Web Dynamics:
Modeling, Simulation, and Experimentation

by

Clara C. Yang

B.S., Mechanical Engineering (1995)

Massachusetts Institute of Technology

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

May 1996

Signature of Author .....
Department of Mechanical Engineering
April 26, 1996

Certified by .....
Kamal Youcef-Toumi
Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted by .....
Ain A. Sonin
Chairman, Department Committee on Graduate Students

Web Dynamics:
Modeling, Simulation, and Experimentation

by

Clara C. Yang

Submitted to the Department of Mechanical Engineering
on April 26, 1996 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

Manufacturing lines which handle thin webs of material often encounter dynamic overshoots and oscillations which result in undesirable failures. These failures generally occur during the dynamic states of line start-up, shutdown, and splices. Splices introduce a new roll of web material when the old roll runs out. In order to improve performance, it is desired to minimize overshoots and reduce settling time. However, on-line testing is costly both in terms of capital and time. Computer modeling and simulation would allow for easier, faster, and cheaper design changes and experimentation.

Much of the difficulty lies in the modeling and model validation. Based on previous work, an extensive model was built using Extend™ software (a dynamic modeling software package by Imagine That). Code for the model utilizes calculations and web handling theory, incorporating the physics of the web and the system elements that interact with the web. On-line sensors collected data which helped to confirm the validity of the model. This model, a worst-case scenario, was used to pinpoint possible areas of control and design improvements for the web handling system. The model can also be quickly adapted to different web paths, allowing for widespread use with minimal training.

Thesis Supervisor: Kamal Youcef-Toumi, Sc.D.

Title: Associate Professor of Mechanical Engineering

Acknowledgments

First, I want to thank Dick, Mark, Bub, Don, and Prof. Youcef-Toumi. Without them, this thesis would never have been possible. This thesis also would not be possible without the support and encouragement of my parents, sister, friends, and of course, Brian. Last, but not least, thank you, Julie, for always letting me usurp your computer and for voluntarily taking time out of your busy schedule to read and edit my thesis. Oh, and JP, I think I won the bet.

Table of Contents

List of Figures

1 Introduction

A web handling system is generally used in manufacturing systems that require the handling of continuous sheets of usually thin material called webs. Web handling systems consist of numerous elements which help to move and control the web. Over the years, much has been studied and perfected under steady state conditions. However, the interrelated elements create an extremely complex dynamic system. Problems with web tension and side-to-side alignment or tracking are expressed mainly during the dynamic states of start-up, shutdown, and splice (material changeover). For example, both tension and speed exhibit overshoots and oscillation. Tension spikes or overshoots are believed to contribute to mistracking and failures, while oscillations effect the quality and consistency of the end product. Eliminating dynamic failures significantly increases overall line efficiency.

Dynamic problems are often difficult to rectify; they usually require many test trials over a range of controls and mechanical strategies. It is possible to reduce the time and capital investment necessary to test different solutions through computer modeling and simulation.

This project began as a part of a reliability improvement plan. The goal was to produce a dynamic model of the web path using Extend[1], a dynamic modeling software package, validating it with data taken from the physical system. The model would then be used to test possible large-scale improvements to the control strategy or physical system in order to eliminate dynamic problems. It also investigated whether modeling work for one material web path would assist in similar models for various web paths across a number of materials and products. The specific problem which this thesis addresses was instances in which dynamic tension spikes were leading to side-to-side tracking problems while the material unwound. Side-to-side alignment of the web is critical, especially when two webs are being combined. If the two webs are not aligned properly, the end product will not meet specifications. Thus, sensors placed in several locations along the web path ensure that the movement of the web is limited.

Web handling systems consist of several basic components including idlers, driven s-wraps, forty-five degree turn bars, unwind rolls, and dancers, which are tension controlling elements. Each plays a critical role in controlling the continuous web's tension, velocity, and position. Idlers are used to guide the web and contain low friction bearings to minimize energy loss. Driven s-wraps consist of two driven rolls which add energy to the web. The web is

[1] Software package developed by Imagine That. Version 3.1, 1995 .

wrapped around the rolls in the form of an "S" as seen in Figure 1. Stationary forty-five

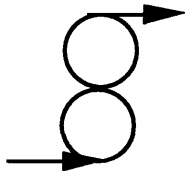

Figure 1: Diagram of s-wrap. The web follows the arrows around the right half of the bottom roll and the left side of the top roll, exiting in the direction of the top arrow.

degree turnbars are covered with a low friction coating and are positioned at a forty-five degree angle with respect to the incoming web but generally alter the web path by ninety degrees. Unwind rolls are basically the material source, or originating point, for the web. Though simple in appearance, dancers are actually quite complex in function. They are usually a collection of idlers that are set to maintain a certain force or pressure, pivoting if the tension in the web running through these idlers is lower or higher than the set force. Thus, dancers are a very effective means of regulating tension in the web. Web handling theory defines web handling as a mass flow conservation issue where the material mass entering each element also exits the element at the same rate. Thus, for steady-state analysis, the system is fairly simple to understand and predict through basic stress-strain relations. In the dynamic realm, analysis and simulation becomes increasingly complex.

Dancers can create changes in the web path, continually altering the distance between elements (the span length). Dancers also introduce control systems that react to any deviations from target web tensions. Dynamically, tension changes generate inputs for the control system as well as changes to span lengths, complicating the basic mass flow calculations. This is discussed in detail in Chapter 2.

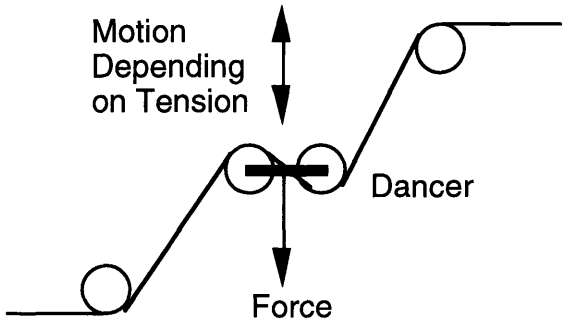

Figure 2: Schematic of dancer.

This thesis specifically examined dynamic problems on the web path that were causing failures, within a few seconds of line start-up or of material splicing from an expiring roll to a full roll. This problem is quickly illustrated through observation of the manufacturing line or web handling system. Sensors located at various critical points on the web path automatically stop the line when the set control limits are exceeded. Ideally, all stops should be eliminated. Modeling would be a step toward this goal. Thus, project objectives included understanding the dynamic system, developing a viable

model, and exploring a new technique to assist future dynamic models of web handling systems.

Extensive research has been done to better understand the dynamics of web handling systems. There are a number of software packages available for dynamic modeling purposes. Oklahoma State University (OSU) has a Web Handling Research Center that continually tries to better model the physics and better predict the uncertainties. The researchers at OSU are developing and improving a computer-based analysis program for multi-span web transport systems called WTS[2]. They have developed and tested extensive modeling techniques and equations for numerous web handling cases. For details, please refer to the Web Handling Research Center's WTS 6.5 User's Guide. Many of the equations and models developed by OSU are high-order, highly complex equations that are often unpractical for general use in corporate settings where it is more important to obtain quick estimations than finely-tuned analyses. However, their work is an effective resource for dynamic modeling of web handling systems. Other web handling research includes thesis work done by Manuel Jaime. His Master's thesis details theoretical models for dynamic situations such as start-up, shutdown, and splices. He used bond graph techniques, breaking down and incorporating the

[2] WTS 6.5 User's Guide, OSU Web Handling Research Center, June 1995.

effective inertias, springs, dampers, and energy sources for each element involved[3].

A practical, easy-to-use modeling technique was still necessary for faster and better design and improvement capabilities. Extend was chosen mainly because previous work and development for similar modeling needs had proven its ease of use and low cost. Because of the extensive use of web handling systems over time, much research has been invested in understanding its physics and improving handling and production. Based on previously developed equations characterizing web systems, code was written on the Extend platform, using MOD-L language to simulate the problematic web path. All code for the entire web path was written in one block because of the computing overhead and software sequencing problems associated with multiple blocks. With multiple blocks, each block represented a different element along the web path. For example, one block would represent a set of idlers and contain all the calculations necessary to simulate web flowing across the idler. In order to calculate in the proper sequence, blocks had to be visually placed in the proper position from left to right which became increasingly difficult as the size of the model grew. Additionally, since each block functioned independently, combining large numbers of blocks meant unnecessarily repeating set-up calculations.

---

[3] Jaime-Esqueda, Manuel, Dynamic Modeling, Reliability Analysis, and Control of Startup Transients in High Speed Web Handling Equipment, M.I.T., May 1994.

This thesis begins with the web handling theory on which the modeling was based. Comparing the model simulations to data collected sensors placed on the web handling system assisted in model validation. Once the model was confirmed, possible improvements and changes could be simulated and tested for improved dynamic response. The model was then adapted for two other cases to show its flexibility and reconfirm its advantages. Finally, some recommendations on further research and improvement needs for both the model and testing instrumentation are addressed.

2 Modeling Methods

There are a few different methods of modeling web handling systems. Two common approaches either focus on the web and its movement or focus on each element that contacts the web. Concentrating on the web helps to pinpoint speed and tension problems in the web. The exact code used for modeling each span and element is detailed in Appendix B.

2.1 Web Span Calculations

Web handling theory begins with mass flow equations. For any given period of time, the amount of material 'entering' an element, idler, turn bar, or driven s-wrap, equals that which is 'exiting'. With zero web tension, the web length between two elements is called the unstretched length. Under increasing tension, this unstretched length of material stretches and becomes longer. Thus, for a given amount of material to pass through an element per unit time, the linear speed of the web must increase with increasing tension. Likewise, if the speed of two adjacent elements begins to differ, the tension of the web between the elements changes. For example, a web flows from element A to element B which are both rotating at the same linear velocity. Then the velocity of element B begins to increase. This results in an increase in tension for the web section between elements A and B.

Mass balance and stress-strain equations explain the relationship between tension, material modulus, stretch ratio (the ratio between the stretched length and the unstretched length), stretched length, and velocity.

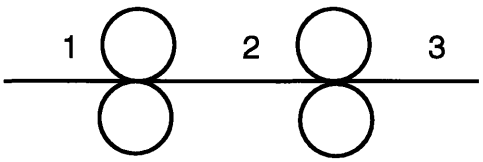

Figure 3: Web Flowing Through Two Elements.

m=mass
A=area
v=velocity
E=modulus
l=length
T=tension
ε=strain
σ=stress
ρ=density

$$\dot{m}_1 = \dot{m}_2 = \dot{m}_3 \tag{2.1}$$

$$\rho_1 A_1 v_1 = \rho_2 A_2 v_2 = \rho_3 A_3 v_3 \tag{2.2}$$

$$\frac{m_1}{A_1 l_1} A_1 v_1 = \frac{m_2}{A_2 l_2} A_2 v_2 = \frac{m_3}{A_3 l_3} A_3 v_3 \tag{2.3}$$

Constant mass gives:

$$\frac{v_1}{l_1} = \frac{v_2}{l_2} = \frac{v_3}{l_3} \tag{2.4}$$

Thus, as length increases, velocity must increase proportionally.

Stress-strain relation is expressed by:

$$E = \frac{\sigma}{\varepsilon} \tag{2.5.1}$$

Multiplying both sides by area gives:

$$E_w = \frac{T}{\varepsilon} \tag{2.5.2}$$

Where E_W is total web modulus; T is tension and:

$$\varepsilon = \frac{\Delta l}{l}$$

Thus:

$$E_w = \frac{T}{\frac{\Delta l}{l}} = \frac{Tl}{\Delta l} \tag{2.5.3}$$

If $\Delta l = l_S - l$ and $l = 1$ (unit length):

$$E_w = \frac{T}{l_s - 1} \text{ or } l_s = \frac{T}{E_w} + 1 \tag{2.5.4}$$

The stretched length can now be easily calculated if the tension is known. Note that the total web modulus, E_w, is a known material property. It is determined by the material modulus of the web multiplied by the cross-sectional area of the web.

Because web can only be pulled, span calculations in the Extend model begin with the span farthest downstream. The pull effect then propagates, imposing tension and velocity on the upstream spans.

2.2 Element calculations

Idlers, turnbars, and dancers are non-driven elements; they are not connected to any outside energy source such as a motor. Idlers are low friction, low inertia rolls that rotate with the moving web. Turnbars are low friction, stationary bars that are generally used to change the direction of the web. Dancers consist of sets of idlers that move with changes in web tension.

Therefore, the dancers' inertia, internal drag, and frictional forces take energy away from the moving web, slowing it down. In the case of dancers, they usually also have an air cylinder set to maintain a predetermined pressure (thus web tension) and position. On the other hand, driven s-wraps generally add energy to the web. S-wraps are used to set the speed of the web at a given point in the web path. This is critical for phasing and stretch issues further downstream.

Before most of the web and dancer calculations can be made, some key pieces of information must be gathered. Dimensions and inertias of all idlers, s-wraps, and the unwind as well as the friction coefficient and diameter of the turnbars are critical to tension and velocity calculations. The length of each web span is also required for stretch calculations which then determine tension in each span.

2.2.1 Inertias

The inertias were calculated directly off assembly drawings. Idler inertias were relatively easy because of their simple symmetric geometry and rotation about their central axis. Dancers were more complex because of their complex geometry, numerous parts, and rotation about its end. Each dimension had to be considered, and each weight was calculated based on volume and material density. Detailed calculations can be found in Appendix A.

2.2.2 Turnbars

Surprisingly, the seemingly simple stationary turnbars were the most difficult element to simulate. Turnbars create a tension drop, easily calculated through the capstan formula. Figure 4 shows the side view of web flowing over a turnbar. The capstan formula is given by equation 2.7. Physically, this is due to the fact that the web is being pulled from downstream. The friction across the turnbar causes a loss in energy, seen through the tension drop. The higher the friction and the greater the area of contact between the web and the turnbar, the larger the difference between the higher downstream tension and the lower upstream tension. With respect to dancers, however, the system is not simple to model. This simple calculation will give accurate tension results, but in a dynamic system, tension, velocity, stretch, and time are all interdependent. Using a single tension calculation adversely affects all of the other calculations as well as the tension calculation in later time steps. This is due to the close correlation between tension, velocity, and stretch as well as to the effect that each web span has on those that surround it. If the tension is recalculated, each of the other values is also affected. For instance, if:

$T = tension$
$v = velocity$
$s = stretch$
$k = constant$

Then,

$$\frac{dT}{dt} = k_1 v \qquad \text{and} \qquad \frac{ds}{dt} = k_2 v \qquad (2.6)$$

Because the simulation repeats the calculations for every time step, the first instantaneous jump disrupts the next time step's calculations. For example, the model begins at a uniform web tension throughout the web path. The first round of calculations uses the initial tension value. If the turnbar is between spans ten and eleven, the tension instantaneously goes through a drastic change at that point. All the tension calculations for spans twelve through forty use this new tension value. However, at the beginning of the next time step, there is a discontinuity between the tension in the twenty-fifth span (where there happens to be a dancer) and that in the first. This is important because the computer takes into account the effect of web spans on each other, especially the spans within the dancer. The dancer regulates tension, so the dancer movement carried from time step one is dramatically different from that which is necessary based on the tension in span one at time step two, resulting in faulty tension spikes.

The Capstan formula where T_{high} is downstream of T_{low} is:

$$e^{\mu\theta} = \frac{T_{high}}{T_{low}} \tag{2.7}$$

μ=coefficient of friction
Θ=angle of wrap

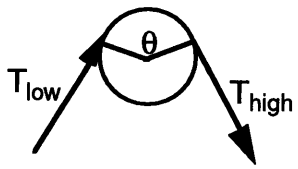

Figure 4: Cross-sectional schematic of web flowing over turnbar.

Through research and numerous trials, code and calculation order proved critical to the outcome of the simulation. Small rearrangements in the order of calculation prevented the simulation from carrying over unintended values. Recalculating tension, stretch, and velocity for the spans around the turnbar in the proper order alleviated most of the problems, resulting in outputs similar to the physical data seen on line.

2.2.3 Dancers

Dancers are collections of idlers which are the primary method of controls in the web handling system. The dancer is connected to a pressure/force source that allows it to dictate tension. As tension in the web increases or decreases, the dancer moves (Figure 2). The dancer pivots on an axis parallel to the axis of rotation of the idlers, so its position is generally determined by the angle from its neutral position. This position is calculated in the model based on the force balance theory. Included in the calculations are the torque exerted on the dancer by the web (including the tension in each span) and dancer inertia. These forces are then balanced by the internal force set point and damping (Equation 2.8). Dancer movement occurs if the forces are out of balance (Equation 2.9). Basically,

$$Torque = \sum T_i d_i \tag{2.8}$$

$$\omega_2 = \left(Torque - F_s - \omega_2 k_{friction}\right) \times DeltaTime \div Inertia + \omega_1 \tag{2.9}$$

$$\theta_2 = \omega_2 \times DeltaTime + \theta_1 \tag{2.10}$$

Where T = Tension in web span
d = Distance from axis of rotation in dancer to web span
i = Span number (only spans within dancer are summed)
ω = Angular velocity of dancer
θ = Angular position of dancer
DeltaTime = Integration time step

The dancer's position is calculated from the angular velocity of this movement (Equation 2.10) and fed into the position control loop which determines position and velocity error. The error is then sent to the speed control loop which speeds up or slows down the controlled motor. Figure 5 diagrams the standard control logic used.

For the specific web path in question, dancer A controls the unwind speed, while dancer B controls the driven s-wrap, as shown in Figure 6.

2.3 Start-Up, Shutdown, Splice

All points driven off the main drive shaft are programmed for a predetermined acceleration rate of two feet per second squared and deceleration rate of five feet per second squared to simulate start-up and

shutdown. These acceleration and deceleration rates are based on the current running systems. Although they are driven off the main drive shaft, each driven point can be adjusted for slightly different maximum velocities. This improves control of draw, thus tension, in the spans. Draw is the ratio of material pull between two spans.

2.3.1 Splices

Splices occur when a roll of material runs out. Because most web handling systems are continuous, a new roll is begun before the end of the old roll (without shutting the line down). The splice is basically the process by which the beginning of the new roll of material is tacked to the tail of the old roll, and the motor clutch switched to drive the new roll. Splices are a little more complicated to simulate than start-up and shutdown. The approach taken here was to reduce the tension to zero in the web span coming off the unwind for a brief period (four times delta time, or four times five thousandths of a second) just before the splice. DeltaTime is the integration time step used in the computer simulation. This period of time correlates with the amount of web left slack by the operators when setting up a new roll for the splice. Extra slack web increases web storage to allow for more unwind motor reaction time. The motor must accelerate the new roll to full speed once the clutch is shifted. During this splice window, the inertia of the unwind roll also shifts

from that of an empty roll to a full roll. The detailed calculations can be found in the computer code in Appendix B.

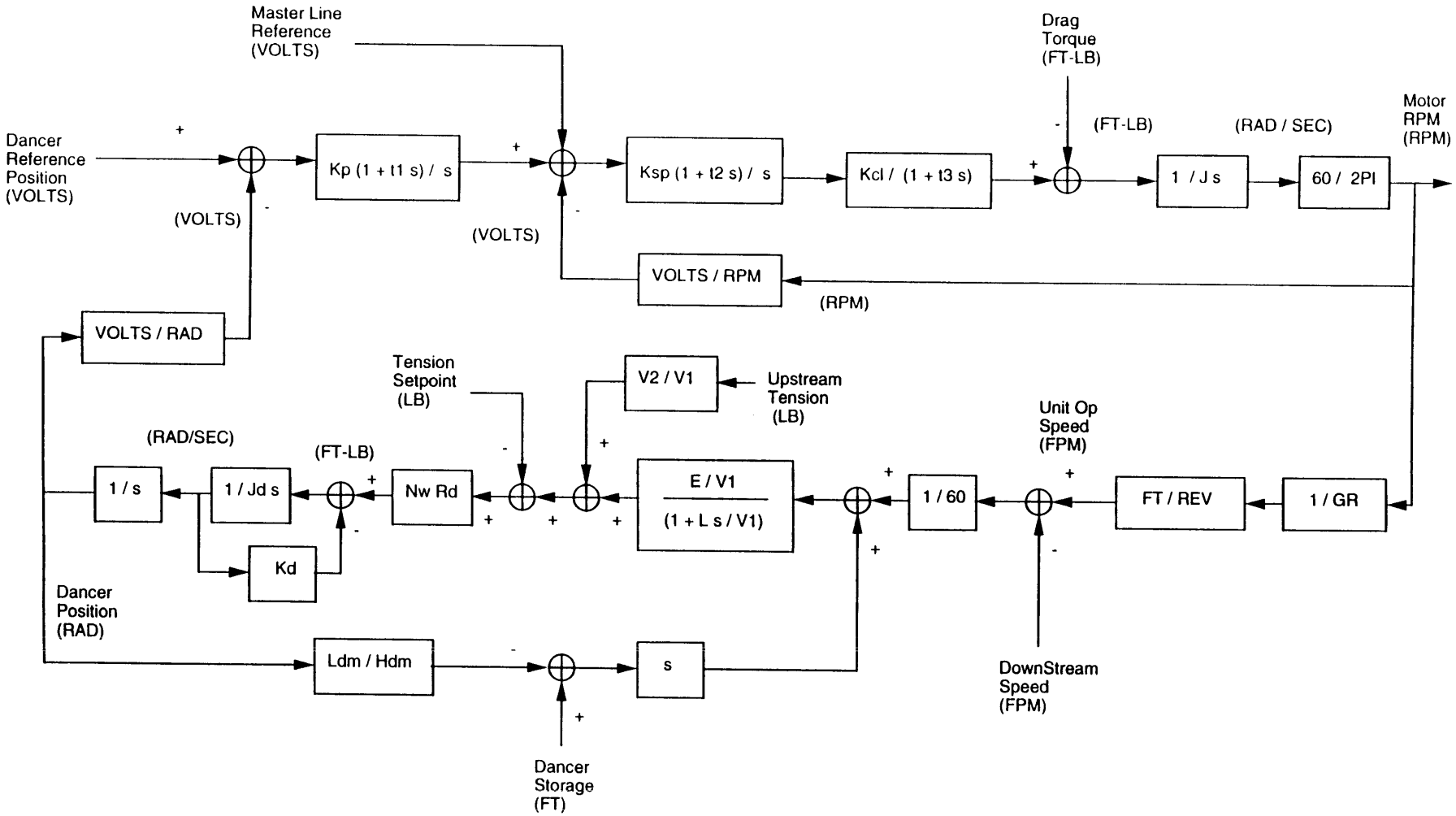

Figure 5: Block Diagram of Controller. Created by Dick Cornett.

WEB PATH

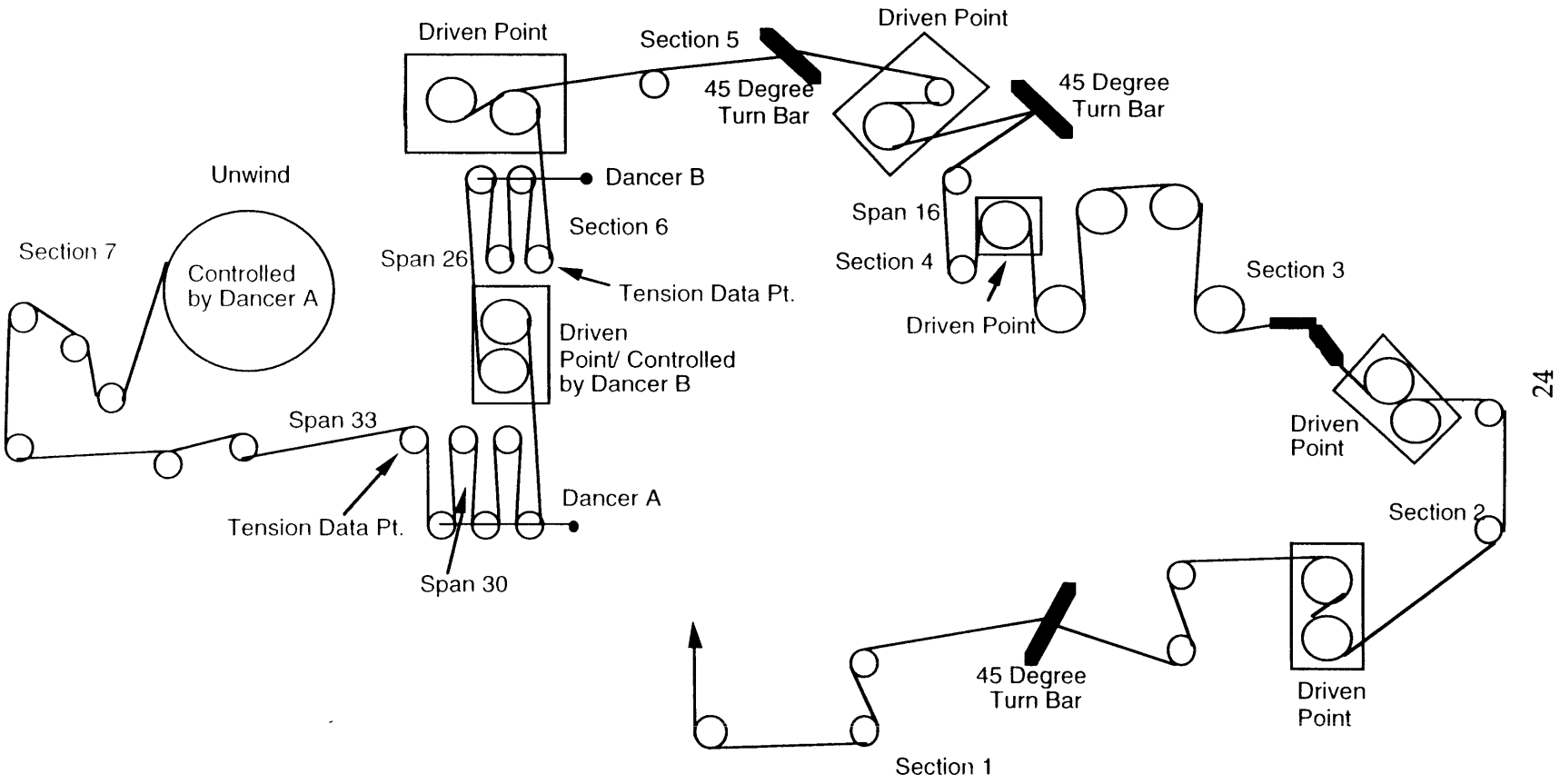

Figure 6: Material Web Path.

3 Model Validation

Once the model had been developed, data was collected off the actual line to validate it. Because of limited resources and access to line time, a few strategic points of data collection were selected. The tension sensor locations were chosen to better understand dynamic tension coming off the unwind where most of the problems tended to occur. (Figure 6 depicts the specific locations.) Linear variable differential transformers were used to measure the position (or change in position) of the idler to which it was attached. The sensor automatically translates the position into the load applied by the web and tracked by a voltage output. This output can be calibrated to pounds of tension. The dancer position signal was also taken to better understand the dancer's response to tension changes. Comparing the web speed and tension on line start-up revealed the effect of the acceleration as well as any time delays in the physical system. Monitoring the splice signal allowed for similar comparisons during splice. The splice signal reveals the period during which the actual splice is occurring with two electrical leads that display an output when touching (the exact moment when the two webs are being attached). This signal can be also compared to the tension and velocity signals to reveal time delays in the response of spans further downstream. Additionally, it helps to verify that the tension spikes and dynamics seen can truly be attributed to the splice.

Before the model could be validated, the data had to be assessed for probable accuracy. Surprising results often turned out to be caused by incorrect wiring or labeling. Once consistently reasonable data was collected, it could be compared to the model outputs. Refinements in modeling methods then led to a physically comparable model.

3.1 S-Wrap Controller

Initially, problems arose when a tension spike in the physical data did not appear in the model at tension test point 2. Details of the actual dancer and s-wrap controller system were investigated further in order to better model the control strategy. Originally, it was believed that the main s-wrap drive was referenced off the main line drive at one hundred percent (It was desired that the s-wrap speed be as close to equal that of the main drive as mechanically possible), and the controller made minor adjustments through a small motor beveled to the main gears. Thus, if the main drive is running at one hundred feet per minute, the s-wrap drive would be driven off the main drive shaft at the same one hundred feet per minute. The controller would then make adjustments on the order of a few feet per minute as determined by the feedback from the dancer. This controller functions as a velocity adder. According to the manufacturer, this controller system responded at a ramp of two seconds for a zero to one hundred percent speed change. If a velocity

increase of five feet per minute were necessary, the controller would accelerate at a constant rate and reach the five feet per minute velocity in two seconds. The difference between the data and the model led raised a suspicion that this controller actually responded much more sluggishly. By increasing the model's controller damping, thus slowing response in the controller, the model began to mimic the physical data results, verifying the physical systems delay in responding to significant changes such as start-up, shutdown, or the splice.

Further investigation and data collection showed that the s-wrap is driven at about ninety percent of the main line reference, and the controller is designed to continually add the additional ten percent, eliminating any possible backlash in the differential gears. By implementing the ninety percent reference into controller B calculations along with the increased damping, model outputs dynamically reflected trends in the data, validating the model.

Although the model and physical data did not identically match pound for pound, the trends seen in both were similar. This was the initial goal — identifying dynamic extremes which could potentially cause problems. Generally, the tension and dancer movement dynamics seen in the model outputs are slightly exaggerated compared to physical dynamic data. Thus, the model would be representing worst case scenarios, which is preferable for real-time practice. Much of the higher order mechanical damping in the

machinery, such as sticky bearings and variable friction are not included in the model. Often these damping factors are extremely difficult to define and inefficient to measure and control. Using the 'worst case' scenario of the model results in a built-in safety factor.

Figures 7 and 8 show model simulation results. Figure 7 plots the velocity of four different web spans, designated by their span number (see Figure 6), for a start-up and splice at twenty-five seconds. The velocity in the spans (33 and 40) between the splice point and dancers oscillates dramatically while the spans downstream of the dancers are much less affected by the splice. Figure 8 plots the tension and dancer positions for start-up and splice. Once again, overshoots and oscillations are seen for both dynamic states with the splice reactions being more drastic. The small line segments seen in both figures, mainly on start-up, are a side effect of numerical integration.

For comparison, Figures 9 and 10 depict data collect during on-line tests. The small oscillations seen throughout these figures are due to slightly elliptical material rolls which are driven at a constant angular velocity. The tension characteristics on start-up (Figure 9) are similar to those seen in the simulation of Figure 8. Both reveal a tension spike near span 23 and small oscillations near span 33. The tension spike most likely results from the slow acceleration and reaction time of the s-wrap controller. Roll 21, as labeled in Figure 6, is accelerating with the main drive as seen in the web speed plot

(Figure 9). However, the s-wrap controlled by dancer B is slow to react and does not accelerate as quickly. The velocity difference then increases between the controlled s-wrap and roll 21. Thus, the tension increases for those spans. Once the velocity reaches steady state and the controller catches up, the tension decreases. Meanwhile, oscillations are seen between dancer A and the unwind roll as the dancer and controlled motor driving the material unwind roll are unable to completely stabilize the system in such a short time interval. The dancer and the motor it controls attempt to maintain tension.

Figure 10 reveals the dynamics seen on-line during a splice. Here, the splice machine is splicing when the splice signal steps above zero. Likewise, the splice is complete once the signal falls back down to zero. This data can also be compared to the simulation seen in Figures 7 and 8. It is interesting to note that the tension in the web falls just before the splice and spikes with the splice. Loss in tension is caused by the web leaving the low inertia roll too quickly. When a new, large inertia roll is spliced in, the web must "pull" the material off the roll, increasing the tension in the spans near the roll. The dancer position also reflects changes in tension, falling just before and then rising just after the splice. This phenomenon is minimized in the model simulation of Figure 8 where the tension and dancer positions (especially the dancer closest to the material roll) fall just before the splice and spike immediately after the splice. The same fall and rise sequence is seen in the velocities of Figure 7 and 10.

Comparisons between the data collected from tests and model simulations verify the validity of the model. Although the exact quantitative tension and velocity are not identical, the key trends and characteristics are reflected in the model, allowing for use of the model as a worst case scenario. The model dynamics are more extreme than the responses seen in the physical system. The model is useful for predictions and large-scale equipment decisions.

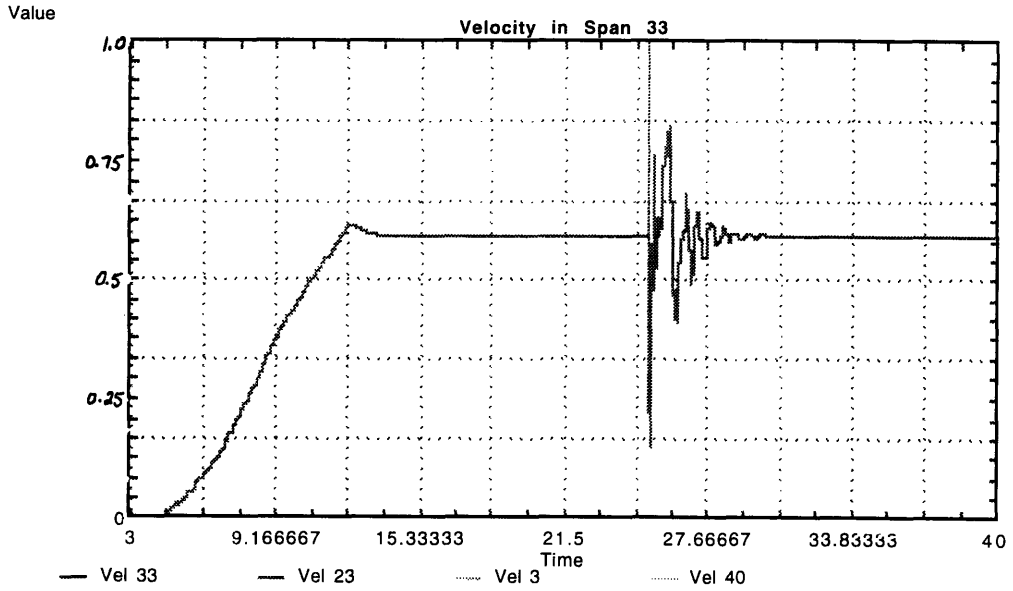

Figure 7a: Velocity in Span 33 (unitless - normalized).

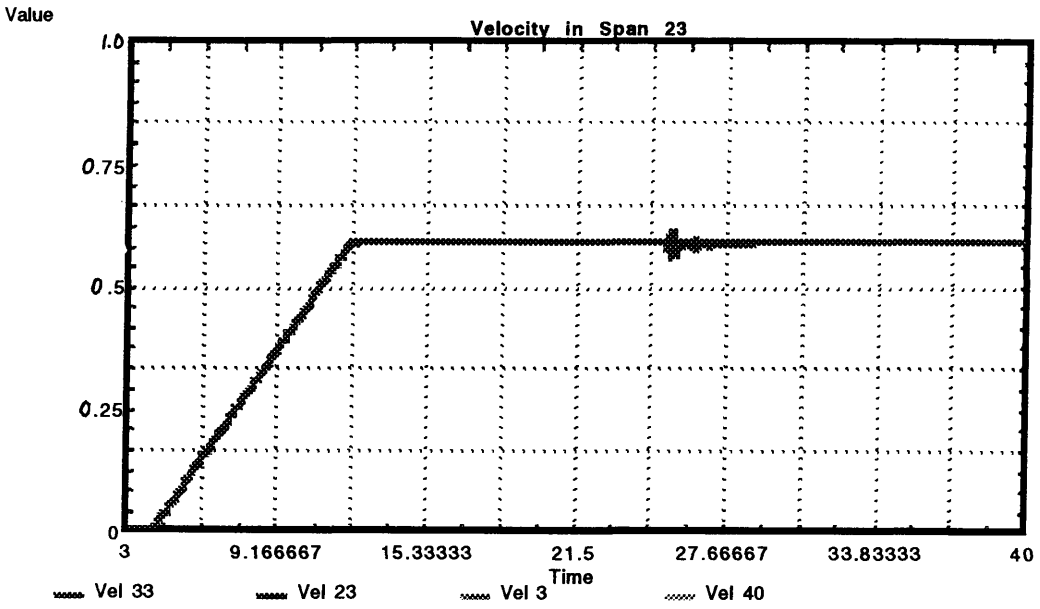

Figure 7b: Velocity in Span 23 (unitless - normalized).

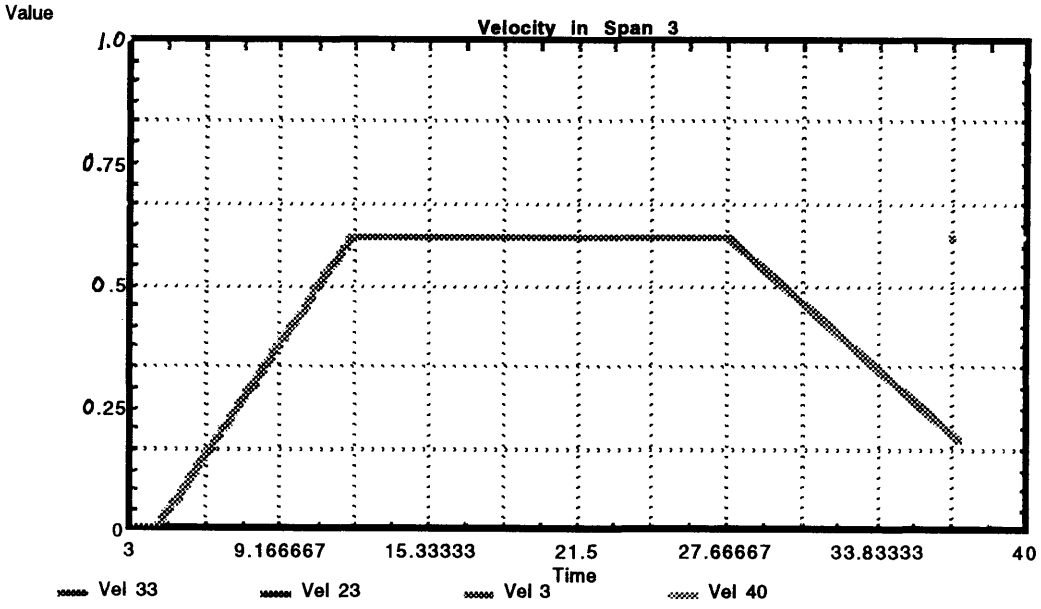

Figure 7c: Model Simulation Velocity for Span 3 (unitless).

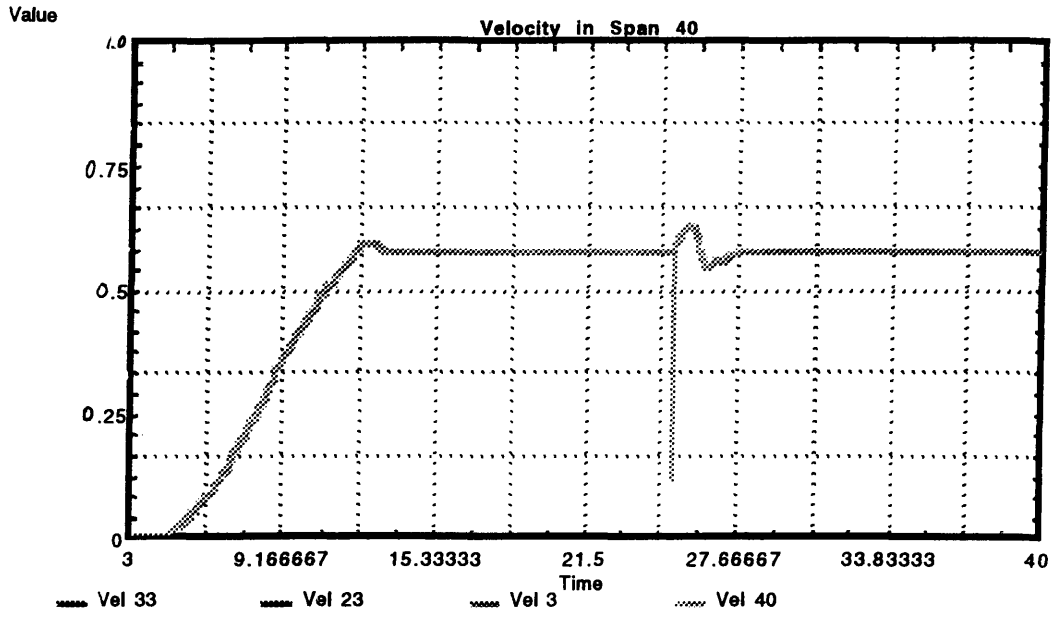

Figure 7d: Model Simulation Velocity for the material unwind roll (unitless).

Figure 7: Model Simulation Velocity Outputs on Start-Up and Splice.

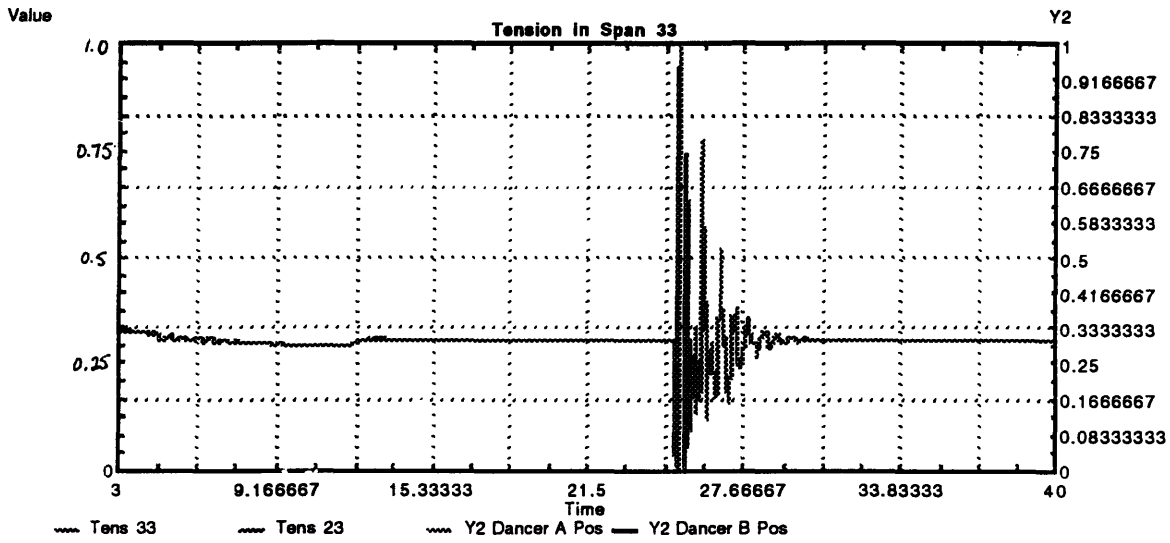

Figure 8a: Tension in Span 33 (unitless) on left axis.

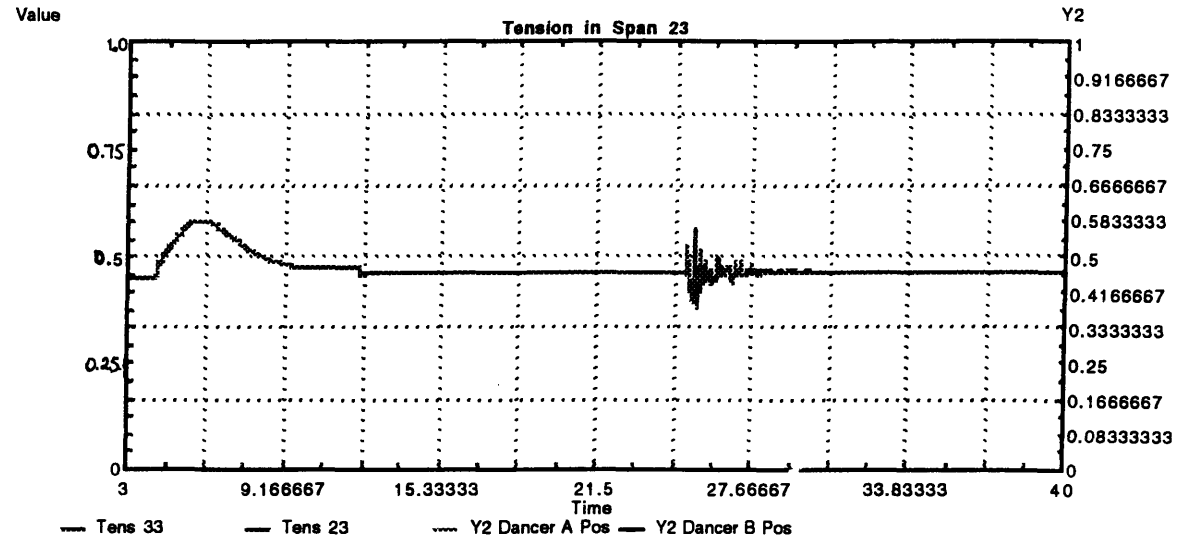

Figure 8b: Tension in Span 23 (unitless) on the left axis.

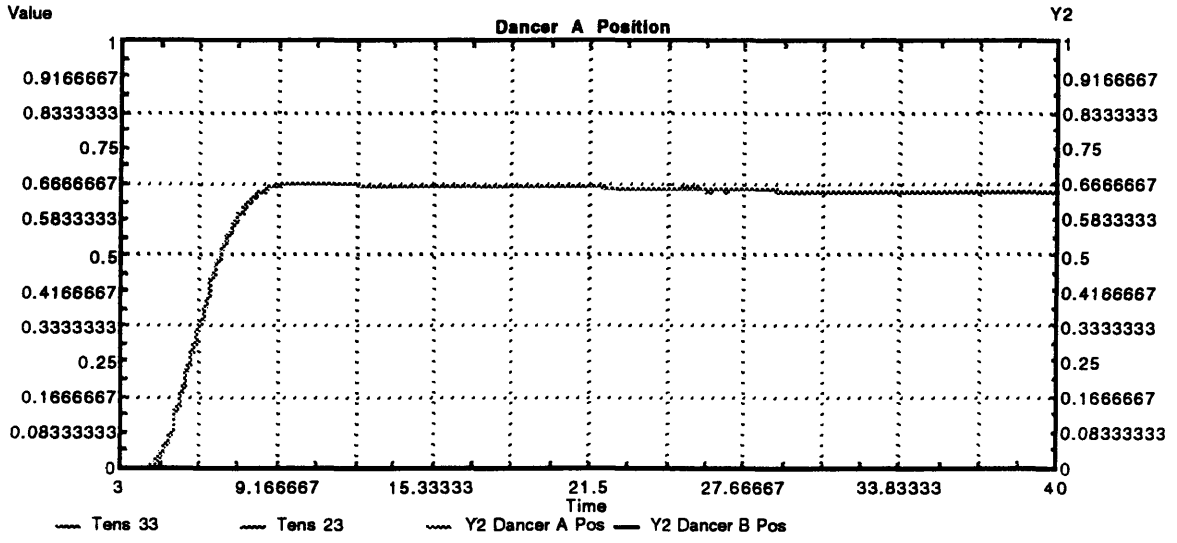

Figure 8c: Model Simulation for Dancer A s Position (maximum range of zero to one).

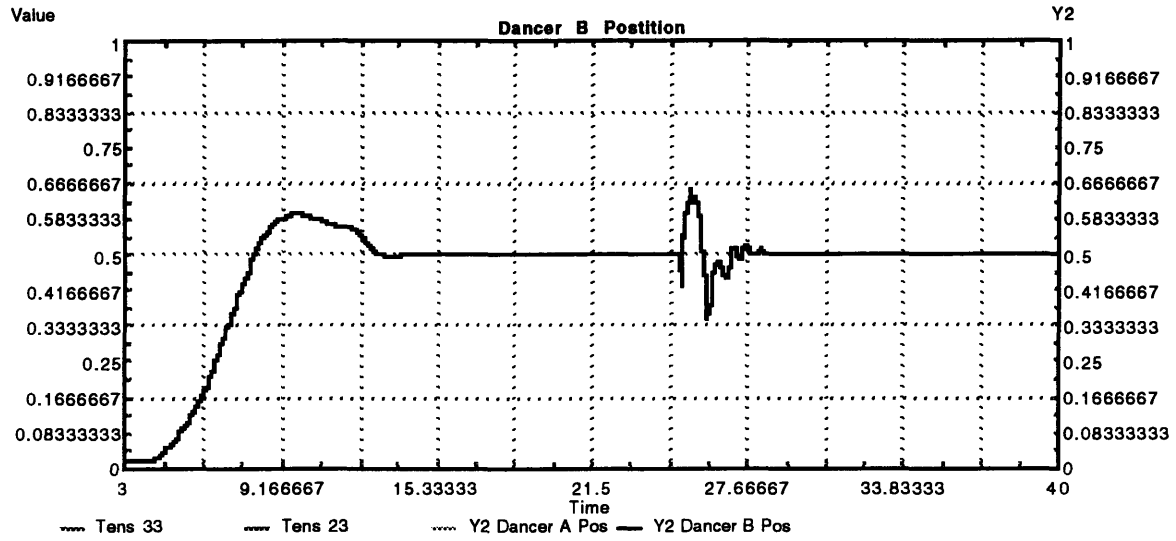

Figure 8d: Model Simulation for Dancer B s Position (maximum range of zero to one).

Figure 8: Simulation Tensions and Dancer Positions on Start-Up and Splice.

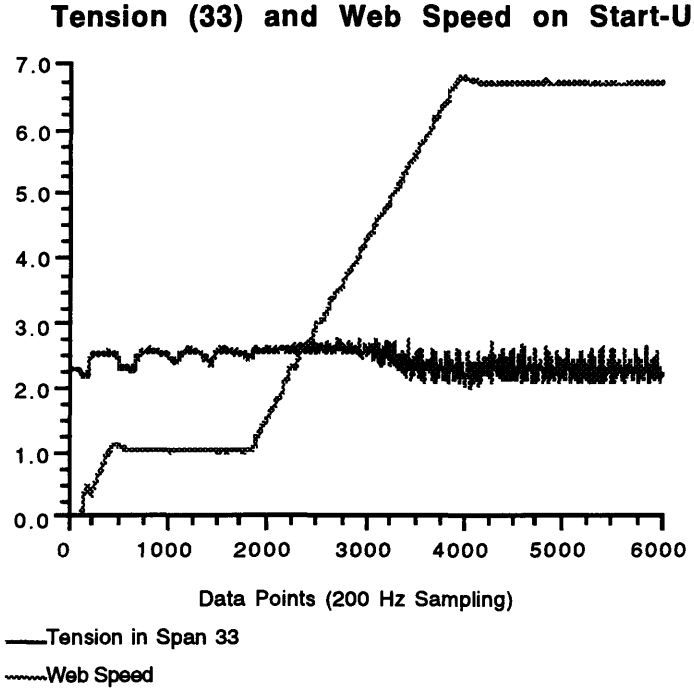

Figure 9a: The tension in span 33 (see Figure 5 for the location of the span) and the web speed on start-up.

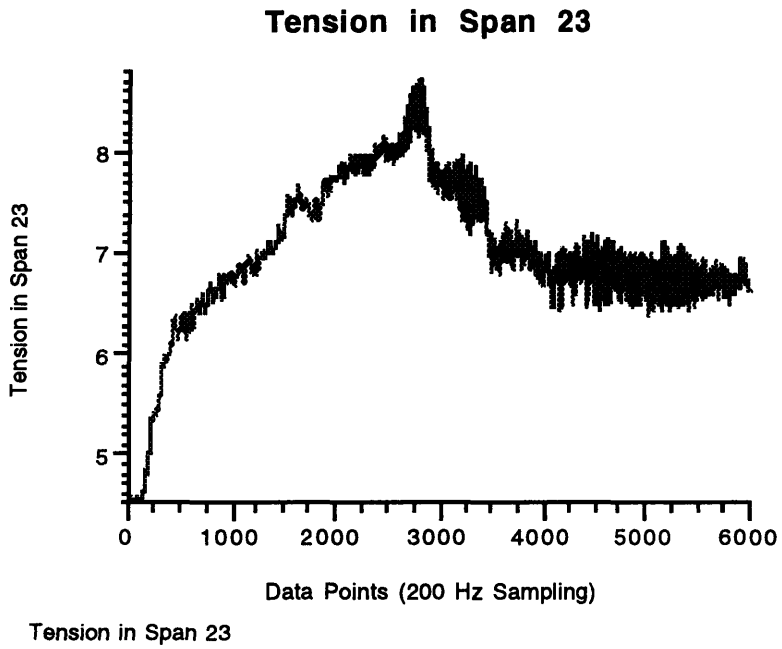

Figure 9b: Tension in span 23 on start-up.

Figure 9: Tension Data Collected from On-Line Testing.

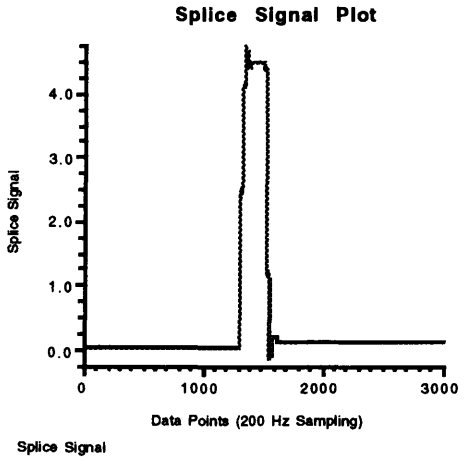

Figure 10a: Splice Signal.

Non-zero during splice.

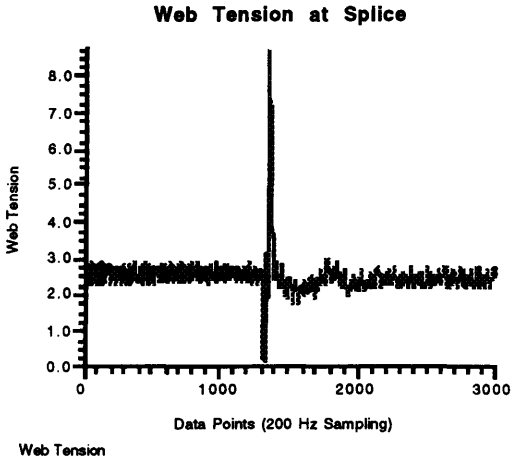

Figure 10b: Web Tension During Splice.

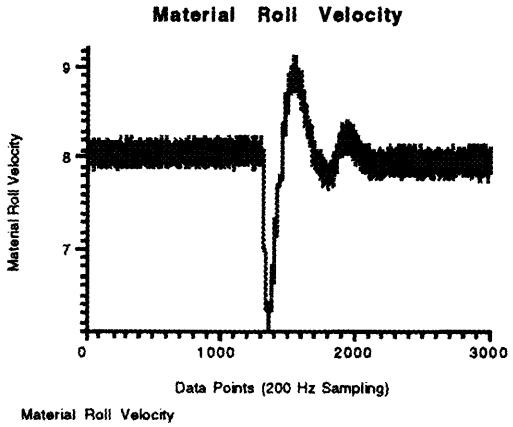

Figure 10c:Velocity taken at material roll.

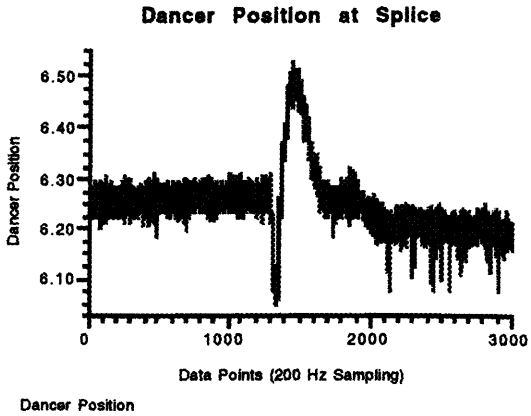

Figure 10d: Dancer position at splice.

Figure 10: Splice Data Collected On-Line

4 Possible Areas of Research and Improvements

Based on scenarios programmed in the model, possible problems and improvements were identified for future study. Unlike on-line testing, the model allows for quick equipment changes. The effect of the changes reveal worst case outcomes and allow for large-scale decisions. On-line controllers can also be easily altered without affecting other aspects of the converting line. Rockwell controllers are the most commonly used. Detailed information on the logic and recommended gain settings can be obtained from Rockwell for its various programmable logic controllers. They allow operators and engineers to easily change their gain settings. Since the model reflects web handling physics and controller logics, it may assist in pinpointing possible trouble spots or particular sensitivities in the system.

4.1 Controller Changes

Changes to the controller are by far easier and less costly than machinery adjustments. The model confirmed the controller's sensitivity to tuning. Small changes to the proportional and integral gain greatly affected the outcome of the model simulation. The controllers are the standard proportional-integral-derivative (PID) controllers used in the standard feedback loop scenario. The dancer position supplies the feedback signal while the main line drive reference supplies the feed-forward signal. The

calculations for simulation are detailed in Appendix B. When mechanical changes are made to the system, the gains should be carefully reset for optimal response.

A relatively simple change would be to smooth out the acceleration and deceleration ramps. Instead of a sharp corner when maximum velocity is reached, an exponential curve would decrease dynamic responses, both overshoots and oscillations. This theory was simulated in the model by inputting a s-curve instead of the linear acceleration ramp, reducing the resulting overshoot and reaching steady state in a shorter time interval. Comparing Figures 7d and 11 show that the s-curve eliminates the overshoot seen in the start-up with a linear acceleration ramp.

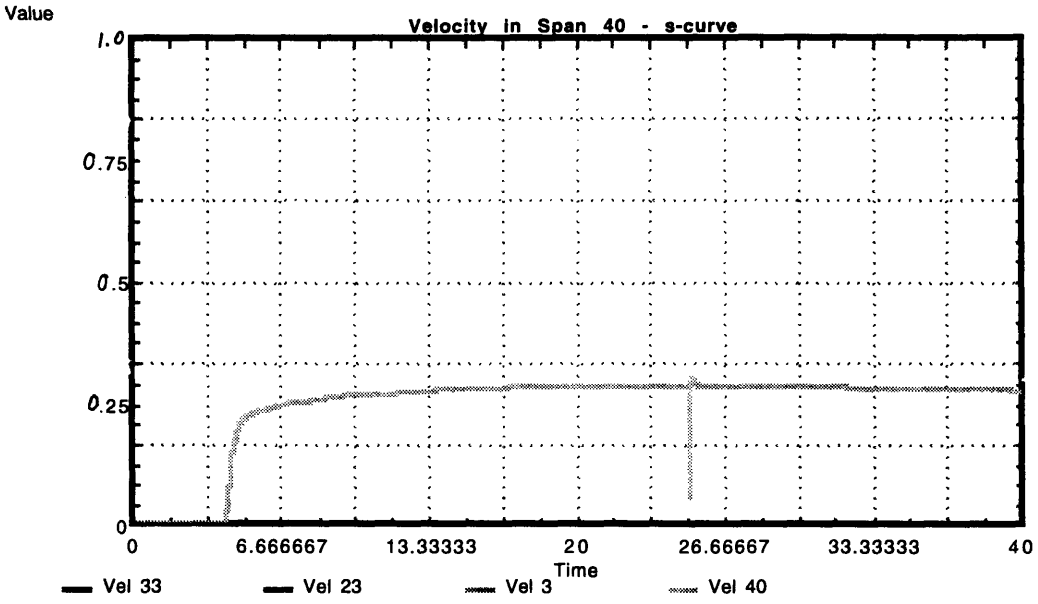

Figure 11: Model simulation of the velocity of the material unwind roll with and s-curve acceleration ramp.

Another area of interest for further investigation is the controller on the s-wrap (the velocity adder). Based on model trials, slightly faster controller response times — or less damping in the model — would significantly reduce tension spikes in start-up. This modification to the model resulted in the elimination of many of the larger tension tension spikes. The damping in the controller for the simulation shown in Figure 12 is half that of the damping of the simulation in Figure 8b. By comparing the two figures, it is easy to see that the faster controller significantly reduced the tension spike seen during start-up. Both figures simulate the tension in span 23.

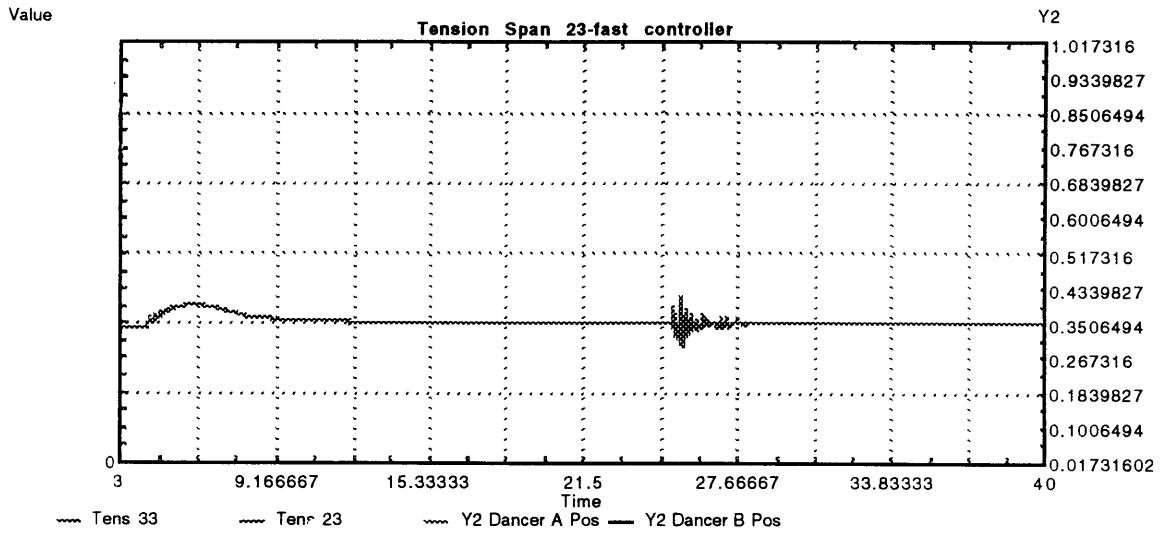

Figure 12: Tension in span 23 with half the damping of the simulation in Figure 8b. The tension spike on start-up is significantly less.

4.2 Mechanical Changes

Because tension and dancer movement dynamics were minimal in both the model and collected data downstream of dancer B, the necessity of dancer A was questioned. Dancers are designed to reduce the dynamics in the web handling system and to help to maintain the constant tension necessary for the manufacturing process. Sometimes, the dynamics of the system are so severe that more than one dancer is necessary. However, dancers should be minimized since they greatly increase the complexity of the line. The entire dancer A and controller system was removed from the model with only minor effects on the output. Thus, future iterations should consider simplifying the web path by eliminating the second dancer and controller.

Currently, a splice procedure consists of the transitory attachment of the leading edge of the new web to the tail of the old web. At the same time, the clutch switches to the new roll to accelerate it to full speed. If it were mechanically possible, a moving splice, where acceleration of the full roll occurs before the end of the old roll would be ideal. Tension spikes would be reduced during splice by removing the instantaneous need to accelerate a high inertia roll to full speed. This is analogous to a track relay where the next runner starts running before the previous runner stops. The baton exchange occurs only when both are in motion.

5 Other Applications of Dynamic Web Modeling

An Extend model developed for one web path can quickly be adapted for other web paths. Two examples follow of simple web path models built in a few hours that produced results that assisted in equipment decision making.

5.1 Example 1

A new material and a new roll size were to be incorporated into an existing web handling system. It was unknown whether the current brake would be strong enough to accommodate the larger, heavier roll. A model was built to help determine ranges for physical testing in the evaluation of new equipment needs. The model consisted of the unwind (material) roll, idlers, a pulling force (s-wrap), and a damping force which simulated the brake mechanism. The varying combinations of roll diameter and unwind roll damping force presented a testing limitation. The team only had one day of on-line testing time to make their decision whether to purchase new equipment. Using the model, combinations could quickly be tested for extremes on all other limits. Thus, a finite range of damping could be chosen for on-line testing for specific roll diameters.

It was believed that a new brake might be necessary to compensate for the higher inertia roll. Various cases with the new roll were simulated to test this theory. It was discovered that the increased moment arm due to increased roll diameter would compensate for the larger inertia. Thus, the force from the existing brake was sufficient to slow the larger inertia because the force was applied at the larger radius (effectively increasing the moment arm). With model results and comparable physical data, it was decided that the new equipment would not be necessary. The circumstances around this example are proprietary and thus cannot be detailed.

5.2 Example 2

Another material's path, whose model was adapted off the previously developed model in a couple hours, is depicted in Figure 14. The actual physical parameters such as web span lengths, material roll inertia, and controller gains were quickly and easily replaced in the model. Data for this other material was collected in the same manner as the original web's dynamic data. A comparison of the results from the model and collected data were almost identical. Both show a slight tension spike at initial start-up with a dip and plateauing as the web speed plateaus. The tension rises slightly with shutdown. Figure 14 shows the model results while Figures 15 and 16 graph physical data. A tension spike and oscillations are seen at the moment acceleration begins, revealing sluggishness in the controller

response and problems with an instantaneous acceleration. Small oscillations are again seen throughout the plot of collected data. This is also due to slightly elliptical rolls moving at a constant angular velocity, resulting in noise in the system. The frequency of oscillation of the noise corresponds to half of a revolution (maximum radius is reached twice per revolution for an ellipse. It is important to remember that the tension sensors used were not highly accurate and possessed considerable internal noise which added to the decrease in accuracy.

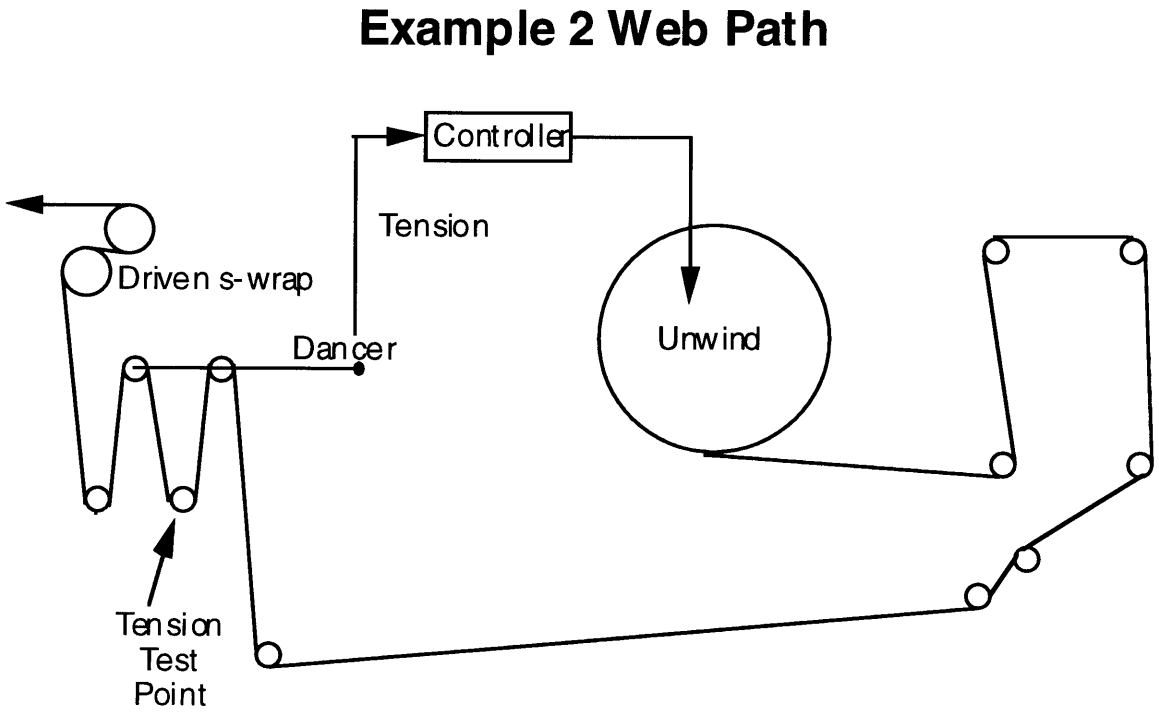

Figure 13: Example 2 Web Path Diagram

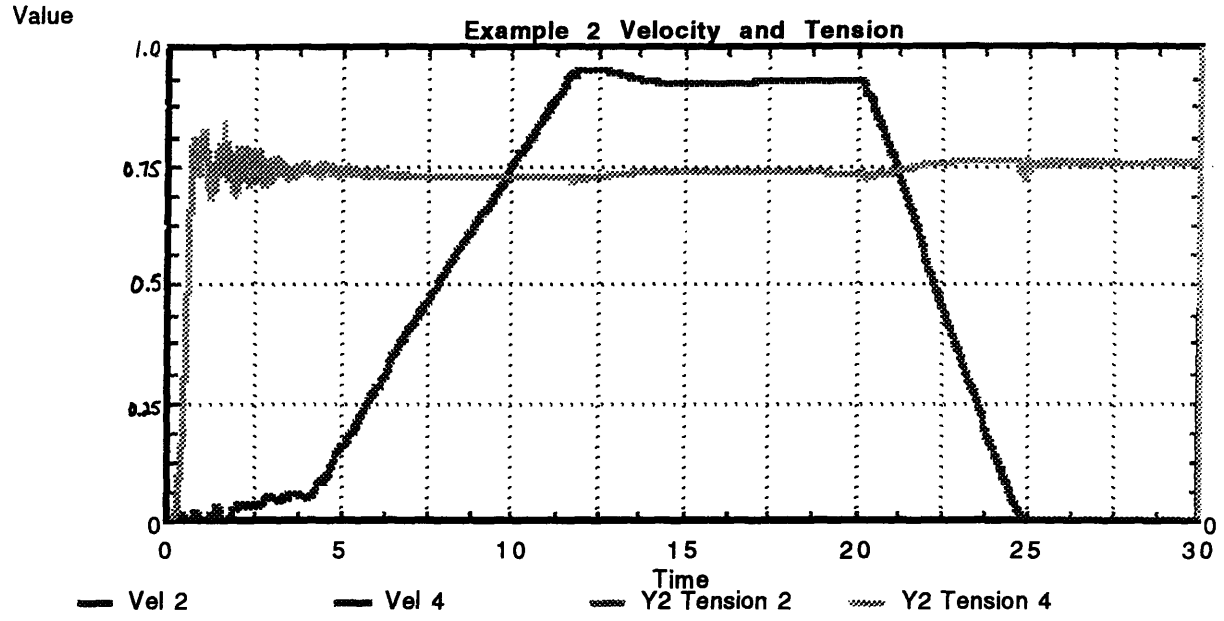

Figure 14: Model Simulation Velocity and Tension Outputs for Example 2. Plots correspond to simulations at the data point shown in Figure 10. Velocity ramps up and then down at 20 seconds. Tension spikes and oscillates, mainly on start-up due to dancer and system dynamics.

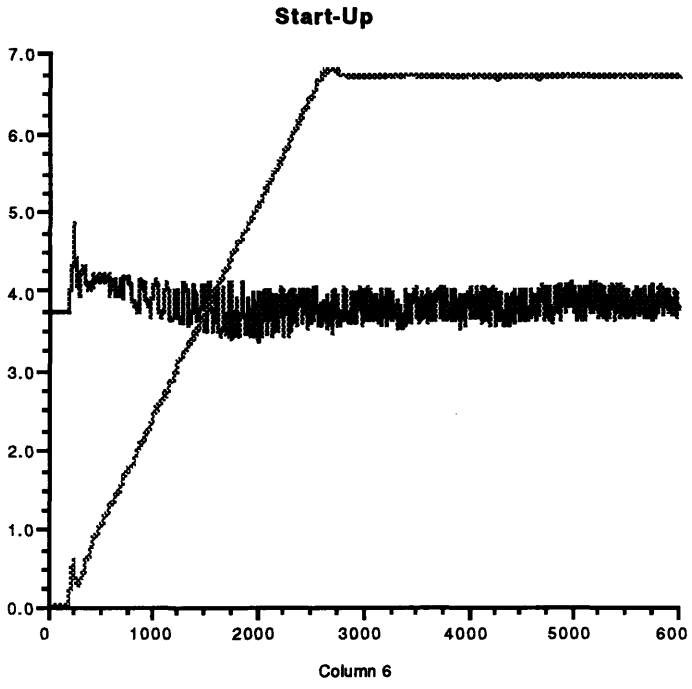

Figure 15: Example 2 Tension Data for Start-Up

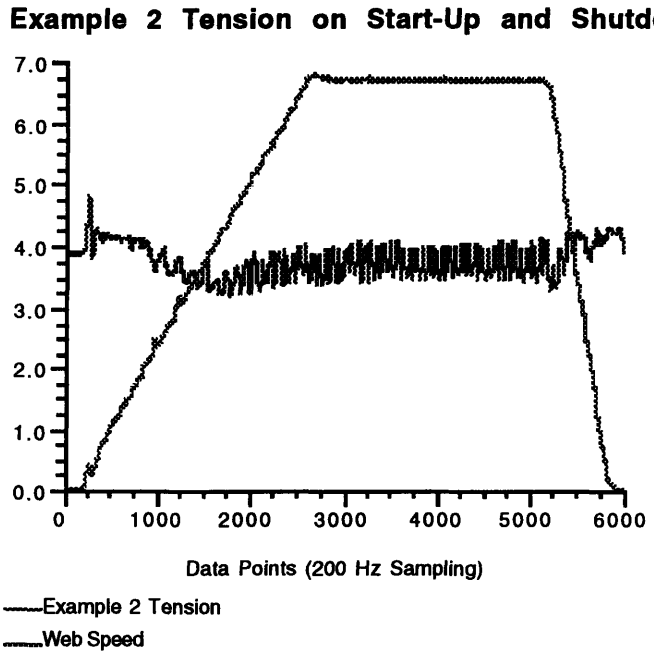

Figure 16: Data collect for Example 2 from on-line tests.

These results further confirm the reliability and potential benefits of dynamic modeling.

6 Conclusion and Recommendations

Computer dynamic modeling is a valuable resource for improving line reliability at low cost. Though the current models reliably simulate physical web handling systems, continued work and improvement of the web models will result in faster and more accurate simulations. Before achieving this accuracy through model refinements, increased sensor work is needed in order to better understand the physical system. The current tension sensors are delicate and easily damaged under manufacturing conditions and use. They are also fairly inaccurate and difficult (if not impossible) to accurately calibrate. Thus, they were most useful in understanding trends and spikes but not to take specific tension readings. Research into better instrumentation and controller measurements will promote a more accurate understanding of the actual line. Simpler user interfaces will promote widespread use of dynamic models, improving long-term reliability.

Based on comparisons between model outputs and data collected, computer dynamic modeling is a reasonable reflection of the physical system. With the development of basic models for each element in the web path, new web paths can be quickly and reliably constructed. Dynamic models can potentially speed up design and improvements, reducing overall costs.

Physical Parameters and Inertia Calculations for Web Path Elements

32.17400

	Idler 1	Idler 2	Idler 3	Idler 4	s-wrap	Dancer 1	Dancer 2
Inner radius (Ri)(ft)	0.08100	0.03988	0.03988	0.07292			
Outer radius (Ro)(ft)	0.08333	0.05271	0.05271	0.08208			
Length (ft)	1.25000	1.25000	1.25000	1.15000			
Density (lb/ft^3)	170.03520	170.03520	170.03520	72.57600	489.02400	170.03520	
x-section width (ft)							
x-section height(ft)			More NE				
Weight (lb)			0.00021				
Nylon ends in-rad	0.06375	0.01000	0.02483	0.05313			
Nylon ends out-rad	0.07625	0.04621	0.03333	0.07833			
NE height	0.12500	0.01000	0.05646	0.08333			
NE density	72.57600	72.57600	72.57600	72.57600			
Bearing weight (lb)	0.04000	0.04000	0.04000	0.09000			
Bearing in-rad	0.04417	0.04167	0.04167	0.05750			
Bearing out-rad	0.06902	0.04592	0.04592	0.02600			
Inertia (J)	0.00240	0.00184	0.00185	0.00305		2.52136	4.76196

Assume use of Fafner 9302K bearings

J=Mass(Ri^2+Ro^2)/2g +Bearing mass(Rbi^2+Rbo^2)/2g+NE mass(Rni^2+Rno^2)/2g

Appendix A: Inertia Calculations

Appendix B: Web Path Code

Written in conjunction with Bub Stuebe and Mark Southman.

```
// Declare constants and static variables here.

//*************************************************************
//
//*****************  SIMULATION************************
//
//*************************************************************


//*****************VARIABLE
DECLARATIONS************************

INTEGER   i,                Idler,          Mstart,
          M,                K,              L;

REAL      Ramp_time,        Ar1,            V1max,
          Ar7,              Ard1,           Ard7,
          Ard10,            Ard15,
          Ard19,            Ard22,          Ard27,
          VelRefold1,       V[41],          F[41],
          Lu[41],           N[41],
          Rs[41],     X[41],          Z[41],          g,

          Dpi,              Dpo,            Jpc,            Ar10,
          Jpr,              Fpts1,          Tptdm,
          Fpbs,             Tpbdm,          Tpd,            Dpr,
          Hptr,
          Nptd1,            Sheet_tork_pt1, Sheet_tork_pt2,
          Hptd1,
```

Hptd2, Nptd2, Fpts2,
Hd_perr2, Kptdmp, Hd_nlim,
Hd_plim, Jd1,
Jd2, Rdi, Rdi2, Rdi3,
Rdi4, Jdi, Jdi2, Jdi3,

Jdi4, Filcd, Cdivd,
Tdicd,
Cdivd2, Tdicd2, Cdivd3,
Tdicd3, Filvd, Cdivd4,
Tdicd4, Ri,
Ji, Civd, Ticd,
Rs_in,
Rs_out, Str, Hpbr, Tptd,
Tpbd, Texd, RsTest,
Hd_rpos,
Hd_perr1, Kff1, Kp1, Kd1,
Ki1, ViRef1, VelRef1, Ar15,
VelRef2, ViRef2, Kff2, Kp2,
Kd2, Rd1, Rd2, Ksi2,
Ksp2, VelErr2, Ki2,
TorkOut2,
VelErr1, Ksp1, Ksi1, Tp1,
Ti1, Tt1, Tt2, Tcl1,
Tcl2, Ti2, Tc2, Tp2,
TorkOut1, Tc1, T1, Tload,
BeltForce, K1, BDPT, MPT,
Dbelt, TU2, T2, Ar19,
Ar22, V7max, V10max,
V15max, V19max, V22max,
V27max, Js,
Ds, Ar27, COF, Theta,
Fo[41], Stretch, Str17, Str18,
Str20, Str4, MaxDelV, DelV,
StartRamp, SpliceTime, Kptdmp2;

```
//******************VARIABLE
INTIALIZATION************************
on initsim
{
        X[1]=0.667;
        X[2]=0.083;
        X[3]=0.50;
        X[4]=0.222;
        X[5]=0.027;
        X[6]=0.333;
        X[7]=0.333;
        X[8]=0.222;
        X[9]=0.278;
        X[10]=0.333;
        X[11]=0.250;
        X[12]=0.250;
        X[13]=0.250;
        X[14]=0.083;
        X[15]=0.083;
        X[16]=0.417;
        X[17]=0.278;
        X[18]=1.000;
        X[19]=0.583;
        X[20]=0.667;
        X[21]=0.278;
        X[22]=0.361;
        X[23]=0.222;
        X[24]=0.236;
        X[25]=0.236;
        X[26]=0.375;
        X[27]=0.389;
        X[28]=0.236;
        X[29]=0.236;
        X[30]=0.250;
        X[31]=0.250;
        X[32]=0.306;
```

```
        X[33]=0.556;
        X[34]=0.056;
        X[35]=0.139;
        X[36]=0.333;
        X[37]=0.167;
        X[38]=0.236;
        X[39]=0.250;
        X[40]=0.000;


        FOR i=1 to 40
        {
                V[i]=0.0;
                F[i]=4.0;
                N[i]=0.0;
                Rs[i]=1;
                Z[i]=X[i];
                Lu[i]=Z[i]/((F[i]*454.0/10.0/Modulus)+1);          //Nonzero
initial tension
        }

//MISC PARAMETERS
g=32.174;
StartRamp=4.0;
SpliceTime=25.0;

//Dancer 1 Parameters
        Fpts1=4.5;
        Nptd1=0.0;
        Sheet_tork_pt1=0.0;
        Hptd1=PI/6.0;
        Kptdmp=20.5;
        Hd_nlim=0.0;
        Hd_plim=PI/3.0;
        Jd1=2.5/g;
        Rd1=14.0/12.0;
```

```
        Hd_rpos=PI/6.0;

//Dancer 2 Parameters
        Fpts2=4.45;
        Nptd2=0.0;
        Sheet_tork_pt2=0.0;
        Kptdmp2=3.5;
        Hptd2=PI/6.0;
        Jd2=4.8/g;
        Rd2=16.0/12.0;
        Hd_rpos=PI/6.0;

//Idler 1 Parameters
        Rdi=.081;
        Jdi=0.0024/g;
        Filcd=0.05;
        Filvd=0.05/1000.0*60.0;
        Cdivd=Filvd*Rdi*Rdi;
        Tdicd=Filcd*Rdi;

//Idler 2 Parameters
        Rdi2=.03988;
        Jdi2=0.00184/g;
        Cdivd2=Filvd*Rdi2*Rdi2;
        Tdicd2=Filcd*Rdi2;

//Idler 3 Parameters
        Rdi3=.03899;
        Jdi3=0.00185/g;
        Cdivd3=Filvd*Rdi3*Rdi3;
        Tdicd3=Filcd*Rdi3;

//Idler 4 Parameters
        Rdi4=.07292;
        Jdi4=0.00305/g;
        Cdivd4=Filvd*Rdi4*Rdi4;
```

```
        Tdicd4=Filcd*Rdi4;

//Speed Controller 1 Parameters
        Ksi1=0.05;
        Ksp1=3.5;
        Tc1=20.0;
        Tcl1=Tc1/1000.0;
        Ti1=0.0;
        Tt1=0.0;
        TorkOut1=0.0;
        VelErr1=0.0;
        ViRef1=0.0;
        VelRef1=0.0;


//Speed Controller 2 Parameters
        Ksi2=0.06;
        Ksp2=4.0;
        Tc2=20.0;
        Tcl2=Tc2/1000.0;
        Ti2=0.0;
        Tt2=0.0;
        TorkOut2=0.0;
        VelErr2=0.0;
        ViRef2=0.0;
        VelRef2=0.0;

//Roll/Unwind Parameters
        Dpi=7.0/12.0;
        Dpo=34.0/12.0;
        Jpc=0.0/g;
        Jpr=74.5/g;
        Tpd=0.1;
        Dpr=Dpo;
        BDPT=1.0;
        MPT=1.0;
```

```
        DBelt=3.0;
        K1=BDPT/MPT*Dpr/Dbelt;
        Tload=0.5;
        }
// SIMULATION

on simulate
{

//Dancer 1 Position Controller Parameters
If (CurrentTime<StartRamp)
        {
        Kff1=0.0;
        Kff2=0.0;
        Kp1=0.0;
        Kp2=0.0;
        Ki1=0.0;
        Ki2=0.0;
        }
Else
        {
//Dancer 1 Position Controller Parameters
        Kff1=1.0;
        Kp1=Kp1dialog;
        Kd1=0.0;
        Ki1=0.05;

//Dancer 2 Position Controller Parameters
        Kff2=1.0;
        Kp2=Kp2dialog;
        Kd2=0.0;
        Ki2=0.05;
        }

//Ramp Up Calculations
        Ramp_time=8.0;
```

```
Ar1=1/Ramp_time;
Ard1=1/3.0;
Ard7=(0.9937)/3.0;
Ard10=(0.9937)/3.0;
Ard15=(0.9937)/3.0;
Ard19=(0.9937)/3.0;
Ard22=(0.9824)/3.0;
Ard27=(0.9875)/3.0;


If (CurrentTime>StartRamp)
//{
        //V1max=0.0556;
        //V7max=0.0544;
        //V10max=0.0533;
        //V15max=0.0522;
        //V19max=0.0511;
        //V22max=0.0500;
        //V27max=0.0489;
//}
//ELSE
{

        Ar1=(1.00)/Ramp_time;
        Ar7=(0.9937)/Ramp_time;
        Ar10=(0.9937)/Ramp_time;
        Ar15=(0.9937)/Ramp_time;
        Ar19=(0.9937)/Ramp_time;
        Ar22=(0.9833)/Ramp_time;
        Ar27=(0.9875/60.0)/Ramp_time;
        V1max=1.00;
        V7max=0.9937;
        V10max=0.9937;
        V15max=0.9937;
        V19max=0.9937;
        V22max=0.9823;
```

```
        V27max=0.9875;
        //}
//V[1]=V1max/exp(1.6/CurrentTime);
V[1]=Ar1*DeltaTime+V[1];
V[1]=Min2(V[1],V1max);
//V[7]=V7max/exp(1.6/CurrentTime);
V[7]=Ar7*DeltaTime+V[7];
V[7]=Min2(V[7],V7max);
//V[10]=V10max/exp(1.6/CurrentTime);
V[10]=Ar10*DeltaTime+V[10];
V[10]=Min2(V[10],V10max);
//V[15]=V15max/exp(1.6/CurrentTime);
V[15]=Ar15*DeltaTime+V[15];
V[15]=Min2(V[15],V15max);
//V[19]=V19max/exp(1.6/CurrentTime);
V[19]=Ar19*DeltaTime+V[19];
V[19]=Min2(V[19],V19max);
//V[22]=V22max/exp(1.6/CurrentTime);
V[22]=(Ar22*DeltaTime)+V[22];
V[22]=Min2(V[22],V22max);
//V[27]=V27max/exp(1.6/CurrentTime);
//V[27]=Ar27*DeltaTime+V[27];
//V[27]=Min2(V[27],V27max);
}

//If (CurrentTime>20.0)
//{
//V[1]=V[1]-Ard1*DeltaTime;
//V[1]=Max2(V[1],0.0);
//V[7]=V[7]-Ard1*DeltaTime;
//V[7]=Max2(V[7],0.0);
//V[10]=V[10]-Ard1*DeltaTime;
//V[10]=Max2(V[10],0.0);
//V[15]=V[15]-Ard1*DeltaTime;
//V[15]=Max2(V[15],0.0);
//V[19]=V[19]-Ard1*DeltaTime;
```

```
        //V[19]=Max2(V[19],0.0);
        //V[27]=V[27]-Ard1*DeltaTime;
        //V[27]=Max2(V[27],0.0);
        //}
        If ((CurrentTime>(SpliceTime-8.0*DeltaTime)) and
(CurrentTime<(SpliceTime-2.0*DeltaTime)))
        {
        Jpr=1.0;
        Dpr=7.0/12.0;
        }
        Else
        //If (CurrentTime>SpliceTime)
        {
        Jpr=74.5/g;
        Dpr=34.0/12.0;
        }
//*****************************SECTION
1*********************************

//Span Calculations for Section 1
        Idler=2;
        Mstart=1;
        Ri=Rdi;
        Ji=Jdi;
        Civd=Cdivd;
        Ticd=Tdicd;
        FOR M=1 to 3
        {
                K=M+1;
                Str=(Z[M]-Lu[M])/Lu[M];
                Rs_in=Rs[K];
                Rs_out=Rs[M];
                Lu[M]=(V[K]*Rs_in-V[M]*Rs_out)*DeltaTime+Lu[M];
                F[M]=Modulus*Str*10.0/454.0;

                Rs[M]=Lu[M]/Z[M];
```

```
            IF (M==Mstart+Idler)
            {
            Goto Sect1a;
            }
            IF (N[K]<0.0)
            {
                  N[K]=((F[M]-F[K])*Ri-
N[K]*Civd+Ticd)*DeltaTime/Ji+N[K];
            }
            ELSE
            {
                  N[K]=((F[M]-F[K])*Ri-N[K]*Civd-
Ticd)*DeltaTime/Ji+N[K];
            }
            V[K]=N[K]*Ri;
      }
      Sect1a:
            F[4]=F[3]/exp(COF*Theta);
            Str4=F[4]*454.0/(Modulus*10.0);
            V[4]=V[7]*(1.0+Str4);     //reference velocity
            Rs[4]=1.0-Str4;

      Ri=Rdi;
      Ji=Jdi;
      Civd=Cdivd;
      Ticd=Tdicd;
      FOR M=5 to 6
      {
            K=M+1;
            Str=(Z[M]-Lu[M])/Lu[M];
            Rs_in=Rs[K];
            Rs_out=Rs[M];
            Lu[M]=(V[K]*Rs_in-V[M]*Rs_out)*DeltaTime+Lu[M];
            F[M]=Modulus*Str*10.0/454.0;
```

```
            Rs[M]=Lu[M]/Z[M];

            IF (M==6)
            {
            Goto Sect2;
            }
            IF (N[K]<0.0)
            {
                  N[K]=((F[M]-F[K])*Ri-
N[K]*Civd+Ticd)*DeltaTime/Ji+N[K];
            }
            ELSE
            {
                  N[K]=((F[M]-F[K])*Ri-N[K]*Civd-
Ticd)*DeltaTime/Ji+N[K];
            }
            V[K]=N[K]*Ri;
      }


//**************************SECTION
2*******************************
Sect2:
//Span Calculations for Section 2
      Idler=2;
      Mstart=7;
      Ri=Rdi;
      Ji=Jdi;
      Civd=Cdivd;
      Ticd=Tdicd;
      FOR M=7 to 9
      {
            K=M+1;
            Str=(Z[M]-Lu[M])/Lu[M];
            Rs_in=Rs[K];
            Rs_out=Rs[M];
```

```
            Lu[M]=(V[K]*Rs_in-V[M]*Rs_out)*DeltaTime+Lu[M];
            F[M]=Modulus*Str*10.0/454.0;

            Rs[M]=Lu[M]/Z[M];

            IF (M==Mstart+Idler)
            {
            Goto Sect3;
            }
            IF (N[K]<0.0)
            {
                  N[K]=((F[M]-F[K])*Ri-
N[K]*Civd+Ticd)*DeltaTime/Ji+N[K];
            }
            ELSE
            {
                  N[K]=((F[M]-F[K])*Ri-N[K]*Civd-
Ticd)*DeltaTime/Ji+N[K];
            }
            V[K]=N[K]*Ri;
      }

//*****************************SECTION
3*********************************
Sect3:
//Span Calculations for Section 3
      Idler=4;
      Mstart=10;
      Ri=Rdi;
      Ji=Jdi;
      Civd=Cdivd;
      Ticd=Tdicd;
      FOR M=10 to 14
      {
            K=M+1;
            Str=(Z[M]-Lu[M])/Lu[M];
```

```
            Rs_in=Rs[K];
            Rs_out=Rs[M];
            Lu[M]=(V[K]*Rs_in-V[M]*Rs_out)*DeltaTime+Lu[M];
            F[M]=Modulus*Str*10.0/454.0;

            Rs[M]=Lu[M]/Z[M];


            IF (M==Mstart+Idler)
            {
            Goto Sect4;
            }
            IF (N[K]<0.0)
            {
                  N[K]=((F[M]-F[K])*Ri-
N[K]*Civd+Ticd)*DeltaTime/Ji+N[K];
            }
            ELSE
            {
                  N[K]=((F[M]-F[K])*Ri-N[K]*Civd-
Ticd)*DeltaTime/Ji+N[K];
            }
            V[K]=N[K]*Ri;
      }

//**************************SECTION
4*******************************
Sect4:
//Span Calculations for Section 4
      Idler=2;
      Mstart=15;
      Ri=Rdi;
      Ji=Jdi;
      Civd=Cdivd;
      Ticd=Tdicd;
      COF=0.4;
```

```
        Theta=3.14;
        FOR M=15 to 17
        {
                K=M+1;
                Str=(Z[M]-Lu[M])/Lu[M];
                Rs_in=Rs[K];
                Rs_out=Rs[M];
                Lu[M]=(V[K]*Rs_in-V[M]*Rs_out)*DeltaTime+Lu[M];
                F[M]=Modulus*Str*10.0/454.0;

                Rs[M]=Lu[M]/Z[M];


                IF (M==Mstart+Idler)
                Goto Sect4a;

                IF (N[K]<0.0)
                {
                        N[K]=((F[M]-F[K])*Ri-
N[K]*Civd+Ticd)*DeltaTime/Ji+N[K];
                }
                ELSE
                {
                        N[K]=((F[M]-F[K])*Ri-N[K]*Civd-
Ticd)*DeltaTime/Ji+N[K];
                }
                V[K]=N[K]*Ri;

}

Sect4a:
                F[18]=F[17]/exp(COF*Theta);
                Str18=F[18]*454.0/(Modulus*10.0);
                V[18]=V[22]*(1.0+Str18);     //reference velocity
                Rs[18]=1.0-Str18;
                Goto Sect5;
```

```
//*****************************SECTION
5*******************************
Sect5:
//Span Calculations for Section 5
        Idler=2;
        Mstart=19;
        Ri=Rdi;
        Ji=Jdi;
        Civd=Cdivd;
        Ticd=Tdicd;
        //For M=19 to 21
        //{
        M=19;
                K=M+1;
                Str=(Z[M]-Lu[M])/Lu[M];
                Rs_in=Rs[K];
                Rs_out=Rs[M];
                Lu[M]=(V[K]*Rs_in-V[M]*Rs_out)*DeltaTime+Lu[M];
                F[M]=Modulus*Str*10.0/454.0;

                Rs[M]=Lu[M]/Z[M];


                //IF (M==Mstart+Idler)
                //{
                //Goto Sect6;
                //}
                //IF (N[K]<0.0)
                //{
                        //N[K]=((F[M]-F[K])*Ri-
N[K]*Civd+Ticd)*DeltaTime/Ji+N[K];
                //}
                //ELSE
```

```
            //{
                    //N[K]=((F[M]-F[K])*Ri-N[K]*Civd-
Ticd)*DeltaTime/Ji+N[K];
            //}
            //V[K]=N[K]*Ri;
        //}


            F[20]=F[19]/exp(COF*Theta);
            Str20=F[20]*454.0/(Modulus*10.0);
            V[20]=V[22]*(1.0+Str20);    //reference velocity
            Rs[20]=1.0-Str20;

            M=21;
            Str=(Z[M]-Lu[M])/Lu[M];
            Rs_in=Rs[K];
            Rs_out=Rs[M];
            Lu[M]=(V[K]*Rs_in-V[M]*Rs_out)*DeltaTime+Lu[M];
            F[M]=Modulus*Str*10.0/454.0;

            Rs[M]=Lu[M]/Z[M];


            IF (N[21]<0.0)
            {
                    N[21]=((F[20]-F[21])*Ri-
N[21]*Civd+Ticd)*DeltaTime/Ji+N[21];
            }
            ELSE
            {
                    N[21]=((F[20]-F[21])*Ri-N[21]*Civd-
Ticd)*DeltaTime/Ji+N[21];
            }
            V[21]=N[21]*Ri;
```

Sect6:

```
//Span Calculations for Section 6
        Idler=4;
        Mstart=22;
        Ri=Rdi2;
        Ji=Jdi2;
        Civd=Cdivd2;
        Ticd=Tdicd2;
        FOR M=22 to 26
        {
                K=M+1;
                Str=(Z[M]-Lu[M])/Lu[M];
                Rs_in=Rs[K];
                Rs_out=Rs[M];
                Lu[M]=(V[K]*Rs_in-V[M]*Rs_out)*DeltaTime+Lu[M];
                F[M]=Modulus*Str*10.0/454.0;

                Rs[M]=Lu[M]/Z[M];


                IF (M==Mstart+Idler)
                {
                Goto Dancer1;
                }
                IF (N[K]<0.0)
                {
                        N[K]=((F[M]-F[K])*Ri-
N[K]*Civd+Ticd)*DeltaTime/Ji+N[K];
                }
                ELSE
                {
                        N[K]=((F[M]-F[K])*Ri-N[K]*Civd-
Ticd)*DeltaTime/Ji+N[K];
                }
                V[K]=N[K]*Ri;
```

```
        }


Dancer1:
//Dancer 1 Calculations - Developed by Bub Stuebe

        Sheet_tork_pt1=F[23]*(11.0/12.0)+F[24]*(11.0/12.0)+F[25]*Rd1+F[26]*Rd
1;
        Hptd1=Nptd1*DeltaTime+Hptd1;
        Nptd1=((Sheet_tork_pt1-2.0*Fpts1*Rd1-2.0*Fpts1*11.0/12.0)-
Nptd1*Kptdmp)*DeltaTime/Jd1+Nptd1;
        IF (Hptd1>Hd_plim)
        {
                Hptd1=Hd_plim;

        }
        ELSE
        {
                IF (Hptd1<Hd_nlim)
                {
                        Hptd1=Hd_nlim;

                }
        }

        Z[23]=X[23]-Hptd1*11.0/12.0;
        Z[24]=X[24]-Hptd1*11.0/12.0;
        Z[25]=X[25]-Hptd1*Rd1;
        Z[26]=X[26]-Hptd1*Rd1;

        Hd_perr1=Hptd1-Hd_rpos;


//Dancer 1 Position Controller Calculations

        MaxDelV=7.5*DeltaTime*Specon/100.0;
```

```
        VelRefold1=Velref1;
        ViRef1=Hd_perr1*Ki1*DeltaTime*(V[22]/V22max)+ViRef1;  //scaled
velocity for ramp up
        VelRef1=0.9*V[22]*Kff1+Hd_perr1*Kp1*(V[22]/V22max)+ViRef1+Npt
d1*Kd1;  //scaled velocity for ramp up
        //90% Vel Ref in Specon
        DelV=VelRef1-VelRefold1;
        If (DelV>MaxDelV) DelV=MaxDelV;
        VelRef1=VelRefold1+DelV;


//Speed Controller 1 Calculations
        VelErr1=VelRef1-V[27];
        Tp1=VelErr1*Ksp1;
        Ti1=VelErr1*Ksi1*DeltaTime+Ti1;
        T1=Tp1+Ti1;
        TorkOut1=T1;

//Controlled S-wrap Calculations
        Js=0.01/g;
        Ds=4.0/12.0;
        //Js=Inertia of Driven S-Wrap
        //Ds=Diameter of S-Wrap rolls
        N[27]=(TorkOut1*Ds/2.0)*DeltaTime/(2.0*Js)+N[27];
        V[27]=N[27]*Ds/2.0;
        //V[27]=VelRef1;

Sect7:
//**************************SECTION
7**********************************

//Span Calculations for Section 7
        Idler=12;
        Mstart=27;
        FOR M=27 to 39
        {
```

```
IF (M==28)
{
        Ri=Rdi4;
        Ji=Jdi4;
        Civd=Cdivd4;
        Ticd=Tdicd4;
}
IF (M==30)
{
        Ri=Rdi4;
        Ji=Jdi4;
        Civd=Cdivd4;
        Ticd=Tdicd4;
}
IF (M==32)
{
        Ri=Rdi4;
        Ji=Jdi4;
        Civd=Cdivd4;
        Ticd=Tdicd4;
}
ELSE
{
        Ri=Rdi2;
        Ji=Jdi2;
        Civd=Cdivd2;
        Ticd=Tdicd2;
}

K=M+1;
Str=(Z[M]-Lu[M])/Lu[M];
Rs_in=Rs[K];
Rs_out=Rs[M];
Lu[M]=(V[K]*Rs_in-V[M]*Rs_out)*DeltaTime+Lu[M];
F[M]=Modulus*Str*10.0/454.0;
```

```
            Rs[M]=Lu[M]/Z[M];

            If ((CurrentTime>(SpliceTime-8.0*DeltaTime)) and
(CurrentTime<(SpliceTime-2.0*DeltaTime)))
            F[M]=0.5;
            IF (M==Mstart+Idler)
            {
            Goto Dancer2;
            }
            IF (N[K]<0.0)
            {
                  N[K]=((F[M]-F[K])*Ri-
N[K]*Civd+Ticd)*DeltaTime/Ji+N[K];
            }
            ELSE
            {
                  N[K]=((F[M]-F[K])*Ri-N[K]*Civd-
Ticd)*DeltaTime/Ji+N[K];
            }
            V[K]=N[K]*Ri;
      }

Dancer2:
//Dancer 2 Calculations - Developed by Bub Stuebe

      Sheet_tork_pt2=F[27]*(10.0/12.0)+F[28]*(10.0/12.0)+F[29]*(13.0/12.0)+F[3
0]*13.0/12.0+F[31]*Rd2+F[32]*Rd2;
      Hptd2=Nptd2*DeltaTime+Hptd2;
      Nptd2=((Sheet_tork_pt2-2.0*Fpts2*Rd2-2.0*Fpts2*13.0/12.0-
2.0*Fpts2*10.0/12.0)-Nptd2*Kptdmp2)*DeltaTime/Jd2+Nptd2;
      IF (Hptd2>Hd_plim)
      {
            Hptd2=Hd_plim;

      }
      ELSE
```

```
        {
                IF (Hptd2<Hd_nlim)
                {
                        Hptd2=Hd_nlim;

                }
        }
        Z[27]=X[27]-Hptd2*10.0/12.0;
        Z[28]=X[28]-Hptd2*10.0/12.0;
        Z[29]=X[29]-Hptd2*13.0/12.0;
        Z[30]=X[30]-Hptd2*13.0/12.0;
        Z[31]=X[31]-Hptd2*Rd2;
        Z[32]=X[32]-Hptd2*Rd2;

        Hd_perr2=Hptd2-Hd_rpos;


//Dancer 2 Position Controller Calculations
        ViRef2=Hd_perr2*Ki2*DeltaTime*(V[22]/V22max)+ViRef2;
//v[22]/V22max scales velocity on ramp-up
        VelRef2=V[22]*Kff2+Hd_perr2*Kp2*(V[22]/V22max)+ViRef2+Nptd2*
Kd2;  //V[22]/V22max scales velocity on ramp-up
        //VelRef2=V[22]*Kff2;

//Speed Controller 2 Calculations
        VelErr2=VelRef2-V[40];
        Tp2=VelErr2*Ksp2;
        Ti2=VelErr2*Ksi2*DeltaTime+Ti2;
        T2=Tp2+Ti2;
        Tt2=(T2-TorkOut2)*DeltaTime+Tt2;

        //TorkOut2=T2;
```

```
//Plot Output Variables
        Con1Out=V[33];
        Con2Out=V[23];
        Con3Out=V[3];
        Con4Out=V[40];
        Con5Out=F[33];
        Con6Out=F[23];
        Con7Out=Hptd1;
        Con8Out=Hptd2; }
```

BIBLIOGRAPHY

Jaime-Esqueda, Manuel, Dynamic Modeling, Reliability Analysis, and Control of Startup Transients in High Speed Web Handling Equipment, Massachusetts Institute of Technology, May 1994.

Lin, K.C., Campbell, M.K. , Newton, J.P., WTS 6.5 User's Guide, Oklahoma State University Web Handling Research Center, June 1995.