ_{bin} and show that it can be made smaller than τ_+ with a finite string of input notes. In the worst cases, where all input notes are skewed at θ_{in_-} or a degenerate series of notes occurs, (3-28) indicates that τ_{bin} converges to τ_+ while never becoming smaller than τ_+.

The results of this and the preceding section serve two purposes: to give theoretical grounding to the "common sense" underlying the tuning process, and to show that under the assumptions of Section 3.4 and the tuning rule of (3-9), τ_{bin} converges to either τ_- or τ_+, or eventually enters $[\tau_-, \tau_+]$, where it fluctuates while always remaining in that interval.

3.7 Error in Deskewing Performance

The finite width of each bin causes the situation shown in Figure 3.9; τ_{bin} intersects G at only a single input skew, $\theta_{in_{intersect}}$. In that bin, notes with $\theta_{in} > \theta_{in_{intersect}}$ are under-corrected (exit with positive skew) and notes with $\theta_{in} < \theta_{in_{intersect}}$ are over-corrected (exit with negative skew).

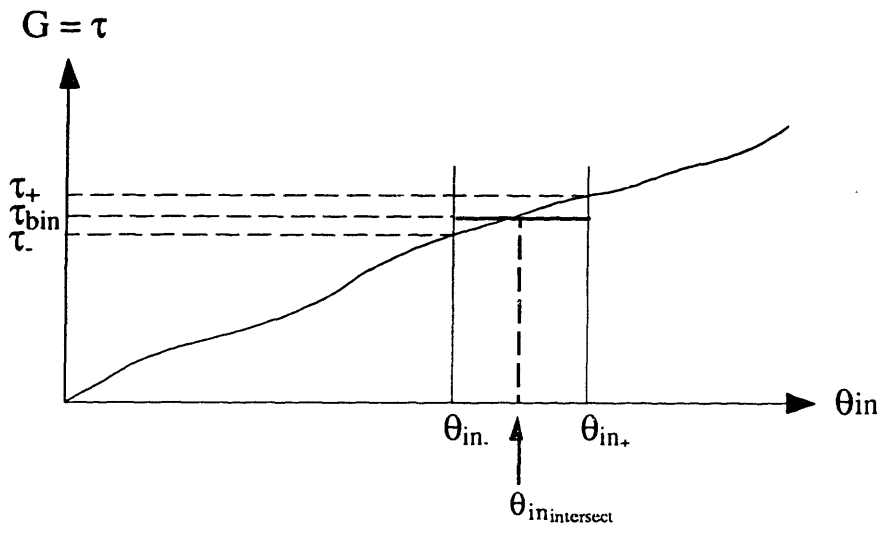


Figure 3.9: Bin Value is Correct at One Input Skew

A smaller bin produces less over- and under-correction at the edges of the bin; in the limiting case of an infinitely narrow bin, all input skews could be perfectly straightened. However, very small bins are not necessarily desirable because the total number of notes that must be input to assure frequent tuning can become prohibitive. The whole purpose of the tuning procedure is to allow τ_{bin} to adapt to changes in the shape of G that may come from worn belts, old notes, or shifts in humidity, which makes it desirable to have a large number of notes falling into each bin. This presents a design tradeoff: with narrower bins, less error is possible, but less-frequent tuning occurs.

A rough worst-case analysis of the problem can be performed with five basic assumptions. First, for simplicity, all bins are of equal width. Second, the input skew of notes entering the deskewer is evenly distributed among all bins, an approximation that is not probabilistically correct but will serve adequately in this basic analysis. Third, the deskewer is tuned from a state in which each bin's τ_{bin} is set to 0 (the machine starts with no initial information). Fourth, each bin tunes to its τ_- as slowly as is allowed by the bounds given in (3-26) and (3-28). Each bin is tuned to τ_- because that value causes maximal under-correction for any future notes that enter at the bin's θ_{in_+}, and to make the tuning even slower, the constant θ_{out_j} in (3-26) and (3-28) is taken as θ_{in_+}. This is the largest possible value of θ_{out} for the bin (it can occur during the first control cycle that falls in the bin, when $\tau_{bin}= 0$). Finally, it is assumed that $\frac{dG}{d\theta_{in}}$ has an upper bound, known as $\left.\frac{dG}{d\theta_{in}}\right|_{max}$.

The goal of the analysis is to minimize the error at one particular skew, which is placed at the right (θ_{in_+}) edge of a bin after the ideal number of bins has been determined.Taking the width of single bin as $\Delta\theta_{in}$ and using (3-26), the maximum error in τ_{bin} comes at the θ_{in_+} edge of the bin. The skew being optimized for error is chosen to be θ_{in_+}, which is assumed to be greater than 0° (although an identical analysis can be performed for negative θ_{in}):

$$\varepsilon_\tau = \frac{\theta_{in_+}}{\alpha}\left(1-\frac{\alpha}{\beta}\right)^n + \left.\frac{dG}{d\theta_{in}}\right|_{max} \Delta\theta_{in} \tag{3-35}$$

The largest possible error in θ_{out} (for the chosen skew and its accompanying bin) is thus:

$$\varepsilon_\theta = \beta\left[\frac{\theta_{in_+}}{\alpha}\left(1-\frac{\alpha}{\beta}\right)^n + \left.\frac{dG}{d\theta_{in}}\right|_{max} \Delta\theta_{in}\right] \tag{3-36}$$

$\Delta\theta_{in}$ is now eliminated from (3-36). If N is the total number of notes fed during the tuning process, $\theta_{in_{range}}$ is the total range of input skews encountered during machine operation, and n is the number of notes in each bin (the notes are evenly distributed in the bins as assumed above):

$$\Delta\theta_{in} = \frac{n\theta_{in_{range}}}{N} \tag{3-37}$$

Substituting (3-37) into (3-36):

$$\varepsilon_\theta = \beta\left[\frac{\theta_{in_+}}{\alpha}\left(1-\frac{\alpha}{\beta}\right)^n + \left.\frac{dG}{d\theta_{in}}\right|_{max}\frac{\theta_{in_{range}}n}{N}\right] \tag{3-38}$$

(3-38) is a sum of a decaying exponential (because $0 < \left(1-\frac{\alpha}{\beta}\right) < 1$) and an increasing linear term, so it has a minimum at some n > 0. Taking the derivative of ε_θ with respect to n and setting it to zero yields:

$$\frac{\theta_{in_+}}{\alpha}\ln\left(1-\frac{\alpha}{\beta}\right)\left(1-\frac{\alpha}{\beta}\right)^n = -\left.\frac{dG}{d\theta_{in}}\right|_{max}\frac{\theta_{in_{range}}}{N} \tag{3-39}$$

Solving for n:

$$n = \frac{1}{\ln\left(1-\frac{\alpha}{\beta}\right)}\ln\left[\frac{-\left.\frac{dG}{d\theta_{in}}\right|_{max}\frac{\alpha\theta_{in_{range}}}{N\theta_{in_+}}}{\ln\left(1-\frac{\alpha}{\beta}\right)}\right] \tag{3-40}$$

Using the n of (3-40) in (3-37) gives the number of bins that yields the minimum largest-possible error in the particular bin associated with θ_{in_+}:

$$\text{Optimal number of bins} = \frac{\theta_{in_{range}}}{\Delta\theta_{in}} = \frac{N\ln\left(1-\frac{\alpha}{\beta}\right)}{\ln\left[\frac{-\left.\frac{dG}{d\theta_{in}}\right|_{max}\frac{\alpha\theta_{in_{range}}}{N\theta_{in_+}}}{\ln\left(1-\frac{\alpha}{\beta}\right)}\right]} \tag{3-41}$$

(3-41) can be used as a rough guide to the optimal number of bins given a limited amount of time for learning, a desire to minimize error in one particular bin, and worst-case assumptions. Similar analyses can be performed if the main concern is drift in G, but the rate of drift and the note throughput must be known.

To find the order-of-magnitude of the optimal number of bins, (3-41) is now evaluated for various values of N, given θ_{in_+}, $\theta_{in_{range}}$, the values of α and β from Section 4.2, and an estimate of $\left.\frac{dG}{d\theta_{in}}\right|_{max}$ taken from the 0°-line of Figure 4.1. α is 0.05 °/ms, β is 1 °/ms, and $\left.\frac{dG}{d\theta_{in}}\right|_{max}$ is roughly 7 ms/°. The operational range ($\theta_{in_{range}}$) of the deskewer is approximately 50° (-25° to 25°), and for the purposes of evaluation θ_{in_+} is taken as 20°. Inserting all numerical values yields:

$$\text{Optimal number of bins (for above numerical values)} = \frac{-.0513\,N}{2.8365 - \ln(N)} \qquad (3\text{-}42)$$

For a training set of N=1000, which can be completed in under one hour, (3-42) gives the optimal number of bins as 13. For N=10000, which requires approximately one day, the most desirable number of bins is 80. These results are very conservative; they imply that 80 to 100 notes (N divided by the number of bins) are needed to tune a bin, while the results of Section 4.3 suggest that approximately twenty notes are needed. It appears that in actual operation, more bins can be used than suggested by (3-42). This underestimate is attributable to the approximate nature of the preceding analysis, the assumptions of worst-case tuning behavior, and the safety factors on α and β (applied in Section 4.2 to guarantee theoretical convergence of the tuning process) that reduce the ratio α/β.

3.8 Limitations of the Analyses

There are five main discrepancies between the idealizations of this chapter's analyses and the actual deskewer. The first is a detail of the hardware implementation: because everything is digitally controlled, continuous intervals in the above derivations (such as references to $[\tau_-, \tau_+]$) actually contain only a discrete number of values, and all measured skews and times are discrete. Note skew, for example, is measured by a digital timer with a resolution of 20 microseconds. The timer's quantization is unimportant given that the skew measurement is rounded to one-millisecond values by the binning process.

The second difference arises in the deviation of the solenoids from their modeled performance. At small θ_{in} (less than approximately 3°) very short solenoid firing times are required, but the solenoid inertia is too large to allow such brief control actions. Thus, τ_{bin} values at the low end of the θ_{in}-range are apt to increase until they are large enough to fire the solenoid, at which point the notes that fall into those bins are over-corrected. The solution to this problem is very simple; in bins containing small θ_{in} the fire times are set to zero, because the notes that fall into those bins cause no jamming. This is equivalent to implementing a threshold beneath which notes are considered fully deskewed and not subject to correction. It is convenient to use the design limit of 8° as the cutoff below which the solenoids are not applied.

Third, it was noted in Section 2.3 that the lateral position of notes in the transport belts affects the required deskewing time. Feeding causes a lateral variation of up to approximately 10 mm, which can change the required deskewing time by a few milliseconds. The deskewer does not currently have sensors that detect lateral position, so this effect cannot be compensated for with the present control system.

The fourth discrepancy is that F and G are not static functions. This is the main reason for the use of a tunable open-loop controller; F varies with changing note and environmental conditions, which means that G must change, and the bin values are modified to match the alterations in G. If the rates of change of F and G are slow compared to the rate of note throughput (during a given string of notes F and G undergo little change), this effect is negligible. The open-loop controller is specifically designed to handle the slow variations in required deskewing time caused by factors such as machine wear and changing humidity.

Finally, F does not depend exclusively on θ_{in} and τ. Notes that enter with identical input skews and are given the same solenoid timing are observed to leave with different output skews (as much as several degrees apart). This appears to be a function of inaccuracies in the control hardware (the solenoid inertia causes each deskewing action to have slightly different timing, and the flexibility in the solenoid mounting brackets allows a small amount of solenoid bounce), slight changes in machine velocity during note transport, and variations from bill to bill in surface finish, thickness, and wear. While one of the premises of the tunable open-loop controller is that it accounts for changing note conditions, this is only true for a long string of similarly worn notes; all of the above analyses implicitly assume that the incoming notes are identical in all respects except for skew, and that their required deskewing times are only dependent on θ_{in}. The unavoidable variations in mechanical properties from note to note occur too quickly to be compensated for by the tunable open-loop controller.

The complexity of F is a theoretically important issue because it affects the basic premise of the convergence analyses; it is unknown at this time if the analyses can be successfully modified to account for variability in F and still show some form of convergence. In actual operation, however, the simplified model of F (equation (3-5)) provides a good description of the tuning behavior - the prototype successfully learns how to deskew notes to within the operational limits of Omron's ATM (see Chapter 4).

Chapter 4
Results

4.1 Implementation of Deskewer Control

The deskewer control has been implemented on an IBM PC/AT clone using a Real Time Devices TC-24 timer & digital I/O board. The board has an Am9513 timer chip that is used to both measure skew and count the solenoid fire and wait times, as explained below.

The control program (see Appendix E) uses the additional input variable of note velocity. This increases the binning from one dimension, as used in the analyses of Chapter 3, to two dimensions (which is shown graphically as a plane with velocity on the x-axis, input skew on the y-axis, and firing time on the z-axis) but does not otherwise affect the convergence results given in Chapter 3 because each set of experiments is run at a single velocity. The velocity binning is only a convenience; when experiments are run at a new velocity, the tuning can begin using solenoid times learned at speeds close to the new setting (from the new velocity's bin). If tuning without preexisting information is desired, the bins are emptied.

At the beginning of a set of experiments belt velocity is input from the tachometer by an eight-bit A/D converter and classed into one of 200 bin-coordinates, the same number of bins used in the skew dimension. The number of divisions on the skew axis is large because there is no time constraint on training the deskewer; very narrow bins more fully demonstrate the capabilities of the mechanical hardware by reducing the performance degradation caused by using a single fire time for a wide range of input skews. As a programming convenience, the same number of divisions is used for the velocity and skew dimensions.

Following velocity measurement, the deskewer is ready for operation. The sequence of events triggered by each note is as follows:

1. The infrared beam of the initial-skew optical sensor pair (see Appendix B) on the leading side of the note is blocked. This starts a 50 Khz-clock-rate counter/timer on the Am9513 chip. A flag that indicates direction of note skew is set on a flip-flop (external to the PC).

2. The other initial-skew sensor pair is blocked. This stops the counter and pulls high an interrupt line (IRQ3) on the motherboard of the PC (see Appendix D).

3. The interrupt routine reads the skew timing information, gives it arithmetic sign by examining the direction-of-skew flag, and stores the signed result in the first column of a two-column array (the second column stores the timing information measured after the note is deskewed). The timer value is then classified into one of the 200 bin-coordinates, which cover a range of timer values from -5000 to 5000 (because the largest observed skew measurement at any speed is approximately 4800 counts). The 50 Khz clock rate of the timer means that each count is equivalent to 20 microseconds, so with 200 bins each bin's width is one millisecond. This bin size has been experimentally observed to be approximately the smallest meaningful time distinction; for example, at a test speed of 0.6 m/s (used in the results below), a sensor lateral separation of 5 cm, and with small skew angles, one millisecond of sensor time differential is roughly equivalent to 0.65° of skew.

4. The solenoid firing time is taken from the unique bin determined by the velocity and skew coordinates. Two additional counters on the Am9513 are then programmed; the first holds the amount of time to wait until solenoid firing (the wait time, which is calculated from the velocity of the bill and the dimensions of the deskewer), while the second contains the length of time to pin the note (the fire time, taken from the bin). If the measured skew is less than the allowable 8°, the fire time is set to zero. The interrupt routine exits.

5. After the wait timer has counted down, the fire-time counter begins to decrement. This fires the solenoid on the leading side of the bill. The choice of solenoid (left or right side) is controlled by the flip-flop that recorded initial-skew direction; the selection is accomplished by logic gates external to the PC. After the fire timer counts down to zero, the solenoid lifts.

6. One of the two outgoing-skew optical sensor pairs is blocked by the leading side of the recently deskewed note, which starts a fourth timer on the Am9513 chip. A second flip-flop is set to hold the direction of outgoing note skew.

7. The other outgoing-skew sensor pair is blocked and the timer stops. This triggers a second interrupt, on a completely separate interrupt line (IRQ5).

8. The second interrupt reads the raw timing data, gives it arithmetic sign by checking the second flip-flop, and stores the data in the second column of the array alongside the note's previously-measured incoming skew. The velocity and input skew values are again examined to determine which bin should be tuned, and the measured output-timing value is then used to alter the control action in that bin.

Steps 1-8 are repeated for each note, although in practice there can be interleaving of the input-skew interrupts and the output-skew interrupts. The program maintains internal pointers to properly correlate a given output interrupt with its appropriate input interrupt.

When training with no preexisting information, the deskewer's bins are initially set to a uniform firing time of one millisecond, which is insufficiently long to trigger solenoid action. A slight movement is visible but no contact is made with the passing bill. Thus, the first notes (in any region of skew/velocity space) pass through the machine unchanged, and the tuning rule increases the firing time until it is large enough to fire the solenoids.

4.2 Bounds on α and β

The tuning algorithm and convergence results of Chapter 3 depend on knowledge of α and β, the bounds on $\left|\frac{\partial F}{\partial \tau}\right|$. These numbers are determined by direct evaluation of experimental data; the solenoid firing time is hard-coded and notes of varying skew are fed through the machine. The timing is then changed, and the process is repeated. Figure 4.1 shows a graph of several level curves of F obtained via this method, all taken at a belt velocity of 0.6 m/s (equivalent to four notes/second throughput).

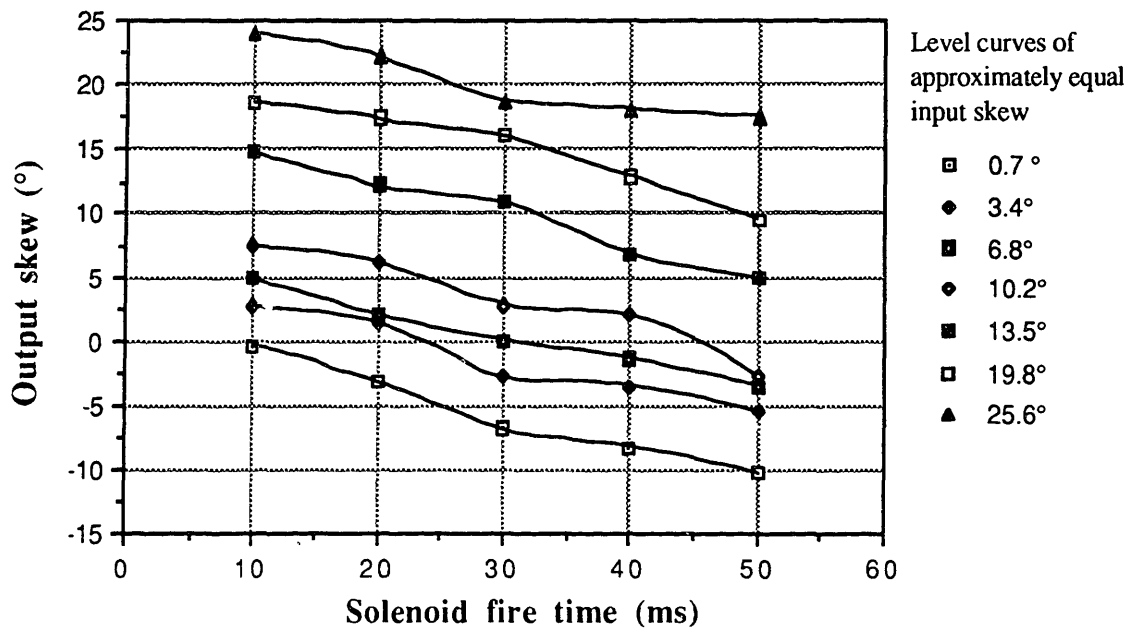


Figure 4.1: Experimental Graphs of $F(\theta_{in},\tau)$

Data points have been connected into level curves by rounding the time differences between sensor obscuration to the nearest two milliseconds; higher precision and statistical treatment of multiple tests at a single input skew and firing time are not needed because the only data extracted from the experiments are rough bounds on α and β. After the timing data has been grouped it is converted to skew angle. Although level curves with less than the 8° of input skew required for solenoid activation appear on the graph, there is no problem with the control algorithm; the data was not taken during tests of the controller. The chosen fire times, all equal to or greater than 10 milliseconds, were hard-coded into the control program and were sufficiently long to completely extend the solenoids.

As was noted in Section 3.5, α and β must be conservatively specified, so values found from Figure 4.1 can be easily made "safer" by increasing β and decreasing α. Examination of Figure 4.1 shows that the largest change in output skew per 10-millisecond change in solenoid firing time is approximately 5°, giving $\beta = 0.5$ °/ms. However, to ensure convergence, β is multiplied by a safety factor of 2, and is thus taken as $\beta = 1$ °/ms. Similarly, α is initially read from the graph as a 1° decrease in skew for a 10-millisecond increase in firing time, but applying a safety factor of 0.5 yields $\alpha = 0.05$ °/ms. The bound on α is not as critical as that on β, because the only necessary condition α contributes to the convergence analyses is $\beta > \alpha > 0$.

4.3 Tuning in a Single Bin

With the conservative value of β found in the preceding section, (3-13) was used to tune the controller. The testing velocity was again 0.6 m/s. The 38 notes graphed in Figure 4.2 all entered the deskewer with 30 to 31 milliseconds of input-skew-sensor time difference (all were in a single bin), equivalent to a range of 19.8° to 20.4° skew. The bin's initial fire-time value was one millisecond, so the graph displays the complete tuning of the bin.

After the nineteenth note the largest output skew is 7.3°, while the root-mean-square (R.M.S.) average skew for notes 19 to 38 is 3.73°; both figures fall within the design limit of 8°, thus indicating successful deskewer performance. The results shown in Figure 4.2 are typical of the deskewer's tuning behavior in bins that are trained from one-millisecond initial timing values (except those that contain notes with a skew of less than 8°, in which the solenoids are not allowed to fire). The output skews of notes in a bin's range are typically not more than 5° after approximately twenty notes have fallen into the bin and adjusted the stored firing time.

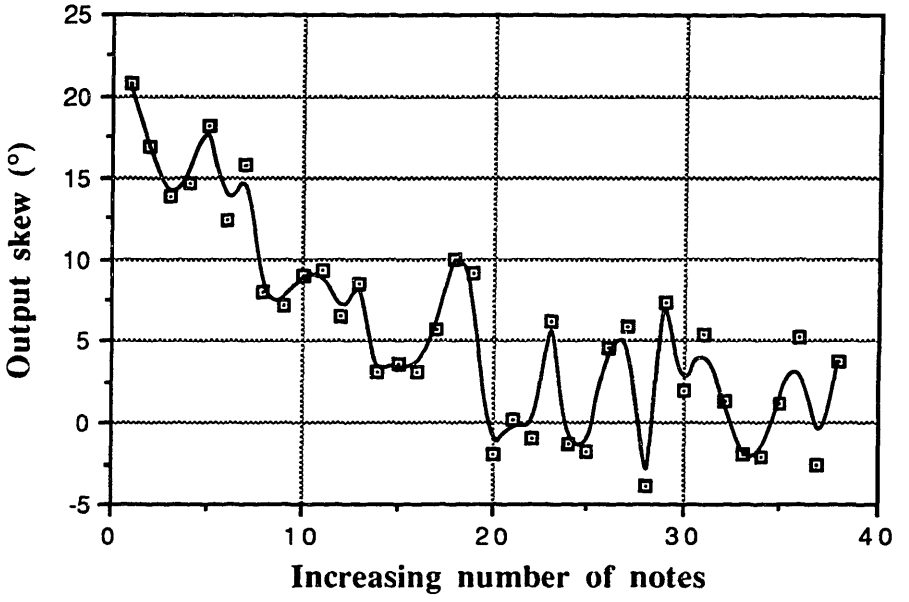


Figure 4.2: Experimental Graph of Output Timing in 19.8° - 20.4° Bin

The small fluctuating output skew in the later notes of the graph appears to be an unavoidable consequence of the choice of a tunable open-loop controller. The solenoid firing time for a given note can be close to the necessary value and can yield adequate results, but the tuning algorithm can only alter the control mapping after the note has exited the deskewer. The slight variation from note to note cannot be effectively compensated for without a control system that adjusts the control action for each bill as it is deskewing - essentially, a closed-loop controller. However, after the nineteenth note the amount of residual skew observed in Figure 4.2 falls within the 8° design limit of Omron's machines; the tunable open-loop controller's performance can still be considered acceptable.

Chapter 5
Conclusions

In response to the Omron Corporation's need for both analytical models of the note-handling process and error-correcting systems for its cash-handling machines, a prototype note deskewer has been designed, analyzed, and built; an analytical method for predicting the behavior of notes subject to frictional contact forces has been formulated; and a tunable open-loop controller has been implemented to compensate for the varying environmental and note-quality conditions that affect the required solenoid timing. The deskewer has met the design goals of low cost, simple programmability, and reliable, low-maintenance operation successfully enough for engineers at Omron to indicate that it may be included in future cash-handling equipment.

The analyses of bill-handling by application of frictional force, in Chapter 2 and Appendix C, present a method for evaluating the performance effects of changing design parameters (such as solenoid position in the deskewer, or note velocity in the feeder). The Coulomb-friction model is presently being used in simulations of new feeder technology and may be applied to other Omron paper-handling machines (such as subway ticket vending machines) in the future.

To compensate for the varying environmental and note-quality conditions that affect the required solenoid timing, a tunable open-loop controller has been implemented as described in Chapters 3 and 4. The controller successfully reduces output skew to less than the the design limit of 8°; the R.M.S. average is approximately 4°. The open-loop scheme incorporates the adjustability of a closed-loop controller (over a series of notes; there is no individual feedback) but has advantages in both the cost and complexity of control hardware. Its sensor requirements are met by inexpensive optical sensor pairs that take a single measurement of skew, and the control hardware needs little more than timers (to measure skew and program the solenoids) and a small amount of memory (to hold the solenoid firing times). If β is rounded up to the nearest power of two, the tuning of the control times can be accomplished by bit-shifting and adding the skew feedback measurements to the appropriate memory location, thereby eliminating the need for floating-point mathematics.

All three results - the design, the analysis, and the controller - can be extended. Alternative skew sensors should be examined as the first change to the deskewer's hardware. While the optical sensor pairs that are currently used are inexpensive, established technology, they do not possess an infinitely thin beam (thus causing variation in triggering time) and must be correctly aligned to ensure that skew is measured without bias. If a cheap, accurate, and fast skew sensor can be developed, the measurement of skew can be performed more precisely, which improves the quality of solenoid timing selection and adjustment.

The analytical methods generally yield equations that are solved by numerical simulation. Because computer solutions are used, other effects that are usually handled numerically, such as bending and buckling, can be easily added. For example, a program that incorporates both the friction model of Chapter 2 and a model of the large-deformation buckling behavior of a note is now being written to simulate a new feeder design.

The tunable open-loop controller can be extended in both the complexity of the technique and the variety of systems to which it is applicable. Instead of constant values in the bins, for example, piecewise-linear (or quadratic, or exponential, or trigonometric, etc.) functions can be used. The practical drawback is that such a scheme does not directly take the solenoid firing times from memory; the control actions must be calculated, which requires more elaborate electronics and may negate the cost advantage of the controller.

While the specific tuning rule used in the deskewer's controller (Equation (3-13)) is shown to converge under assumptions based on qualitative knowledge of the deskewer's behavior, the tunable open-loop controller technique may also be applicable to other systems in which the length of time of control actuation is the important variable (as opposed to the magnitude of control action). A possible example is pulse-width-modulation control of DC-motor speed, in which the duty cycle of the applied voltage is the control action. A controller that periodically measures the motor's angular velocity, adjusts the duty cycle in an attempt to bring the measured speed up (or down) to the desired speed, and then re-checks the angular velocity could be identical in form to the controller used on the deskewer.

Another example is a toaster, in which the type of bread and time of toasting are the control variables. Such a system could have the user press a button identifying the type of bread to be toasted, thus starting a toasting cycle that runs for the length of time stored for the selected type. At the end of the cycle the user indicates whether the bread has been satisfactorily toasted, and the feedback is used to modify the toasting time for future slices of the same type. This

control system might be useful if the required toasting period changes over time; for example, if the toaster's heating elements degrade with age.

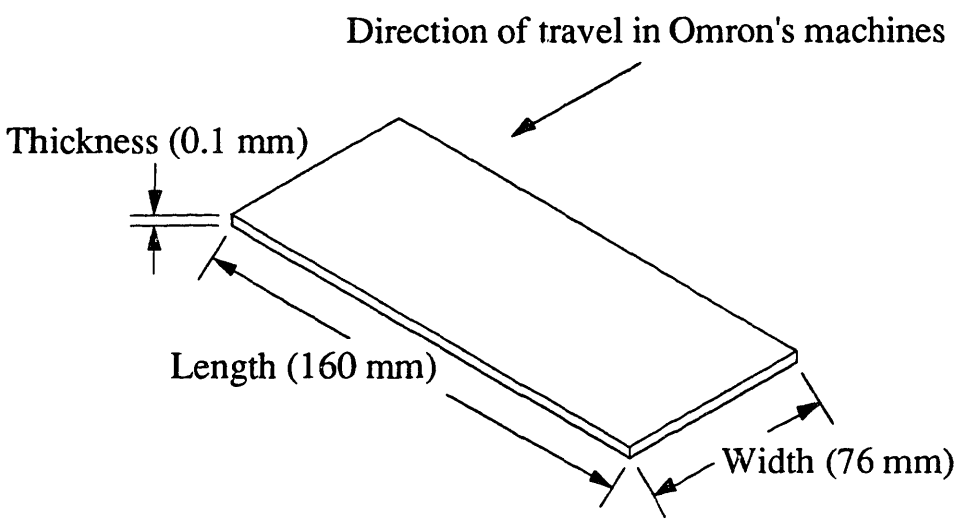
Both the motor and toaster examples are present topics of research, as are the requirements that a general system must meet to allow the effective, convergent application of a tunable open-loop controller. In addition, a comprehensive literature review is being done to place the controller in the appropriate area of control theory.

Appendix A
The Yen

There are three types of bank notes circulated in Japan: ¥10000, ¥5000, and ¥1000 (in early 1992, ¥1000≈$7.70). The ¥10000 note is the most commonly handled denomination in Japanese ATM; Omron's HX-ATM, their most modern as of 1992, stores a minimum of twice as many ¥10000 as either of the smaller bills - 1000 ¥1000 notes, 1000 ¥5000 notes, and 2000 ¥10000 notes. The largest bills are so prevalent that the HX-ATM can be configured to handle only ¥10000 notes, and an external storage cartridge that holds 2000 ¥10000 bills is available to boost the maximum storage capacity to 6000 ¥10000 bills.

All experiments on the deskewer are conducted using dummy notes of the same size and mass as the ¥10000 bill; the same dummy notes are used by Omron in the development and testing of their bill-handling equipment.

The dimensions and mass of the ¥10000 note are shown in Figure A.1. The ¥5000 and ¥1000 bills are slightly shorter in length and are thus lighter.



Mass: 1 gram ± 0.1 gram (humidity changes cause weight variation)

Figure A.1: ¥10000 Note Characteristics

Appendix B
The Deskewer

The deskewer concept was created by a team of four people: Harry West, Ross Levinsky, Ryuichi Onomoto (a mechanical engineer from Omron), and Jack Kotovsky. Kotovsky, in close consultation with Onomoto, designed and built the actual prototype for his bachelor's thesis. All diagrams in this section use measurements that were either taken from his thesis [Kotovsky 1989] or directly measured from the deskewer.

The deskewer's optical sensor and solenoid-firing relay circuits were designed by Ichiro Kubo, an Omron electrical engineer, and wired by Kotovsky. Levinsky designed and built all dedicated control hardware, handled all interfacing of the deskewer to the IBM-PC/AT clone, and designed and programmed all control software.

Figure B.1 shows the functional elements of the Omron-supplied test fixture to which the deskewer is attached. Notes enter the transport belts from a feed cartridge identical to those used to store bills in Omron's GX-series ATM; however, the bills are manually inserted into the cartridge in a skewed position to provide rotated test notes for the deskewer. After the notes are deskewed they fall into a small bin at the end of the test fixture. Maximum transport speed of the fixture is approximately 1.5 m/s, equivalent to 10 notes/second throughput.

The deskewer is shown in greater detail at the bottom of Figure B.1. The upper and lower belts (between which the notes are transported) can be clearly seen, as can the two solenoids and four pairs of optical sensors. The whole section is driven by a toothed belt running on the drive gear half-visible on the left side of the drawing.

Figure B.2 lists the parts that make up the main frame of the deskewer. The shafts, belts, rollers, and drive gear are all supplied by Omron and are typical of their note-handling technology.

Figure B.3 lists the optical sensors (supplied by Omron), the solenoids, and various miscellaneous hardware.

Figure B.4 gives dimensions of the deskewer's side plate. The 3mm holes are drilled for the fixed-shaft mounting screws, while the 10mm hole holds the bearings for the drive shaft. The left and right side plates are identical.

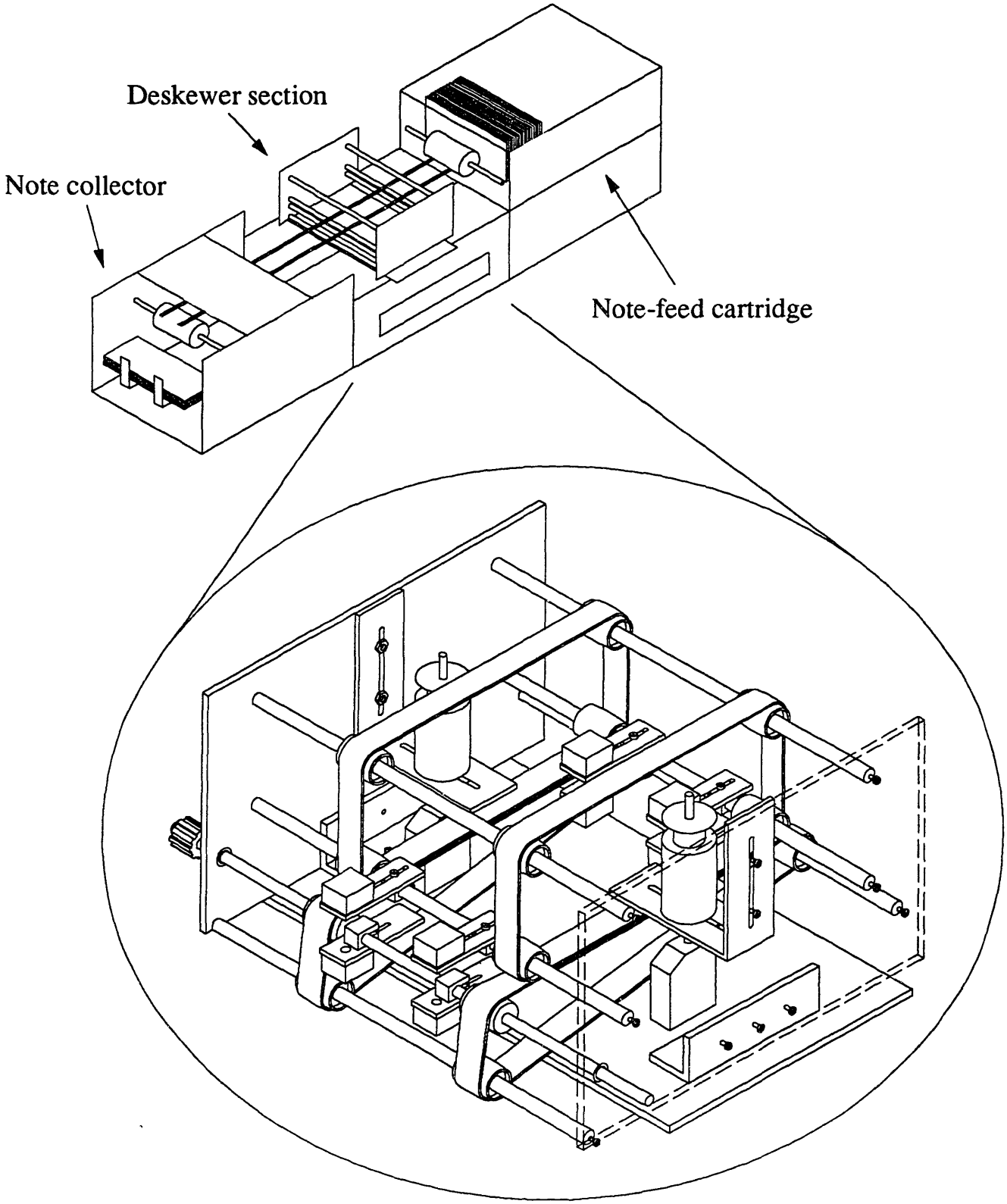
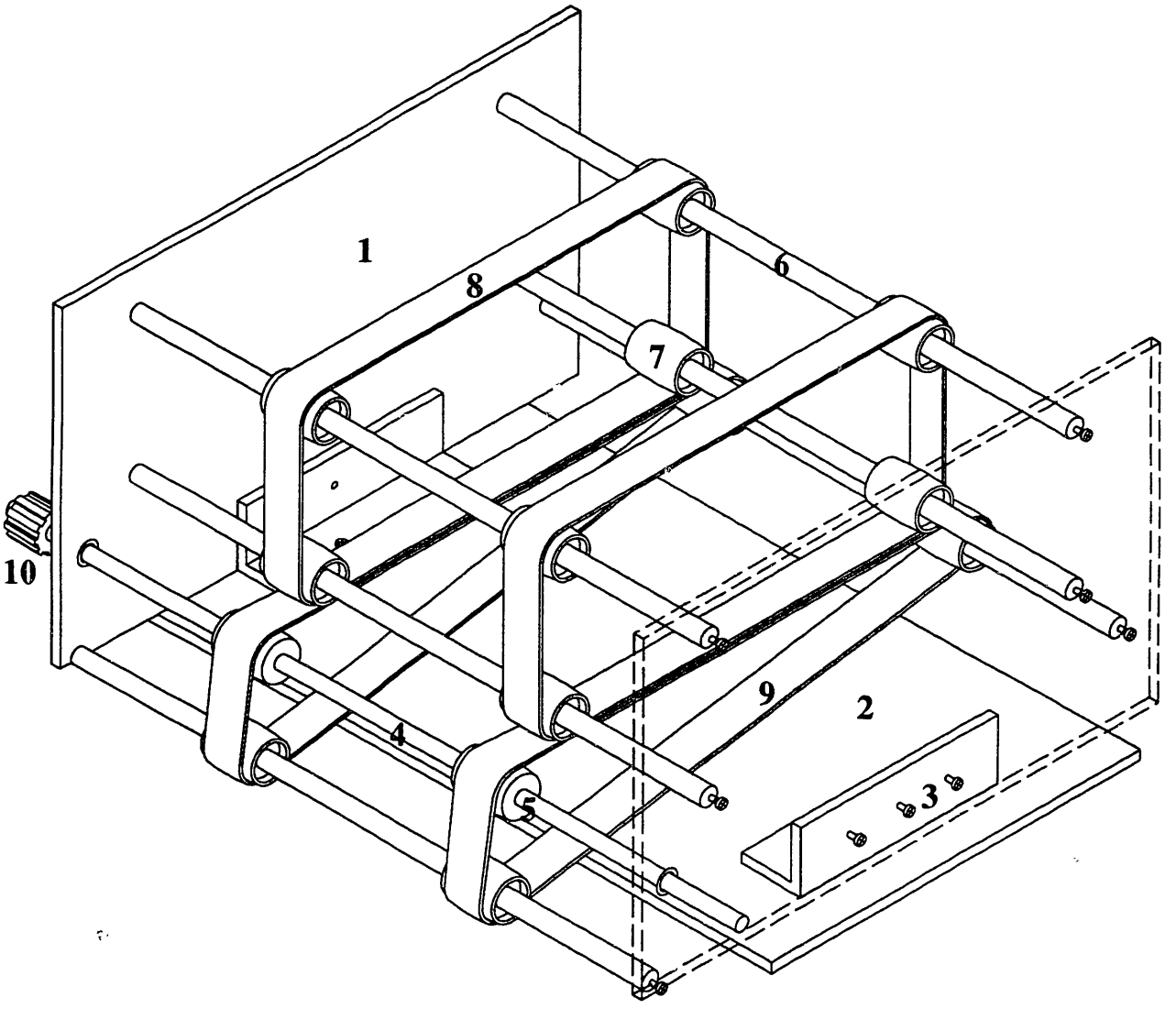
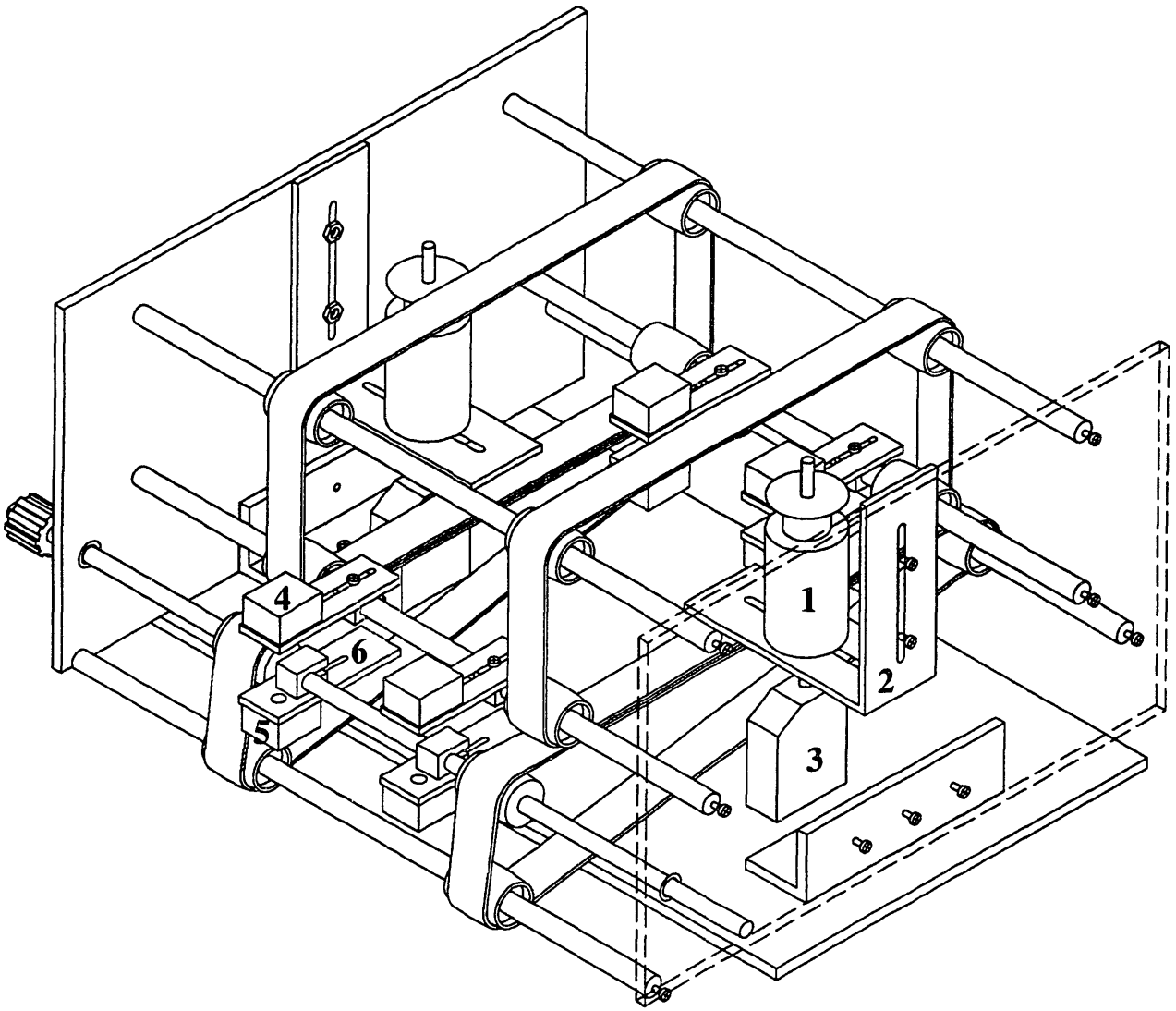


Figure B.1: Test Fixture and Deskewer Closeup



1. 171mm x 105mm x 3mm side plate (2)
2. Bottom plate (1)
3. Angle bracket for side plate/bottom plate attachment (2)
4. 6mm diameter x 250mm length drive shaft (1). Mounted in bearings.
5. 19mm peak-diameter x 18mm width crowned drive roller (2). Set-screwed to drive shaft.
6. 8mm diameter x 190mm length roller shaft (6). Fixed to side plates.
7. 19mm peak-diameter x 18mm width crowned transport-belt roller (12). Free-spinning on roller shafts.
8. 362mm x 9mm x 1mm (when unstretched) rubber transport belt (2)
9. 366mm x 9mm x 1mm (when unstretched) rubber transport belt (2)
10. 14.4mm diameter drive gear (1)

Figure B.2: Main Structure of Deskewer



1. Shindengen F224C-12V solenoid (2)
2. Solenoid mounting bracket (2)
3. Solenoid pinning block (2)
4. Omron/Sharp M601P phototransistor (4)
5. Omron infrared LED (4)
6. Optical sensor mounting assembly, consisting of shaft clamp and adjustable mounting plate (8)

Note: Lateral spacing of the optical-sensor pairs is 5 cm from infrared-beam center to infrared-beam center.

Figure B.3: Sensors and Solenoids

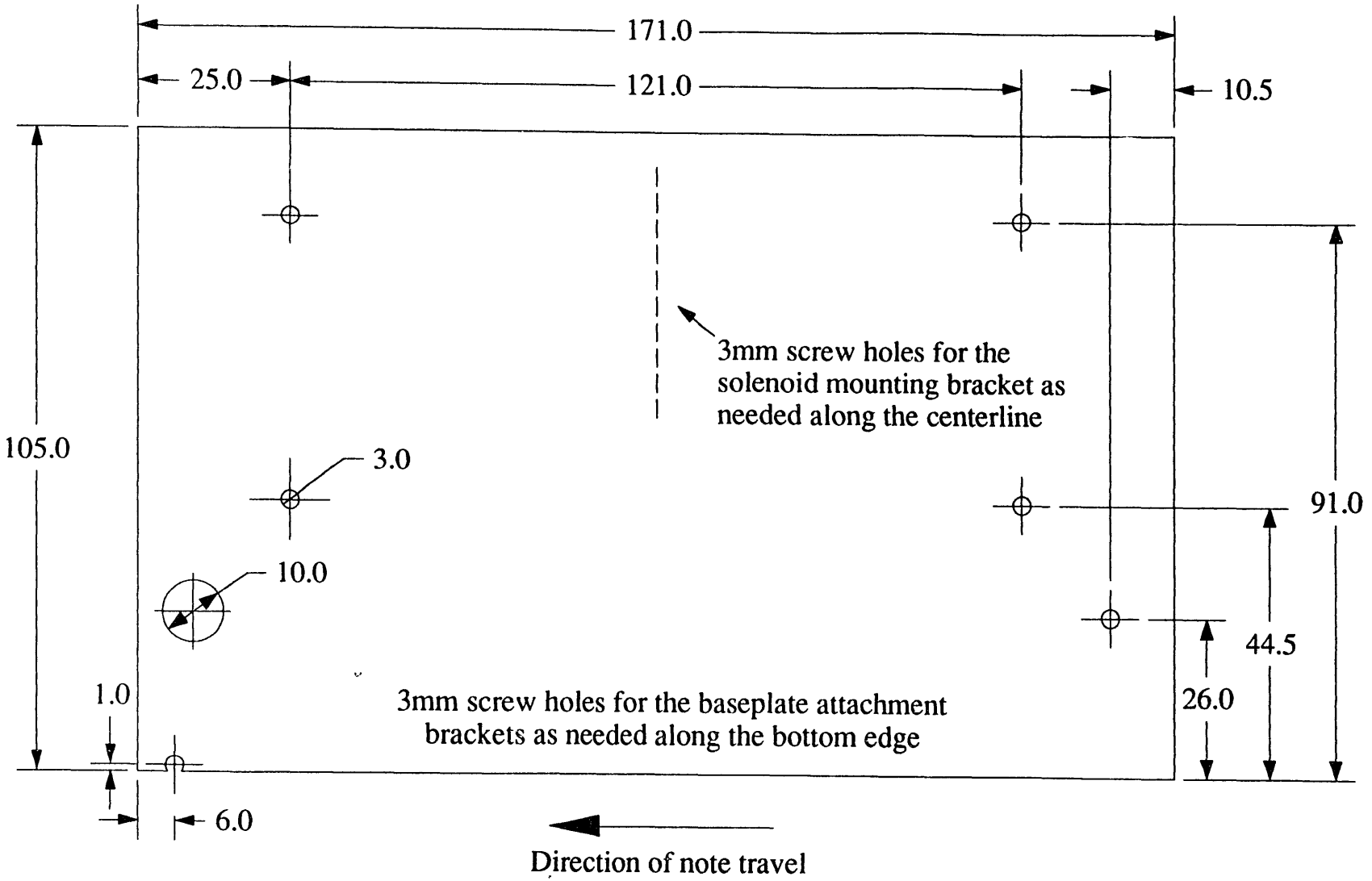


Figure B.4: Side Plate of Deskewer

Appendix C
Feeder Analysis

C.1 Origin of Model

The model presented in this appendix was formulated during investigations of the origins of skew. One theory on the origin of skew held that a note's skew increased during the feeding process if it entered the rollers at a slight angle; this idealized model of the system was developed and numerically solved to test that hypothesis.

The yen is modeled as a flat, rigid plate which compresses in a direction normal to its surface, and the feeder as two zero-compliance rollers with a fixed gap in between. The force on the note comes from its compression by the rollers, which remain undeformed. This is in contrast to the actual system in which the rollers are sheathed in rubber and are touching each other with a spring-set force; the force on the note comes from the deformation of the rubber by the presence of the note.

The system is not modeled with deformable rollers because such a case is mathematically complex (see [Wong 1986], [Wong 1984], and both [Soong 1981] for typical solution methods) and is generally solved with a finite element code or custom-written software. A deformable-note model gives qualitatively similar forces on the yen with less computational difficulty.

C.2 Direction of Frictional Force Vector

The feeder is composed of two rollers of possibly different radius. A plan view is shown in Figure C.1 and a side view is shown in Figure C.2; the coordinate system used in the analysis is also shown in those two drawings. The rollers are assumed to be infinitely long in the Y-direction. The note's center-of-mass X-position and Y-position are X_C and Y_C, respectively. The corresponding velocities are $\dot{X}$ and $\dot{Y}$. The angular velocity about the center of mass is $\dot{\theta}$ (when treated as a vector in equations (C-1) and (C-3) it is printed in boldface; elsewhere it is the scalar magnitude of angular velocity and is printed in plain italic).

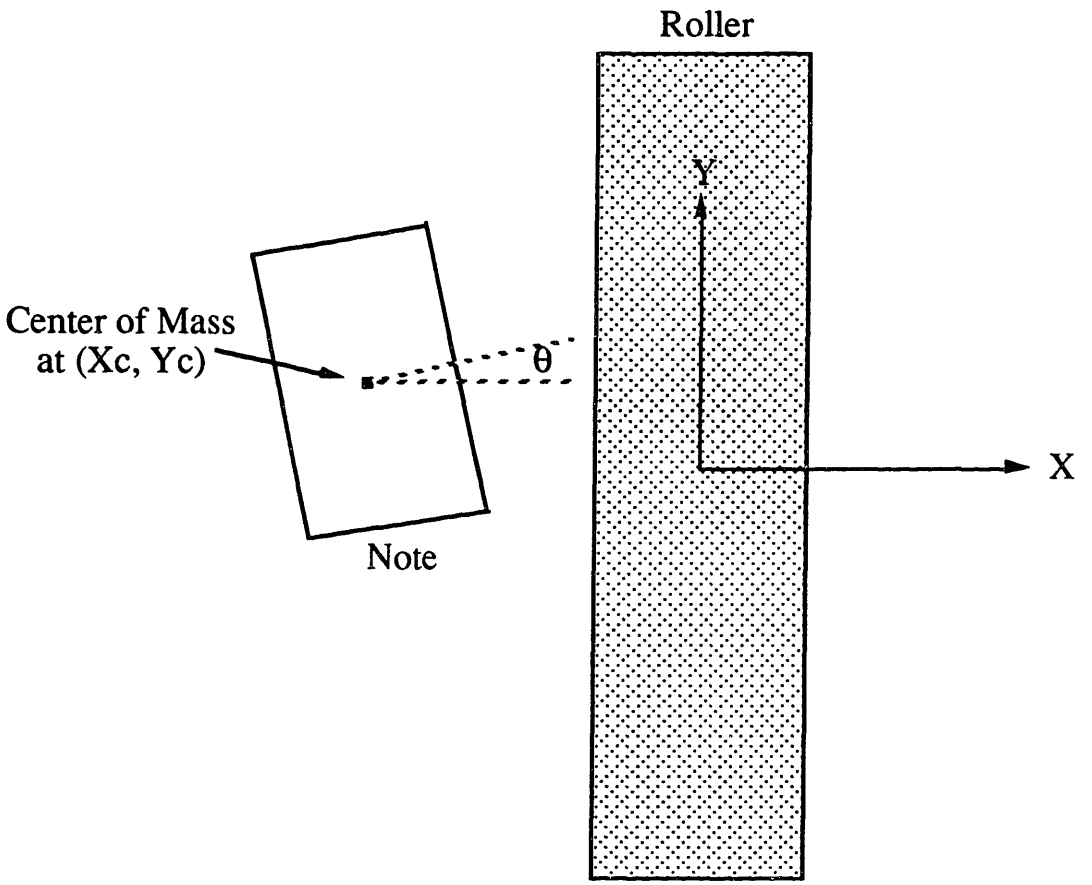


Figure C.1: Plan View of Feeder Model

The velocity of an arbitrary differential element, dA, of the yen's surface is now given. $\mathbf{r_{dA}}$ is the vector from the center of mass to dA:

$$\mathbf{V_{dA}} = \dot{X}_c\hat{\mathbf{i}} + \dot{Y}_c\hat{\mathbf{j}} + \dot{\theta} \times \mathbf{r_{dA}} \tag{C-1}$$

$\mathbf{r}_{dA}$ is given as:

$$\mathbf{r_{dA}} = (X - X_c)\,\hat{\mathbf{i}} + (Y - Y_c)\,\hat{\mathbf{j}} \tag{C-2}$$

$\dot{\theta} \times \mathbf{r_{dA}}$ is now be determined in (C-3), and substitution into (C-1) gives (C-4):

$$\dot{\theta} \times \mathbf{r_{dA}} = -\dot{\theta}(Y - Y_c)\,\hat{\mathbf{i}} + \dot{\theta}(X - X_c)\,\hat{\mathbf{j}} \tag{C-3}$$

$$\mathbf{V_{dA}} = \left[\dot{X}_c - \dot{\theta}(Y - Y_c)\right]\hat{\mathbf{i}} + \left[\dot{Y}_c + \dot{\theta}(X - X_c)\right]\hat{\mathbf{j}} \tag{C-4}$$

The local frictional force applied by the rollers always acts in the direction of the local relative velocity between the slipping roller surface and the moving yen element dA. The roller

surface velocity is equal to the final velocity of a perfectly fed note, $V_f\,\hat{\mathbf{i}}$, so the relative velocity is (ignoring the small component of roller velocity in the Z-direction):

$$\mathbf{V_{rel}} = \mathbf{V_f} - \mathbf{V_{dA}} = \left[V_f - \dot{X}_c + \dot{\theta}(Y - Y_c)\right]\hat{\mathbf{i}} - \left[\dot{Y}_c + \dot{\theta}(X - X_c)\right]\hat{\mathbf{j}} \qquad \text{(C-5)}$$

However, $\mathbf{V_{rel}}$ is normalized so it is a unit-directional vector of the frictional force on the element. A new vector, $\hat{\mathbf{V}}_{\mathbf{fric}}$, is defined as the local unit direction of the frictional force:

$$\hat{\mathbf{V}}_{\mathbf{fric}} = \frac{\mathbf{V_{rel}}}{|\mathbf{V_{rel}}|} = \frac{\left[V_f - \dot{X}_c + \dot{\theta}(Y - Y_c)\right]}{\sqrt{\left[V_f - \dot{X}_c + \dot{\theta}(Y - Y_c)\right]^2 + \left[\dot{Y}_c + \dot{\theta}(X - X_c)\right]^2}}\hat{\mathbf{i}} - \frac{\left[\dot{Y}_c + \dot{\theta}(X - X_c)\right]}{\sqrt{\left[V_f - \dot{X}_c + \dot{\theta}(Y - Y_c)\right]^2 + \left[\dot{Y}_c + \dot{\theta}(X - X_c)\right]^2}}\hat{\mathbf{j}} \qquad \text{(C-6)}$$

C.3 Feeder Gap Calculation

An expression for the gap between the rollers is computed to determine the forces on the note. Referring to Figure C.2, if H_u is the distance from the X-axis to the surface of the upper roller (for $-R_u \le X \le R_u$), and H_l is the corresponding distance to the lower roller (for $-R_l \le X \le R_l$):

$$H_l = \left(R_l + \frac{H}{2}\right) - R_l \cos\phi_l \qquad \text{(C-7)}$$

$$H_u = \left(R_u + \frac{H}{2}\right) - R_u \cos\phi_u \qquad \text{(C-8)}$$

Also:

$$\cos\phi_u = \frac{\sqrt{R_u^2 - X^2}}{R_u} \qquad \text{(C-9)}$$

$$\cos\phi_l = \frac{\sqrt{R_l^2 - X^2}}{R_l} \qquad \text{(C-10)}$$

Inserting equations (C-9) and (C-10) into (C-7) and (C-8):

$$H_u = \left(R_u + \frac{H}{2}\right) - \sqrt{R_u^2 - X^2} \qquad \text{(C-11)}$$

$$H_l = -\left(R_l + \frac{H}{2}\right) + \sqrt{R_l^2 - X^2} \qquad \text{(C-12)}$$

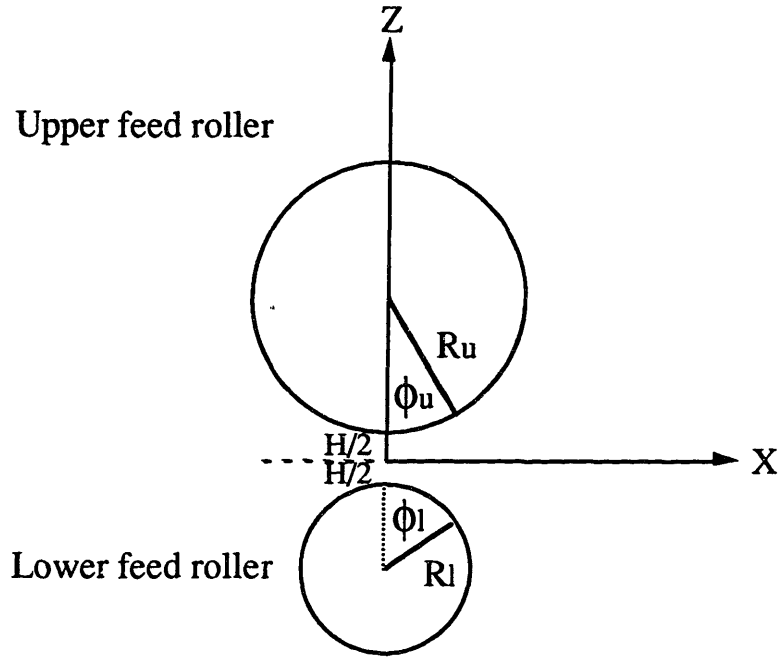


Figure C.2: Side View of Feeder Model

The total gap between the two rollers is H_u - H_l:

$$\Delta H(X) = R_u + R_l + H - \sqrt{R_u^2 - X^2} - \sqrt{R_l^2 - X^2} \qquad \text{(C-13)}$$

The remainder of this analysis assumes $R_u = R_l$. $\Delta H(X)$ becomes:

$$\Delta H(X) = 2\left(r - \sqrt{r^2 - X^2}\right) + H \qquad \text{(C-14)}$$

Acceptable values of X in (C-13) and (C-14) are those in which $-r < X < r$ and r is the smaller of R_u and R_l.

C.4 Yen Model

Referring to Figure A.1, when the note is perfectly fed (no skew), its width is aligned with the X-axis, its length is aligned with the Y-axis, and its thickness is aligned with the Z-axis. The yen's compressive behavior in the Z-direction is modeled in order to determine frictional forces from the rollers. A simple linear response is assumed, and all material deformation due to the frictional forces is neglected. δ is the strain in the Z-direction:

$$\delta(X) = \frac{\left(T - \Delta H(X)\right)}{T} \qquad \text{(C-15)}$$

$$\frac{\sigma_n(X)}{\delta(X)} = E = \text{note's modulus of elasticity} \tag{C-16}$$

$$\sigma_n(X) = \delta(X)E = \frac{(T - \Delta H(X))}{T} E \tag{C-17}$$

C.5 Motion of the Note

An expression for **dF**, the force on a differential element of the note's surface, can now be derived. μ is the coefficient of friction, which is here assumed constant over the yen:

$$\mathbf{dF} = \mu\, \sigma_n(X)\, \hat{\mathbf{V}}_{\mathbf{fric}}\, dX\ dY = \mu \frac{(T - \Delta H(X))}{T} E\, \hat{\mathbf{V}}_{\mathbf{fric}}\, dX\ dY \tag{C-18}$$

The total force on the note is calculated by integrating **dF**. F_x is the X-direction force, F_y is the Y-direction force, and R, the integration region, is the area of the note in contact with the rollers. The factor of two occurs because the note is pinched on both sides:

$$F_x = \iint_R 2\mu \frac{E}{T}\left(T - 2\left(r - \sqrt{r^2 - X^2}\right) - H\right) \frac{\left[V_f - \dot{X}_c + \dot{\theta}(Y - Y_c)\right]}{\sqrt{\left[V_f - \dot{X}_c + \dot{\theta}(Y - Y_c)\right]^2 + \left[\dot{Y}_c + \dot{\theta}(X - X_c)\right]^2}}\, dX\, dY \tag{C-19}$$

$$F_y = \iint_R -2\mu \frac{E}{T}\left(T - 2\left(r - \sqrt{r^2 - X^2}\right) - H\right) \frac{\left[\dot{Y}_c + \dot{\theta}(X - X_c)\right]}{\sqrt{\left[V_f - \dot{X}_c + \dot{\theta}(Y - Y_c)\right]^2 + \left[\dot{Y}_c + \dot{\theta}(X - X_c)\right]^2}}\, dX\, dY \tag{C-20}$$

Total torque on the note is taken about the center of mass:

$$\begin{aligned}\tau = &\iint_R -2(Y - Y_c)\mu \frac{E}{T}\left(T - 2\left(r - \sqrt{r^2 - X^2}\right) - H\right) \frac{\left[V_f - \dot{X}_c + \dot{\theta}(Y - Y_c)\right]}{\sqrt{\left[V_f - \dot{X}_c + \dot{\theta}(Y - Y_c)\right]^2 + \left[\dot{Y}_c + \dot{\theta}(X - X_c)\right]^2}}\, dX\, dY \\ &+ \iint_R -2(X - X_c)\mu \frac{E}{T}\left(T - 2\left(r - \sqrt{r^2 - X^2}\right) - H\right) \frac{\left[\dot{Y}_c + \dot{\theta}(X - X_c)\right]}{\sqrt{\left[V_f - \dot{X}_c + \dot{\theta}(Y - Y_c)\right]^2 + \left[\dot{Y}_c + \dot{\theta}(X - X_c)\right]^2}}\, dX\, dY\end{aligned} \tag{C-21}$$

The position and rotation of the note are obtained by the classical equations of motion. M is the mass of the yen and I is the yen's moment of inertia about an axis through its center of mass and perpendicular to the its surface:

$$\ddot{X}_C = \frac{F_x}{M} \tag{C-22}$$

$$\ddot{Y}_C = \frac{F_y}{M} \tag{C-23}$$

$$\ddot{\theta} = \frac{\tau}{I} \tag{C-24}$$

C.6 Example Solution of Feeder Equations

An analytical solution to the feeder problem is difficult even with this simple model because of the nonlinearity in the integrands and the time-varying limits of integration in equations (C-19) to (C-21), so a calculated solution is found in the same manner as that of the deskewer problem. First a spacewise integration is performed over the yen-roller contact area to determine the force on the bill, and then a time-domain integration is performed to find the motion of the bill over a small time interval. The fineness of the integration steps is adjusted until consistent results are obtained.

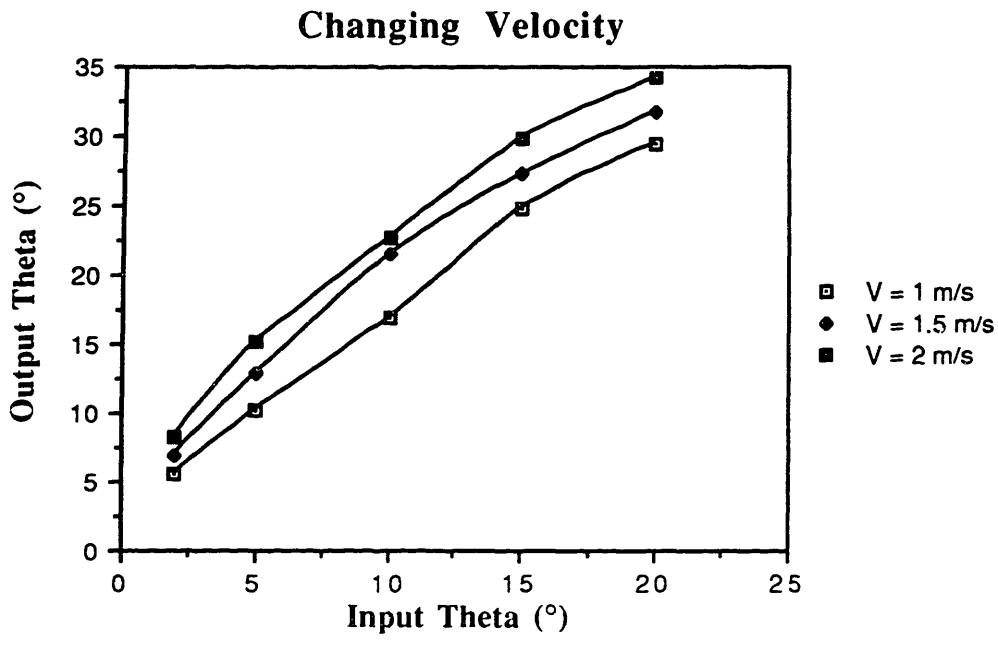


Figure C.3: Sample Graph of Feeder Simulation Results

Figure C.3 displays the input skew - output skew relation for a note that is accelerated from 0.1 m/s to final velocities from 1 to 2 m/s (the incoming speed is greater than zero so the note

moves into the rollers). The note properties are those given in Appendix A, the roller radius is 45 mm, and the roller gap is 0.09 mm. High-speed videotape of feeding indicates that Figure C.3 is qualitatively correct; for a fixed input skew, a note's output skew is larger at higher final velocities.

Appendix D

Implementing IBM-PC and PC/AT Interrupt Control

This appendix is a guide to using hardware interrupts on the IBM-PC and PC/AT, with example code written in Borland Turbo C. While coding details may differ in other implementations of C, the necessary steps should be identical. The PC/AT has 16 interrupt lines, as compared to eight in the original PC, but IRQ0 through IRQ7 are handled identically in both machines; the code presented below will run on either type of computer. For convenience they are both referred to as the "PC."

For additional information see [Dunford 1983] and [Sargent 1987].

D.1 General Information

1. A hardware interrupt line must be dedicated to the user's process; on the PC interrupts are triggered by a rising edge on one of the motherboard's eight *I*nterrupt *ReQ*uest lines (IRQ0 through IRQ7). A line that is not being used for another function (such as the system clock or serial port) must be chosen to avoid conflict because, for example, the computer cannot determine whether the user's external hardware or the parallel port has just brought IRQ4 high. They must each have a separate, dedicated line.

2. Various sources suggest that the user must initialize the 8259 interrupt control chip, but this only caused problems in the deskewer application. It appears that the chip is initialized upon startup, as might be expected given that it must handle keyboard input. The only required initialization is the enabling of the specific interrupts that the user intends to utilize. This is accomplished by setting bits in the 8259's interrupt mask register, in which a low bit means that the interrupt is enabled. Bit 0 corresponds to IRQ0, bit 1 to IRQ1, bit 2 to IRQ2, and so on. IRQ are enabled by logically ANDing the mask register with a byte in which the desired IRQ bits have been set to zero and all other bits have been set to one. For example, to enable IRQ3 and IRQ5 while leaving all other interrupts unchanged, AND the mask register with binary 11010111 (hex D7). This is detailed in Section D.2.

3. The interrupt can only access global variables; passing it local variables is impossible because the interrupt is asynchronously triggered by external events and cannot be directly called by other functions.

4. Writing to the screen during an interrupt gives unpredictable results. The `cprintf` function seems to work most reliably.

D.2 Initializing Interrupts

1. The user must first tell Turbo C that an interrupt function is to be used. This is done with an interrupt function prototype on the global level:

```
void interrupt your_interrupt_name_here(void);
```

2. Also on the global level, declare a pointer to the interrupt. This allows the address of the preexisting interrupt service routine (if there is one) to be restored when the program finishes.

```
void interrupt (*dummy_name_for_existing_interrupt) (void);
```

3. Due to a hardware quirk, the "interrupt number" used to initialize the interrupt is equal to the IRQ number plus eight. Thus, the interrupt number for IRQ5 is 13.

4. The initialization of the interrupt requires a standard segment of code. In order, with comments between statements:

```
disable();
```

Eliminates interrupt servicing during the initialization phase.

```
dummy_name_for_existing_interrupt = getvect(interrupt_number);
```

Sets a pointer to the existing interrupt routine. Remember: *interrupt_number* = IRQ number + 8.

```
setvect(interrupt_number, your_interrupt_name_here);
```

Sets the appropriate interrupt vector table entry to the address of the user's routine.

```
old_mask = inportb(0x21);
```

Saves the preexisting settings in the 8259's interrupt mask register.

```
outportb(0x21, old_mask & interrupt_mask);
```

Sets the mask bits in the 8259 to enable servicing of the user's interrupt. Recall that the interrupt is enabled if the bit is zero; for example, the mask that enables both IRQ7 and IRQ2 is binary 01111011 (hex 7B).

```
enable();
```

Allows interrupts to be serviced again.

5. As a recap, the complete initialization code is:

```
disable();
dummy_name_for_existing_interrupt = getvect(interrupt_number);
setvect(interrupt_number, your_interrupt_name_here);
old_mask = inportb(0x21);
outportb(0x21, old_mask & interrupt_mask);
enable();
```

D.3 Ending the Program

1. At the end of the program the preexisting interrupt service routine addresses should be restored. In order, with comments between statements:

```
disable();
```

Disables the servicing of interrupts during the restoration process.

```
outportb(0x21, old_mask);
```

Resets the bits in the 8259's mask register to their original state.

```
setvect(interrupt_number, dummy_name_for_existing_interrupt);
```

Resets the appropriate interrupt vector table entry to the address that existed before the user's routine was enabled.

```
enable();
```

Re-enables interrupt servicing.

2. The complete code for restoring the initial interrupts is as follows. It is placed at the end of the program:

```
disable();
outportb(0x21, old_mask);
setvect(interrupt_number, dummy_name_for_existing_interrupt);
enable();
```

D.4 Interrupt Functions

1. The user must supply an interrupt routine, which is almost identical to a standard C function. The declaration line contains the only significant difference. It must be:

```
void interrupt your_interrupt_name_here(void)
```

2. The first line in the interrupt function must be

```
disable();
```

which prevents other interrupts from being serviced during the execution of the user's routine. The last lines must be:

```
enable();
outportb(0x20,0x20);
```

The `enable` allows interrupts to be serviced again, while the `outportb` to 0x20 tells the 8259 that the interrupt routine has been completed.

3. The complete structure of the user's routine is :

```
void interrupt your_interrupt_name_here(void)
{
        User's in-function variable name declarations here
        disable();
        User's code here
        enable();
        outportb(0x20,0x20);
}
```

D.5 Sample Program

The following code is a simple, tested implementation of a complete interrupt-driven program in Turbo C. Every time IRQ3 is brought high the global variable `gNumber_3_counter` is incremented. The present value of `gNumber_3_counter` is asynchronously printed in the main loop of the program.

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>

#define  MASK 0xf7            /*  Mask for allowing IRQ3 to be serviced */
#define  IRQ3 11              /*  IRQ # + 8 = # used in setvect routine */

void interrupt My_interrupt_3(void);   /* User's IRQ3 routine prototype */
void interrupt (*Old_interrupt_3) (void);   /* Pointer to original IRQ3 */

int  gNumber_3_counter;  /* Global counter variable for number of IRQ3s */

main()
{
  unsigned char  Old_mask;                      /* Preexisting 8259 mask */

  disable();       /* Disable interrupt servicing during initialization */
  Old_interrupt_3 =  getvect(IRQ3);   /*  Save preexisting IRQ3 address */
  setvect(IRQ3,My_interrupt_3);   /* Set address of user's IRQ3 routine */
  Old_mask = inportb(0x21);           /*     Save preexisting 8259 mask */
  outportb(0x21, Old_mask & MASK);    /*                     Enable IRQ3 */
  enable();                           /*  Re-enable interrupt servicing */

  _setcursortype(_NOCURSOR);                    /*  Turn off the cursor */
  clrscr();
  gNumber_3_counter = 0;                        /* Reset global counter */
  printf ("Number of times interrupt 3 has occurred = ");

  while (!kbhit()){              /* Main loop - print gNumber_3_counter */
    gotoxy(42,1);
    cprintf ("%d",gNumber_3_counter);    /* Hit any key to exit program */
  }

  _setcursortype(_NORMALCURSOR);       /*             Regain the cursor */
  disable();                           /*     Stop servicing interrupts */
  outportb(0x21,Old_mask);             /*         Restore original mask */
  setvect(IRQ3, Old_interrupt_3);      /*       Restore original vector */
  enable();                            /* Re-enable interrupt servicing */
}

void interrupt My_interrupt_3(void)   /* Int. function declaration line */
{
  disable();                           /*    Disable interrupt servicing */
  gNumber_3_counter ++;
  enable();                            /*     Enable interrupt servicing */
  outportb(0x20,0x20);                 /* Tell 8259 that routine is done */
}
```

Appendix E
Deskewer Control Program

The following sections list the code of the two headers and four segments that comprise the deskewer's control program. All code is written in Borland Turbo C for the IBM-PC.

Constant.h declares the constants used in all program sections.

Globals.h contains global declarations, such as the skew and control time structures that are accessed by the interrupts. Note that there is an alternative global include file, "globextn.h", which is included in all program segments except main.c. It has the same variables as globals.h, but all are declared as external (because they are declared and initialized in main.c by globals.h).

Main.c clears variables through a call to reset.c, contains the event loop that processes user input, and handles display of the global variables. It also writes the control time structure to disk upon program exit.

Init.c handles input of the control time structure, initializes interrupts, and initializes the Am9513 timer chip.

Reset.c clears variables and measures belt velocity through an A/D chip attached to a tachometer.

Interrupt.c contains code for the two interrupts used in the deskewer control. IT3 (called by IRQ3) measures the input skew, records it in the global skew data structure, and programs the solenoid wait and fire times. IT5 (called by IRQ5) measures the outgoing skew, saves it in the global skew-data structure, and performs the tuning on the control time bins.

All interrupt and training algorithm code was written by Ross Levinsky. Matthew Selick. provided assistance with Turbo C, display routines, modularization, and the technical aspects of interrupts.

E.1 Constant.h

```
#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <fcntl.h>
#include <sys\stat.h>
#include <io.h>

#define   INTR_3       11
#define   INTR_5       13
#define   BA           0x240 /*Base address and ports of I/O board*/
#define   PPI_A (BA+0)
#define   PPI_B (BA+1)
#define   PPI_C (BA+2)
#define   CONFIG       (BA+3)
#define   DATA_P       (BA+4)
#define   COMM_P       (BA+5)
#define   TOTAL 10000      /* Number of bills in array */
#define   TRUE         1
#define   FALSE 0
#define   MASK  0xd7   /* Enable interrupts 3 and 5 */
#define  AVERAGE_COUNT 30   /*Number of samples of velocity to average*/
#define  SKEW_NODES     200 /* Number of skew bins */
#define  VEL_NODES      10  /* Number of velocity bins */
#define  NODE_SIZE      8   /* Bytes for each control action variable */
#define  FILE_NAME    "NET.DAT"
```

E.2 Globals.h

```
void interrupt it3();     /* Declare interrupt handlers, functions */
void interrupt it5();
void interrupt (*old3)(void);
void interrupt (*old5)(void);

struct DataType
{
    int   in;
    int   out;
} gData[TOTAL];

/* This is the definition of the bin structure */
/* The first index is for theta, while the second is velocity. */

union
{
    double         action [SKEW_NODES][VEL_NODES];
    unsigned char  outside[NODE_SIZE*SKEW_NODES*VEL_NODES];
} gControl;

unsigned char   gVelocity;
int             gCount3=0;
int             gCount5=0;  /* Number of int 3, int 5 thrown */
int             gFile;            /* Used to save bins     */
```

E.3 Main.c

```
/*****************************************************/
/*                                                   */
/*                    Main Program                   */
/*                                                   */
/*                          main.c                   */
/*                                                   */
/*****************************************************/
/*A note to anyone who wants to modify this program: */
/*                                                   */
/*  All of the outportb statements to the DATA_P and */
/*  COMM_P ports are taken from the Am9513 data sheet*/
/*                                                   */
/*     YOU CANNOT USE THIS CHIP WITHOUT THE SHEET!   */
/*         DO NOT GUESS AT NEW VALUES!               */
/*                                                   */
/*****************************************************/


#include "constant.h"
#include "globals.h"

/********** main **********/
int main()
{
    int   i;                /* For/next loop counter    */
    int   done=FALSE;       /* True when done           */
    int   offset=0;
    int   hold=FALSE;       /* TRUE for screen hold     */
    int   handle;


    if (!Initialize()) return 1;  /* Exit errorlevel 1   */

    _setcursortype(_NOCURSOR);    /* Turn off the cursor */

    gCount3=gCount5=0;

    done=!ResetMachine();   /* Reset data tables and get velocity */

    while(!done)
    {
          if (kbhit())
                switch (getch())
                {
                      case 'R':
                      case 'r':
                            done=!ResetMachine();
                            break;
                      case 'D':
                      case 'd':
                            offset += 40;
                            if (offset>=TOTAL)
                                  offset=0;
                            hold=TRUE;
```

```
                        gotoxy(1,23);
                        cprintf("HOLD");
                        break;
                    case 'U':
                    case 'u':
                        offset -=40;
                        if (offset<0)
                            offset=TOTAL-40;
                        hold=TRUE;
                        gotoxy(1,23);
                        cprintf("HOLD");
                        break;
                    case 'Q':
                    case 'q':
                    case 3:
                    case 27:
                        done=TRUE;
                        break;
                    case 'H':
                    case 'h':
                        hold=TRUE;
                        gotoxy(1,23);
                        cprintf("HOLD");
                        break;
                    case 'O':
                    case 'o':
                        hold=FALSE;
                        gotoxy(1,23);
                        cprintf("    ");
                        break;
                }

        for (i=offset;i<offset+40;i++)
        {
          gotoxy(40*((i-offset)/20)+1,((i-offset)%20)+2);
          if (i==gCount3)
            cprintf("<%6d> %6d %6d",i,gData[i].in,gData[i].out);
          else
            cprintf(" %6d  %6d %6d",i,gData[i].in,gData[i].out);
        }
        if (!hold)
          offset=(gCount3/40)*40;
    }

    _setcursortype(_NORMALCURSOR);      /*Restore the cursor */

    disable();/*Disable the interrupts as we are restoring them */

    outport(0x21,inport(0x21)|(~MASK)); /*Mask interrupts 3 & 5 */

    setvect(INTR_3,old3);   /* Restore the interrupts */
    setvect(INTR_5,old5);

    enable(); /* Re-enable intrpts. so the system will not crash */

    gotoxy(1,24);
    cprintf("\nThat's all, folks... \n\n");
```

```
      outportb(COMM_P,0xE3); /* Clear timers 3, 4's outputs */
      outportb(COMM_P,0xE4);

      gCount3=0;

      if
((handle=open(FILE_NAME,O_CREAT|O_TRUNC|O_BINARY|O_WRONLY,S_IREAD|S_IWRITE)
)==-1)
            {
                  printf("Error opening file");
            }
  if (write(handle,gControl.outside,NODE_SIZE*SKEW_NODES*VEL_NODES)==-1)
            {
                  printf("Error writing file");
            }
                  if (close(handle)==-1)
            {
                  printf("Error closing file");
            }

      return 0;     /* Return Dos errorlevel 0 */
  }
```

E.4 Init.c

```
/*****************************************************************/
/*                                                               */
/*            Initializing procedure                             */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
/* This initializes our nodes if there's no file named net.dat.  */
/* They are evenly spread throughout the interesting region of   */
/* state space. Velocity and skew are normalized, as is output   */
/* timer information. The user must write the initialization     */
/* code for each specific application.                           */
/*****************************************************************/

#include "constant.h"
#include "globextn.h"

int Initialize(void)
{
    int   i,j;
    int   handle;

    /* Check to see if net.dat exists, and use it if it does.    */

    if ((handle=(open(FILE_NAME,O_BINARY|O_RDONLY)))==-1)

          for (i = 0; i<SKEW_NODES; i++)
            for (j = 0; j<VEL_NODES; j++)
                 gControl.action[i][j] = 10.0;

          else

          {
                if
(read(handle,gControl.outside,NODE_SIZE*SKEW_NODES*VEL_NODES)==-1)
                      {
                      perror("Error reading file");
                      return FALSE;
                      }
                if (close(handle)==-1)
                      {
                            perror("Error closing file");
                            return FALSE;
                      }
          }

    /* Disable interrupts during the setup of the interrupt */
    /* vectors & masks.                                     */

    disable();

    old3 = getvect(INTR_3);        /*    Save the old vectors    */
    old5 = getvect(INTR_5);
```

```
 setvect(INTR_3,it3); /* Set the new vectors for our own */
                      /* service routines                 */
 setvect(INTR_5,it5);

 outportb(0x21,inportb(0x21)&MASK);  /*Un-mask interrupts 3&5 */

 enable(); /* Re-enable interrupts so our program can operate */

/* Initialize the Am9513 timer so we can measure skew */

 outportb(COMM_P,0xFF); /* Master reset */

 outportb(COMM_P,0x5F); /* Load all counters */

 /*******************************************************/
 /* 1,2,4 are mode B. 3 is A. Counters 1 and 2 are used  */
 /* to measure skew. Counter 3 is the wait timer, and it */
 /* gates Couner 4, which is the fire timer. The freq.   */
 /* source is F3, at 50 khz.                             */
 /*******************************************************/

 outportb(COMM_P,0x17); /* Point to Master Mode register */
 outportb(DATA_P,0x00); /* Master Mode LSB */
 outportb(DATA_P,0xC0); /* Master Mode MSB. No autocycling. */

 outportb(COMM_P,0x01); /* Counter 1 mode control register. */
 outportb(DATA_P,0x08); /* Counter 1 mode LSB */
 outportb(DATA_P,0xAD); /* Counter 1 mode MSB */

 outportb(COMM_P,0x09); /* Counter 1 load register. */
 outportb(DATA_P,0x00); /* Counter 1 load LSB */
 outportb(DATA_P,0x00); /* Counter 1 load MSB */

 outportb(COMM_P,0x02); /* Counter 2 mode control register. */
 outportb(DATA_P,0x08); /* Counter 2 mode LSB */
 outportb(DATA_P,0xAD); /* Counter 2 mode MSB */

 outportb(COMM_P,0x0A); /* Counter 2 load register. */
 outportb(DATA_P,0x00); /* Counter 2 load LSB */
 outportb(DATA_P,0x00); /* Counter 2 load MSB */

 outportb(COMM_P,0x03); /* Counter 3 mode control register. */
 outportb(DATA_P,0x02); /* Counter 3 mode LSB */
 outportb(DATA_P,0x0D); /* Counter 3 mode MSB */

 outportb(COMM_P,0x04); /* Counter 4 mode control register. */
 outportb(DATA_P,0x02); /* Counter 4 mode LSB */
 outportb(DATA_P,0xCD); /* Counter 4 mode MSB */

 outportb(COMM_P,0x63); /* Load and arm counters 1 and 2 */

 outportb(CONFIG,0x91); /* Low 4 of C are input, hi 4 are  */
                        /* output; A is input from encoder */

 outportb(COMM_P,0xE3); /* Clear timers 3, 4's outputs */
 outportb(COMM_P,0xE4);
 return TRUE;
}
```

E.5 Reset.c

```
/*****************************************************/
/*                                                   */
/*     Resets the machine and the data structures    */
/*                                                   */
/*            reset.c                                */
/*                                                   */
/*****************************************************/

#include "constant.h"
#include "globextn.h"


int ResetMachine()

{
    int   i,j,k;
    int   velocityTotal=0;
    int   velocity,theta;
    double temp;

    if (gCount3 != gCount5) printf
          ("The interrupt counters are unequal...");

    clrscr();
    gotoxy(1,1);
    cprintf(" number      in    out");
    gotoxy(41,1);
    cprintf(" number      in    out");
    gotoxy(1,24);
    cprintf("<Q> quit   <U> up   <D> down   <R> reset   <H> hold
       <O> off hold");

    for (i=0;i<TOTAL;i++){
          gData[i].in=0;
          gData[i].out=0;
    }

    for (i=0;i<AVERAGE_COUNT;i++){

          int one_sample;

      Velocity_calc:

          outportb(PPI_C,0x00); /* Bring Read/Write low on A/D */
          while ((inportb(PPI_C) & 0x01) != 1);
          outportb(PPI_C,0x10);  /* Bring Read/Write high on A/D */
          one_sample = inportb(PPI_A);

      if (one_sample==128) goto Velocity_calc;
                                /* 128 = bogus value, so kill */
      else velocityTotal += one_sample;
    }

   gVelocity= (int) (velocityTotal/AVERAGE_COUNT);
```

```
    if (gVelocity==0) gVelocity=1;

    gotoxy(25,19);
    cprintf("Velocity = %u  ",gVelocity);
    gCount3=gCount5=0;


    outportb(COMM_P,0xE3); /* Clear timers 3, 4's outputs */
    outportb(COMM_P,0xE4);

    return TRUE;
}
```

E.6 Interpt.c

```
/*********************************************/
/*                                           */
/*            interrupt handlers             */
/*                                           */
/*            interpt.c                      */
/*                                           */
/*********************************************/


#include "constant.h"
#include "globextn.h"

#define  ONE_OVER_BETA       0.33 /* This is the training rate */

/********** it3 **********/
void interrupt it3(void)
{
   int i;  /* For-next loop variable */

    double wait_time,fire_time,unsigned_fire; /* Used before lsb,msb */

    unsigned char wait_lsb, wait_msb; /* For programming wait and */
    unsigned char fire_lsb, fire_msb; /* fire timers.             */

    int velocity, theta;              /* Normalized vel. and skew */
    disable();  /* Disable interrupts during the service routine */

    outportb(COMM_P,0xA1);  /* Hold contents of counter 1 */
    outportb(COMM_P,0x11);  /* Select counter 1 hold register */

    if (gCount3>=TOTAL) gCount3=0;
    gData[gCount3].in = inportb(DATA_P) + 256*inportb(DATA_P);
    outportb(COMM_P,0x61);  /* Reload and rearm counter 1 */
    if ((inportb(PPI_C)&0x02)==2)
          gData[gCount3].in*=-1;


    wait_time =
         0.4*(gVelocity*gData[gCount3].in)*(gVelocity*gData[gCount3].in);
    wait_time = .8*(sqrt(wait_time + 2.5e9))/gVelocity;
    wait_time = wait_time - gData[gCount3].in + 6e5/gVelocity;

    wait_lsb = (unsigned char) ((unsigned int)wait_time)%256;
    wait_msb =  wait_time/256;

    outportb(COMM_P,0xE3);  /* Clear timers 3, 4's outputs */
    outportb(COMM_P,0xE4);

   /* Set skew direction here... */

    outportb(COMM_P,0x0B);     /* Set wait time on timer 3 */
    outportb(DATA_P,wait_lsb);
    outportb(DATA_P,wait_msb);
```

```
    /* This section calculates control value from velocity and a skew */

    velocity = (int)  VEL_NODES*(gVelocity/255.0);
    theta = (int)  (0.5*(SKEW_NODES-1)*((gData[gCount3].in/5000.0)+ 1));


    /* For both directions of skew */

    unsigned_fire = fire_time= gControl.action[theta][velocity];
    if (fire_time<0) fire_time = 0;

    /* Kill fire time if skew is less than 8° */

    if (gData[gCount3].in < (58350/gVelocity)) fire_time = 0;

    fire_lsb = (unsigned char) ((unsigned int)fire_time)%256;
    fire_msb = (unsigned char) ((fire_time)/256);

    outportb(COMM_P,0x0C);      /* Set fire time on timer 4 */
    outportb(DATA_P,fire_lsb);
    outportb(DATA_P,fire_msb);

    outportb(COMM_P,0x6C);      /* Load and arm timers 3 and 4 */

    gCount3++;

    enable();   /* Re-enable interrupts */

    /* Tell the interrupt chip that we have processed the interrupt */

    outportb(0x20,0x20);
}


/********** it5 **********/
void interrupt it5(void)
{
    int   i,j,k;
    int   velocity,theta;
    double temp;

    disable();  /*    Disable interrupts during the service routine */

    outportb(COMM_P,0xA2);  /* Hold contents of counter 2 */
    outportb(COMM_P,0x12);  /* Select counter 2 hold register */

    gData[gCount5].out = inportb(DATA_P) + 256 * inportb(DATA_P);
    outportb(COMM_P,0x62);  /* Reload and rearm counter 2 */


    if ((inportb(PPI_C)&0x04)==4)
          gData[gCount5].out*=-1;

    /* Learning algorithm here */
```

```
    velocity = (int) VEL_NODES*(gVelocity/255.0);

    theta = (int) (0.5*(SKEW_NODES-1)*((gData[gCount5].in/5000.0)+ 1));

    /* For both directions of skew */

    temp = fabs(gData[gCount5].out)*ONE_OVER_BETA;

    if (((long)(gData[gCount5].in)*(long)(gData[gCount5].out))<0)
          temp *= -1;

    gControl.action[theta][velocity] +=  temp;

    if (gControl.action[theta][velocity] <  0)
                gControl.action[theta][velocity] =10;
    /* No negative control actions */

    gCount5++;

    enable();   /*    Re-enable interrupts    */

    /*Tell the interrupt chip that we have processed the interrupt*/

    outportb(0x20,0x20);
}
```

References

Bessette, M.D., "Designing Reliable Paper-Handling Systems," *Machine Design*, pp. 54-58 (9 May 1985).

Dooley, G., and D. Szybist, *TC-24 User's Guide and Short Course on the Am9513 Timer*, Second Edition, Real Time Devices Inc., State College, PA.

Dunford, C., "Interrupts and the IBM PC, Part 1," *PC Tech Journal*, pp. 173-199 (November/December 1983).

Henrici, P., *Elements of Numerical Analysis*, John Wiley and Sons (1964).

Kotovsky, J., *Design of a De-Skewing System for Automatic Bill Handling Machines*, Mechanical Engineering S.B. Thesis, MIT, Cambridge, MA (June 1990).

Rudin, W., *Principles of Mathematical Analysis*, Third Edition, McGraw-Hill (1976).

Sargent, M., and R. L. Shoemaker, *The IBM PC From The Inside Out (Revised Edition)*, Addison-Wesley (1987).

Slotine, J-J. E., and L. Weiping, *Applied Nonlinear Control*, Prentice-Hall (1991).

Soong, T-C., and C. Li, "On the Unbonded Contact Between Plates and Layered Cylinders," *Journal of Applied Mechanics*, Vol. 47, pp. 841-846 (December 1980).

Soong, T-C., and C. Li, "The Steady Rolling Contact of Two Elastic Layer Bonded Cylinders With a Sheet in the Nip," *International Journal of Mechanical Science*, Vol. 23, pp. 263-273 (1981).

Soong, T-C., and C. Li, "The Rolling Contact of Two Elastic-Layer-Covered Cylinders Driving a Loaded Sheet in the Nip," *Journal of Applied Mechanics*, Vol. 48, pp. 889-894 (December 1981).

Strang, G., *Introduction to Applied Mathematics*, Wellesley-Cambridge Press (1986).

Wong, C.J., "Applications of Nonlinear Finite Element Method to Contact Problems and Paper Handling Problems," *Computers and Structures*, Vol. 19, No. 3, pp. 315-320 (1984).

Wong, C.J., "Analysis of Some Paper Handling Problems By a Three-Dimensional Nonlinear Finite Element Algorithm," *Computers and Structures*, Vol. 22, No. 6, pp. 889-903 (1986).