

T-4448

**A TRANSIENT DYNAMIC DISCRETE MODEL
FOR CONVEYOR BELTS**

by

Andrew Iver Hustrulid

**ARTHUR LAKES LIBRARY
COLORADO SCHOOL OF MINES
GOLDEN, CO 80401**

ProQuest Number: 10783935

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10783935

Published by ProQuest LLC (2018). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

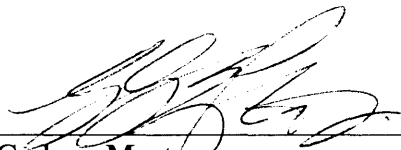
T-4448

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Applied Mechanics).

Golden, Colorado

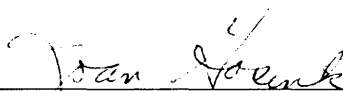
Date 11/16/93

Signed: 
Andrew Iver Hustrulid

Approved: 
Dr. Graham Mustoe
Thesis Advisor

Golden, Colorado

Date 11/14/93


Dr. Joan Gosink
Professor and Head
Department of Engineering

ABSTRACT

This thesis is concerned with the development of an advanced transient-dynamic discrete model for conveyor systems that can be used to improve conveyor design and/or isolate problems in existing complex conveyor systems. The new method features two dimensional non-linear transient dynamic modeling with explicit modeling of the dynamic motion of individual components and the contact between the various conveyor components including the belt, pulleys, and idlers. Previous models have omitted this level of detail. Validation of this numerical model has been performed with a number of analyses on a simple test example. The results of these analyses are in close agreement with analytical solutions. Through the object-oriented programming techniques used in the generation of this model it will be possible to expand its capabilities to model more complex conveyors in the future.

It is envisioned that this model can be used in conjunction with existing experimental tests to improve existing design methodology in conveyor engineering. For example to eliminate belt lift off in vertical curves by using drives and or takeups with feedback controls designed with this new model.

TABLE OF CONTENTS

	Page
ABSTRACT.	iii
LIST OF FIGURES.	vii
LIST OF TABLES.	x
ACKNOWLEDGMENTS.	xi
Chapter 1. INTRODUCTION.	1
1.1 Conveyor Transport.	1
1.2 Conveyor Components	4
1.2.1 Belting.	4
1.2.2 Pulleys	8
1.2.3 Takeups.	9
1.2.4 Drives.	10
1.2.5 Idlers.	11
1.3 Conveyor Operating Considerations.	13
1.4 Standard Belt Tension Analysis.	15
1.5 Statement of the Thesis Problem	16
Chapter 2. STATIC MODELING METHODS.	17
2.1 Introduction.	17
2.2 ANSI/CEMA Standard	18
2.2.1 Calculation of Belt Effective Tension	18
2.2.2 Calculation of Tension at Any Point in a Belt	22
2.3 DIN 22101 Standard and ISO 5048 Standard	22

2.3.1	Calculation of Belt Effective Tension	23
2.3.2	Calculation of Tension at Any Point in a Belt	26
2.4	Computer Models Used in Industry	26
2.4.1	PRO-BELT	27
2.4.2	Belt Conveyor Program	27
2.4.3	BELTSTAT	27
2.5	Conclusions	28
Chapter 3.	DYNAMIC MODELING METHODS	29
3.1	Introduction	29
3.2	Wave-Equation Modeling	30
3.3	Discrete Models	33
3.4	Conclusions.	34
Chapter 4.	OBJECT MODELING METHOD	36
4.1	The Belt Object	36
4.1.1	Belt Coefficients.	36
4.1.2	Geometric Relations.	40
4.1.3	Equations of Motion for the Belt Object.	42
4.2	The Pulley, Takeup, Drive and Idler Objects	52
4.2.1	The Pulley Object	53
4.2.2	The Takeup Object.	53
4.2.3	The Drive and Brake Objects	55
4.2.4	The Idler Object.	56
4.3	Contact Checking	56
4.3.1	Pulley Contact.	57
4.3.2	Idler Contact.	66
Chapter 5.	OBJECT ORIENTED PROGRAMMING CONSIDERATIONS	71

5.1	Introduction to Object Oriented Programming	71
5.2	The Conveyor Class.	71
5.3	The Pulley, Takeup, and Drive Classes	73
5.4	The Idlers Class.	77
5.5	The Belt Class	77
5.6	Using Object-Oriented Techniques to Expand the Model	78
5.7	Conclusions.	78
 Chapter 6. TESTING THE OBJECT MODEL		 79
6.1	Initial Conditions	80
6.2	Stabilization.	81
6.3	Belt Tensions at Static Equilibrium	82
6.4	The Dynamic Response of the System	85
	6.4.1 Dynamic Response of Increasing Takeup Mass	86
	6.4.2 Dynamic Response to a 400 kN-m Torque	87
	6.4.3 Dynamic Response to a 400 kN-m Torque With Axial Damping	88
 Chapter 7. CONCLUDING REMARKS		 91
7.1	Conclusions.	91
7.2	Suggestions for Further Work.	92
 REFERENCES CITED		 94
 APPENDIX.		 96

LIST OF FIGURES

	Page
1.1 54-inch conveyor carries large lumps of abrasive ore on incline	2
1.2 96-inch conveyor at high-capacity coal-loading facility	3
1.3 Nomenclature of components of a typical belt conveyor	4
1.4 Cross-section of a fabric-reinforced belt	5
1.5 Reduced-ply belt.	6
1.6 Steel-cable belt -- all-gum construction	7
1.7 Steel-cable belt -- fabric reinforced construction	7
1.8 Welded steel pulley with grooved lagging	8
1.9 Vertical automatic gravity takeup on an inclined conveyor	10
1.10 Typical speed torque curves for 50-hp, 1800-rpm NEMA design B and C motors; ideal speed torque curve for a typical belt conveyor drive	11
1.11 35° troughing idler	12
1.12 Return belt idler	12
2.1 Variation of temperature correction factor, K_t , with temperature	19
2.2 General section of a conveyor	22
3.1 Unfolded belt for use with the wave equation	30
3.2 Conveyor profile used for wave equation model	31

3.3	Steady state tensions of the conveyor belt	31
3.4	Belt velocities at T_1 and T_2 during stopping	32
3.5	Belt tensions at T_1 and T_2 during stopping	32
3.6	Conveyor Dynamics five-element composite model	34
4.1	Kelvin-Voigt element	37
4.2	Typical belt section	38
4.3	Bending beam	39
4.4	Belt variables in initial and current configurations	40
4.5	Gravity takeup object	54
4.6	Contact between the belt and a pulley	57
4.7	Notation for contact between the belt and a pulley	58
4.8	Friction between the belt and a pulley	60
4.9	Belt/Pulley friction algorithm	65
4.10	Contact between and idler and a belt element	66
5.1	Structure of conveyor class	72
5.2	Pulley class definition	75
5.3	Takeup class definition	76
6.1	System modeled using object model	79
6.2	Initial tension distribution in the belt	80
6.3	Total kinetic and potential energy during the stabilization process	81

6.4	Belt tensions after stabilization with zero contact friction	82
6.5	Belt tensions after 2000 time steps with contact friction	83
6.6	Belt tensions after stabilization with friction	84
6.7	Evolution of the belt tensions during stabilization process with friction	85
6.8	Dynamic tensions resulting from an instantaneous 10% increase in takeup mass	86
6.9	Takeup response from an instantaneous 10% increase in takeup mass	86
6.10	Belt tensions resulting from a 400 kN-m torque applied to the upper pulley for 0.07 seconds	87
6.11	Tensions at two point in the belt versus time	88
6.12	Belt tensions resulting from a 400 kN-m torque applied to the upper pulley for 0.07 seconds with axial belt damping of 10% of critical	89
6.13	Tensions at two point in the belt versus time with belt damping of 10% of critical	90

LIST OF TABLES

	Page
Table 2.1 <i>A</i> and <i>B</i> values for equation [2.3]	21
Table 2.2 Standard values for the coefficient <i>f</i> for belt conveyor installations with filling ratios in the range of 70 to 110%	24
Table 2.3 Standard values for the coefficient <i>C</i> for belt conveyor installations with filling ratios in the range of 70 to 110%	25

ACKNOWLEDGMENTS

The author is sincerely grateful to the many people who have made contributions to this investigation. I would particularly like to thank:

Dr. G.G.W. Mustoe and Dr. A. Harrison for their advice, guidance, ideas, and encouragement throughout the investigation.

Dr. W. A. Hustrulid for his support, editing, and inspiration.

Mr. R. Lloyd from The Conveyor Equipment Manufacturers Association for permission to use the material on conveyor belts contained in Chapter 1.

Mr. J. Hinson from Georgia Duck & Cordage Mill for providing belt moduli for their line of belts.

Mr. G. Dewicki, a fellow graduate student at CSM, for translating the book written by Markusik from Polish to English.

Miss L. Werth for checking the Chapters, Figures, Table, and Equations titles and numbers.

Chapter 1.

INTRODUCTION

Prior to beginning the analysis section of this thesis it is important to briefly review conveyor transport and describe the different components making up a conveyor system. This material has been extracted from the Conveyor Equipment Manufacturer's Association (CEMA) Handbook entitled *Belt Conveyors for Bulk Materials* (1).

1.1 Conveyor Transport

Belt conveyors have attained a dominant position in transporting bulk materials. They can be arranged to follow an infinite number of profiles of paths of travel. Among these are conveyors which are horizontal, inclined, or declined, or, with the inclusion of concave and convex curves, any combination of these. As compared to the 6 to 8% effective limits for truck haulage they can follow grades up to 30 to 35%. A conveyor carrying abrasive ore on an incline is shown in Figure 1.1. Belt conveyor systems provide the means of transporting materials via the shortest distance between the required loading and unloading points. The length of the routes can be extended repeatedly, as required. In some open-pit mining operations, conveyors thousands of feet long are shifted laterally on the bench to follow the progress of excavation at the face.



Figure 1.1 54-inch conveyor carries large lumps of abrasive ore on incline

Belt conveyors are very flexible in their capabilities for receiving material from one or more locations and for delivering it to points or areas, as required by plant flow sheets. They are particularly useful in tunnels beneath stockpiles, from which they can reclaim and, when required, blend materials from various piles. Material can simply be discharged over the head end on each conveyor or anywhere along its length by means of plows or traveling trippers.

Belt conveyors are environmentally more acceptable than many other means of transporting bulk material. They operate quietly, often in their own enclosures, which prevent the escape of dust to the surrounding atmosphere.

Belt conveyors commonly serve vital process units whose very success depends on continuous operation, such as handling coal in power plants, and transporting raw bulk materials in steel plants, in cement plants, and to and from ships in ports, where downtime is very costly. A belt carrying coal at a loading facility is shown in Figure 1.2. Reliability

and safety are outstanding now that stronger and more durable belts, as well as greatly improved mechanical parts and highly sophisticated electrical controls and safety devices are available.

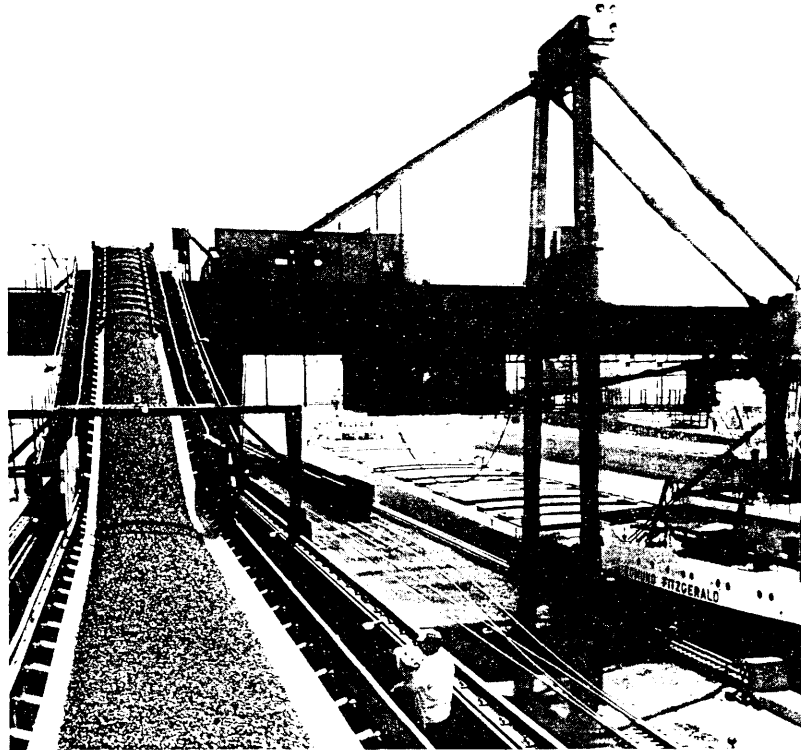


Figure 1.2 96-inch conveyor at high-capacity coal-loading facility

✧ Since most functions of the systems can be monitored from a central control panel or controlled by computer, only a minimum number of operating personnel are needed to inspect the equipment and report conditions that may require attention by the maintenance department. Hence the labor hours per ton required to operate belt conveyor systems are usually the fewest of any method of transporting bulk materials.

In summary, conveyor belts can economically transport various bulk materials at high flow rates, across difficult terrain, with minimal impact on the environment. When properly designed, built, and maintained they offer an attractive alternative to truck or rail transport for long distance applications.

1.2 Conveyor Components

This section describes the different components making up a conveyor system. The nomenclature for typical belt conveyor components is illustrated in Figure 1.3.

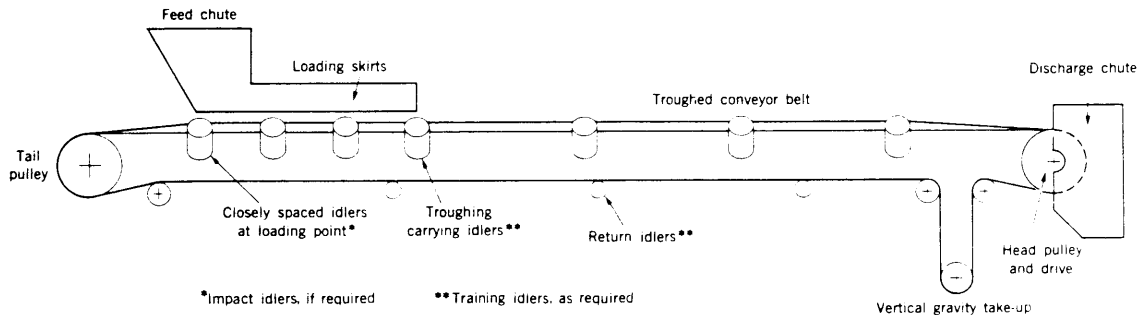


Figure 1.3 Nomenclature of components of a typical belt conveyor

1.2.1 Belting

In general, a conveyor belt consists of three elements: top cover, carcass, and bottom cover. Figure 1.4 illustrates a cross section of a typical belt. The primary purpose of the covers is to protect the belt carcass against damage and any special deteriorating factors that may be present in the operating environment. The belt carcass carries the tension forces necessary in starting and moving the loaded belt, absorbs the impact energy of material loading, and provides the necessary stability for proper alignment and load support over idlers under all conditions of loading.

Rubber or rubber-like compounds are used for the top and bottom covers of conveyor belting and for bonding together various components of the belt carcass. These compounds are produced by mixing rubbers or elastomers with various chemicals in order to obtain reinforcement and to develop the physical properties necessary for service conditions.

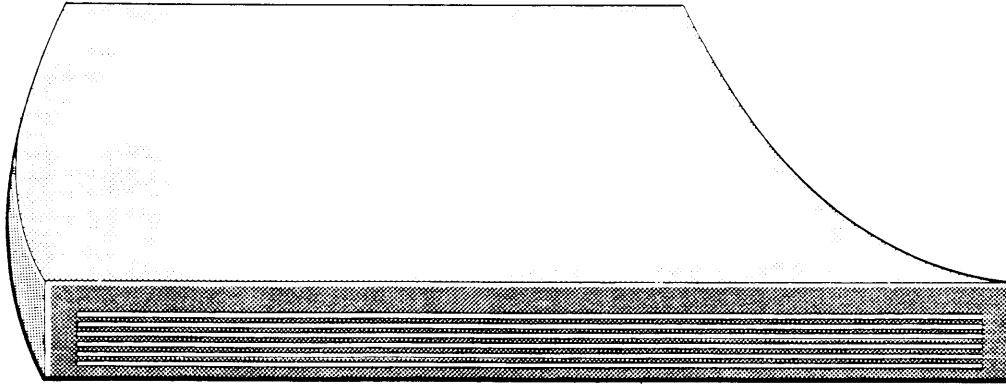


Figure 1.4 Cross-section of a fabric-reinforced belt

Because the primary function of the cover is to protect the carcass, it must resist the wearing effects of abrasion and gouging, which vary according to the type of material conveyed. The top cover will generally be greater in thickness than the bottom cover because the concentration of wear is usually on the top, or carrying side. However, specific characteristics of the material to be conveyed and the operating conditions may require a belt with equal cover thickness on top and bottom.

The belt carcass is the tension element of a conveyor belt. It is the primary reinforcement for belt tear resistance, impact resistance, load support, and mechanical fastener-holding ability. Most conveyor belt carcasses are made of one or more plies of woven fabric. Some high-tension carcasses employ a single layer of parallel steel cables.

Conveyor belt fabric is made of warp yarns, which run lengthwise, and weft yarns, or filling, which run crosswise or transversely. The type of textiles used by belt manufacturers vary greatly. Cotton, viscose rayon, nylon, and polyester are widely used, either in pure form or in various blends and combinations. Typical combinations are cotton warp-nylon filling, rayon warp-nylon filling, and polyester warp-nylon filling. Four types of weave patterns are commonly used: plain weave, straight-warp weave, solid-woven weave, and woven cord weave.

The major types of conveyor belt carcass are multiple-ply, reduced-ply, steel-cable, and solid-woven.

Multiple-ply belt carcasses

The carcass of the multiple-ply belt is usually made up of three or more plies, or layers, of woven belt fabric which are bonded together by an elastomeric compound. Belt strength and load support characteristics vary according to the number of plies and the fabric used, but practical considerations usually limit the number of plies in a carcass to a maximum of eight. The multiple-ply conveyor belt was the most widely used type through the mid-1960s, but today it is often supplemented by reduced-ply belting.

Reduced-ply belts

These belts consist of carcasses with either fewer plies than comparable multiple-ply belts, or special weaves representing a total departure from the ply concept. Figure 1.5 shows a reduced-ply belt. The textile carcasses of such belts comprise high-strength synthetic fibers - usually nylon, polyester, or combinations - in plain-weave or special fabric designs. Load support is provided by extra thick rubber skin layers between plies and/or in the weave design of the fabric itself.

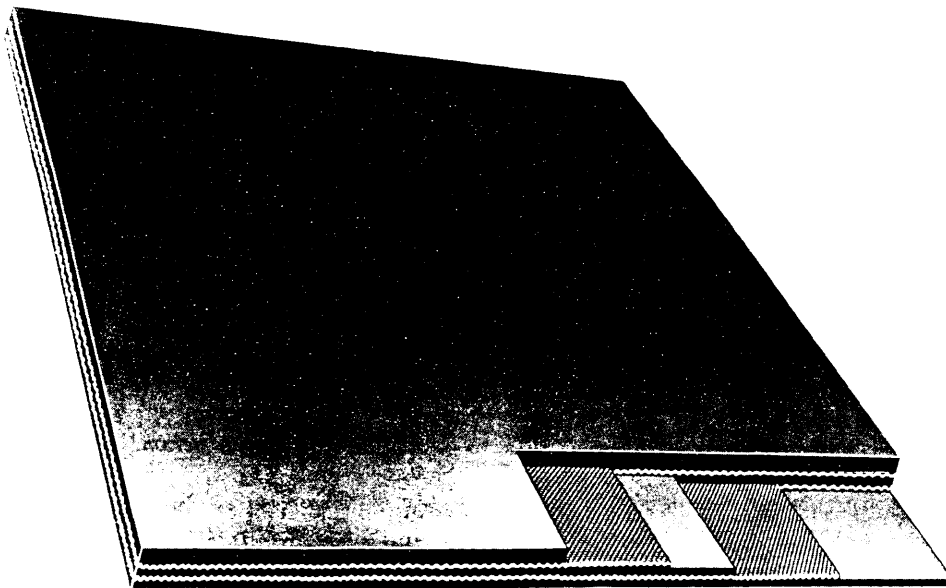


Figure 1.5 Reduced-ply belt

Steel-cable belts

Steel-cable conveyor belts are made with a single layer of parallel steel cables, completely imbedded in rubber, as the tension element. The carcass of steel-cable belting is available in two types of construction, depending on the manufacturer and the service conditions. The all-gum construction uses only cables and cable rubber, as shown in Figure 1.6. Fabric reinforced construction has one or more plies of fabric above and below the cables, but separated from the cables by the cable rubber (See Figure 1.7). Breakers may be included in the fabric-reinforced type. Both types have appropriate top and bottom covers.

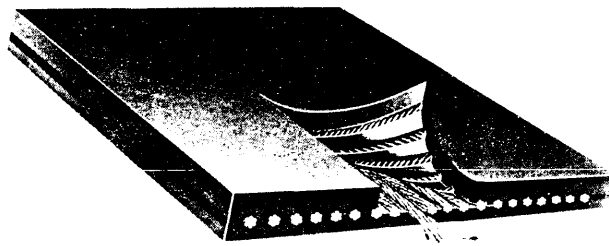


Figure 1.6 Steel-cable belt -- all-gum construction

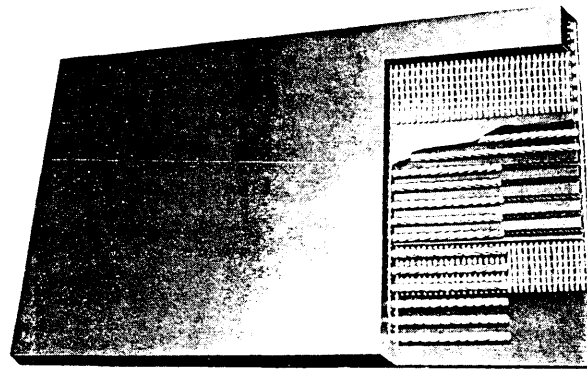


Figure 1.7 Steel-cable belt -- fabric reinforced construction

Steel-cable belting is produced using a broad range of cable diameters and spacing, depending primarily on the desired belt strength. This type of belting is often used in applications requiring operating tensions beyond the range of fabric belts and/or in installations where takeup travel limitations are such that changes in the length of a fabric belt cannot be accommodated.

Solid-woven belts

This type of belting consists of a single ply of solid-woven fabric, usually impregnated and covered with PVC, with relatively thin top and bottom covers. Abrasion resistance is provided by the combination of PVC and the surface yarns of the fabric. Some belting is produced with heavier covers, and thus is not dependent on the fabric for abrasion resistance.

1.2.2 Pulleys

The standard steel pulley is the most commonly used conveyor pulley (See Figure 1.8). They are manufactured in a wide range of sizes and consist of a continuous rim and two end disks fitted with compression hubs. Standard steel drum pulley diameters are: 6, 8, 10, 12, 14, 16, 18, 20, 24, 30, 36, 42, 48, 54, and 60 inches. These nominal diameters are for bare pulleys only, and do not include any increases in diameter by lagging.

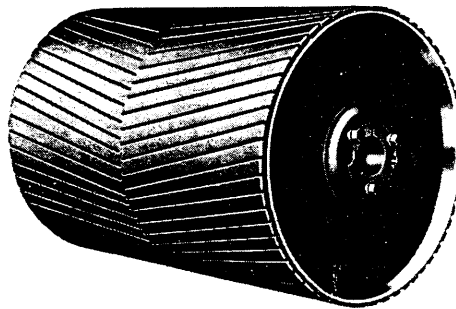


Figure 1.8 Welded steel pulley
with grooved lagging

Conveyor pulleys can be covered with some form of rubber, fabric or other material. One use of lagging is to increase the coefficient of friction between the belt and the pulley. Another purpose is to reduce wear on the pulley face and to effect a self-cleaning action on the surface of the pulley. Drive pulleys which perform in wet or damp conditions are often grooved. These grooves commonly take the shape of a herringbone or chevron-shaped pattern cut into the lagging. In both patterns, the apex points in the direction of belt travel; the purpose is to improve traction between the belt and the pulley. Lagging is always specified for drive pulleys and is preferred on pulleys in contact with the dirty side of the belt.

1.2.3 Takeups

All properly designed belt conveyors require the use of some form of takeup device. The reasons for this are: (1) To insure the proper amount of slack-side tension, T_2 , at the drive pulley to prevent belt slippage; (2) To insure proper belt tension at loading and other points along the conveyor (necessary to prevent loss of troughing contour of the belt between idlers, thus avoiding spillage of the material from the belt); (3) To compensate for changes in belt length; (4) To allow belt storage for making replacement splices (without which storage, small sections of new belt would have to be added, requiring two splices for each splice repair).

The takeup should provide sufficient movement to accommodate acceleration or deceleration surges without having the takeup strike against its stops. It should also allow for some "live" storage of belting so that, in case of an accident, it may not be necessary to splice in a short length of belt, which would require two splices. In addition, takeup movement should provide for changes in belt length due to stretch or shrinkage.

Automatic takeups are the most desirable type for use on any belt conveyor. They can be installed horizontally, vertically, or on an incline. They can be either gravity-operated or power-operated by hydraulic, electric, or pneumatic means. The most common type is the gravity takeup (See Figure 1.9).

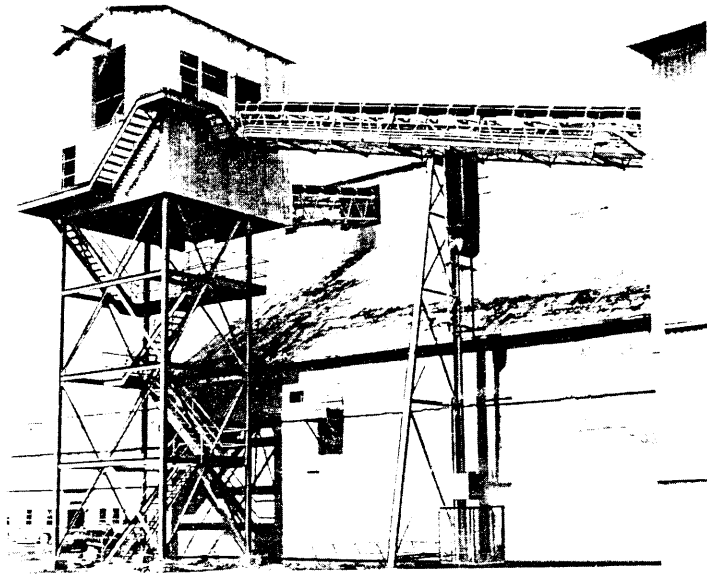


Figure 1.9 Vertical automatic gravity takeup on an inclined conveyor

1.2.4 Drives

A belt conveyor drive must provide sufficient torque at standstill to overcome the static forces and to accelerate the loaded conveyor to running speed within the time limitation imposed by the motor manufacturer. Likewise the acceleration torque must not impart tensions to the conveyor belt which exceed the manufacturer's allowable rating for acceleration. The amount of torque imparted to the belt will vary depending on the relative inertial values of the drive and the moving parts of the conveyor system as well as the profile of the conveyor.

Squirrel-cage alternating current induction motors represent the simplest and most economical means of driving belt conveyors. Unfortunately, The National Electrical Manufacturers Association (NEMA) standard motor designs do not exactly meet ideal speed torque conditions required by a belt conveyor. Other means have to be employed to control the torque. These are usually reduced-voltage control or auxiliary devices. An ideal speed torque curve for a belt conveyor is shown in Figure 1.10.

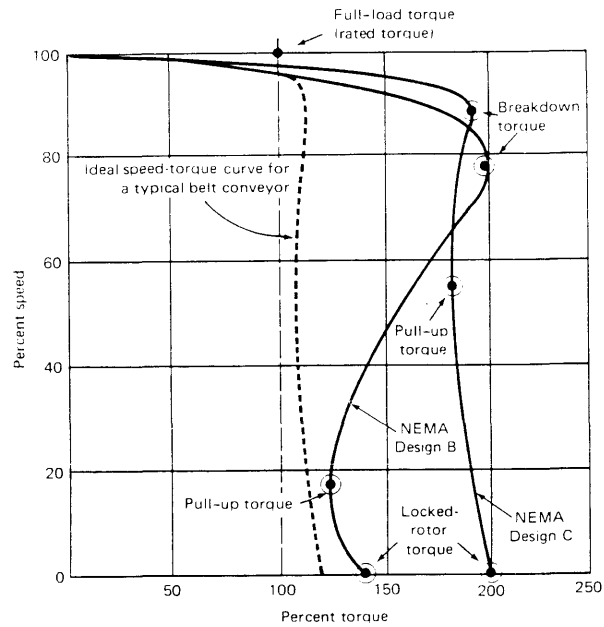


Figure 1.10 Typical speed torque curves for 50-hp, 1,800-rpm NEMA Design B and C motors; ideal speed torque curve for a typical belt conveyor drive

NEMA classifies polyphase squirrel-cage motors into four designations with respect to speed-torque characteristics. Of these, two designs, B and C, satisfy all but a few unusual applications. Typical speed-torque curves are also shown in Figure 1.10.

1.2.5 Idlers

There are two basic types of belt conveyor idlers: carrying idlers, which support the loaded run of the conveyor belt; and return idlers, which support the empty return run of the conveyor belt.

Carrying idlers are of two general configurations. One is used for troughed belts and usually consists of three rolls (See Figure 1.11). The two outer rolls are inclined upward; the center roll is horizontal. The other configuration is used for supporting flat belts. This idler generally consists of a single horizontal roll positioned between brackets which attach directly to the conveyor frame.

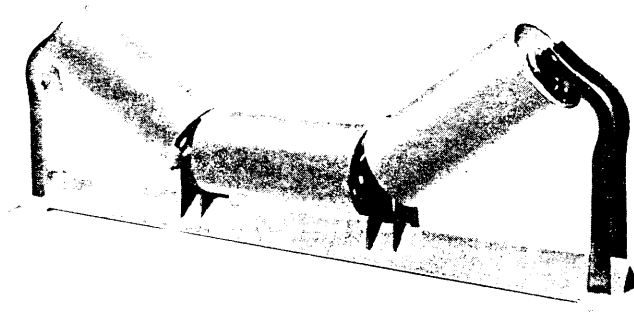


Figure 1.11 35° troughing idler

Return idlers usually are horizontal rolls, positioned between brackets which normally are attached to the underside of the support structure on which the carrying idlers are mounted (See Figure 1.12). Two-roll "V" return idlers are also used for better training and higher load ratings.



Figure 1.12 Return belt idler

The rolls, fitted with anti friction bearings and seals, are mounted on shafts. Frictional resistance of the idler roll influences belt tension and, consequently, the horsepower requirement. Roll diameter, bearing design, and seal requirements constitute the major components affecting frictional resistance.

1.3 Conveyor Operating Considerations

For most of the operating time the belt may be considered to be under steady state (acceleration = 0) conditions with corresponding belt tensions. One important aspect of the CEMA design guidelines is the determinations of these tensions. Special consideration must be given to the additional forces imposed by accelerating from and decelerating to stop. These may result in early splice failure, excessive takeup travel, and other difficulties. The types of acceleration/deceleration effects experienced and a brief discussion of each are indicated below:

Belt Stress

Economy of design dictates the selection of a belt having a carcass strength at or near the normal operating tensions. Consequently, the additional forces resulting from acceleration or deceleration may overstress the belt or its splices, particularly if mechanical splices are used. While this problem is most likely to exist with respect to the belt, there also is the possibility of over stressing the mechanical components such as pulleys, shafts, bearings takeups, etc.

Vertical Curves

Two different problems may be encountered with vertical curves. In the case of concave curves (where the center of curvature lies above the belt) if belt tensions are too high during starting, the belt will lift off the troughing idlers. It is necessary to analyze this problem in regard to full, partial, and no loads. In the case of convex vertical curves, where the center of the curvature lies below the belt, there is the possibility of overloading certain idlers.

Loss of Tension Ratio

During both acceleration and deceleration there exists the distinct possibility of losing the required T_1/T_2 ratio, where T_1 and T_2 are the high and low tensions across the drive pulley respectively, necessary to maintain the desired control of the engagement of the belt and drive pulley. This particularly is true if the takeup is located far from the drive.

If a screw takeup is used and improperly adjusted or the travel of a gravity takeup is too limited, the necessary ratio T_1/T_2 may be lost during the attempt to accelerate the belt conveyor.

During deceleration, the effect of the inertia load may cause a loss of the T_1/T_2 ratio necessary to transmit braking forces from the braking pulley to the belt. This would permit the continued motion of the belt and the load, after the pulley had been stopped.

Load Conditions on the Belt

The belt conveyor may operate satisfactorily during stopping or starting if completely loaded or empty. This, however, may not be so if only portions of the conveyor length are loaded. For example, when a belt conveyor contains a concave curve, a critical condition of starting may be the lifting of the belt at the curve during acceleration because the portion of the belt ahead of the concave vertical curve is loaded, while the remainder of the belt is not.

Coasting

Where there is a system of belt conveyors transferring from one to another, sequence starting or stopping is almost always a prerequisite of design. As an example, a belt with very long centers may transfer to a belt of shorter centers, in which case the time required to decelerate the two belts must obviously be synchronized, despite the differences in the braking forces required. During the accelerating period, the same synchronization is necessary. In either case, the consequences of not making a proper analysis and providing the necessary controls will result in a pile-up at the transfer point and possible destruction of the machinery and belt, plus an inoperative system.

Takeup Movement

During both the acceleration and deceleration cycles, where counterweighted takeups are used, the takeup travel may be insufficient unless these forces are considered. The engineer must consider not only the length of travel, but also the rate of travel, particularly when hydraulic, electric, or pneumatic controls are involved.

Festooning

Without proper consideration of the starting and stopping forces, it is possible that the belt tension may drop to a point, at some spot in the line, where the belt will festoon (buckle). The obvious result is load spillage, entanglement, loss of alignment, etc.

Braking Tensions Taken by Return Run and Tail Pulley

When deceleration is accomplished by means of a brake, the belt tension resulting from the braking force is taken in a direction opposite to that for driving the belt.

For instance, if the drive is at the head end of a horizontal or lift conveyor, power is transmitted from the drive pulley to the carrying side of the belt when the motor is energized. When decelerating with a brake connected to the drive pulley and the motor is de-energized, the braking force may be transmitted from the drive pulley to the return belt. The brake application, therefore, may be significant in determining the amount of the counterweight, the design of the takeup, and the shaft sizes.

1.4 Standard Belt Tension Analysis

In the CEMA approach, the belt system is treated as a rigid body with equivalent mass (W_e/g). The force is then calculated simply as the mass times the acceleration (deceleration). The recommended maximum tensions are expressed in terms of the allowable belt working tension. As an example, under starting conditions belt tension is allowed to reach 150% of the belt working tension. On conveyors with tensions under 75 lbs per ply inch or the equivalent, this maximum can be increased to as high as 180%.

This is a very simple approach providing somewhat limited but rather useful design information. It is noted that the system is considered as "rigid" rather than "elastic". Introduction of "elasticity" into the system although a more true representation adds major complications in analysis. For example, because of the vast differences in carcass construction, from the standpoint of both materials used and methods of manufacture, no single numerical value can express belt stretch as a function of the applied force.

Conveyor belts are often a single link in a material transport system. When a belt is down from an unexpected failure or for repairs the whole system must often shut down.

Since lost production costs are significant; it is desirable to design conveyor systems that have high reliability.

Today, numerous simplifying assumptions are made to keep the engineering work within limits. However, because conveyor systems are continually getting larger, carrying higher loads, and conveying over longer distances more accurate designs are required.

1.5 Statement of the Thesis Problem

Dynamic analysis has become common in the design of modern conveyor belts. The current methods available to design engineers, however, are limited in several ways: (1) Belt/idler interactions are modeled by including the effective mass of the idlers in with the belt mass and applying rubber hysteresis losses as external friction forces; (2) Belt and material flexure losses are also modeled as external friction forces; (3) There is difficulty in incorporating laboratory data into working models. The work performed in this thesis develops a new discrete modeling technique implemented with object-oriented programming techniques which overcomes the limitations of many previous models. This powerful approach may be readily adapted for the analysis of even more complicated future systems.

The remainder of this thesis is organized as follows: Chapter 2 reviews the standards relating to static tension calculations; Chapter 3 reviews two methods of calculating dynamic tensions currently used in industry; Chapter 4 presents the theory and develops the equations of motion required for the new discrete dynamic Object Based Model; Chapter 5 shows the object-oriented programming techniques used; Chapter 6 presents the results from a simple conveyor using the Object Based Model; and Chapter 7 provides conclusions and recommendations for future work. The Appendix contains the computer code for objects making up the conveyor.

Chapter 2.

STATIC MODELING METHODS

A conveyor system operating at a constant velocity (acceleration = 0) is considered to be under "static" conditions by the conveyor belt industry. The calculation of the "static" tensions of a conveyor belt are published by several associations, standards committees, and belt manufacturers. Three standards, CEMA, DIN 22101 and ISO 5048, relating to the calculation of "static" belt tensions are reviewed in this chapter.

2.1 Introduction

There are several forces which influence the belt tensions around the conveyor. The forces acting along the whole length of the belt are; the lift or lowering of the belt mass and material, sliding resistance between misaligned or tilted idlers and the belt, idler bearing resistance, and energy loss from belt and material flexure. Forces which only act on a small portion of the belt are treated as localized effects and include the acceleration of the material as it is fed onto the belt, scraper, plows, and skirt board friction, pulley bearing friction and energy losses from bending the belt around the pulleys. In long belts the localized forces represent a small percentage of the total forces acting on the belt and can be ignored.

There are several terms used by the conveyor belt industry relating to the tensions in the belt. The T_1 tension is defined as the tension in the belt at the tight side of the driving pulley, the T_2 tension is the tension in the belt at the slack side of the driving pulley, and the effective tension, T_e , is the change in tension across the driving pulley or $T_1 - T_2$. The effective tension is the force needed to overcome all of the frictional forces and lift (or lower) the material on the belt, it is the force necessary to move the belt.

2.2 ANSI/CEMA Standard

The Conveyor Equipment Manufacturers Association (CEMA) is responsible for the North American standard on belt conveyors. This section presents the equations necessary for calculating the "static" tensions in the belt using the CEMA approach (1).

2.2.1 Calculation of Belt Effective Tension

The effective tension of a conveyor belt is found using equation [2.1]

$$T_e = LK_f(K_x + C_1K_yW_b g + 0.015C_1W_b g) + W_m g(C_1LK_y \pm H) + T_p + T_{am} + C_1T_{ac} \quad (2.1)$$

where

T_e = belt effective tension, (N)

L = length of conveyor, (m)

H = vertical distance that material is lifted or lowered, (m)

K_f = ambient temperature correction factor

K_x = factor used to calculate the frictional resistance of the idlers and the sliding resistance between the belt and idler rolls, (N/m)

K_y = carrying run factor used to calculate the resistance of the belt and the load to flexure as the belt and load move over the idlers.

C_1 = friction modification factor for declining conveyors

W_b = belt mass, (kg/m) of belt length

W_m = material mass, (kg/m) of belt length

T_p = tension resulting from resistance of belt to flexure around pulleys and the resistance of pulleys to rotation on their bearing, total for all pulleys, (N)

T_{am} = tension resulting from the force to accelerate the material continuously as it is fed onto the belt, (N)

T_{ac} = total of the tension from conveyor accessories, (N)

g = acceleration due to gravity, 9.81 (m/s²)

Equation [2.1] requires four coefficients, K_t , K_x , K_y , and C_I . Each will be discussed briefly below.

K_t - Ambient Temperature Correction Factor

As the ambient temperature decreases, the power required to drive a conveyor increases. The temperature correction factor, K_t , accounts for these changes. The factor is applied to the forces resulting from belt idler interaction and belt flexure. Figure 2.1 shows the K_t values for a range of temperatures.

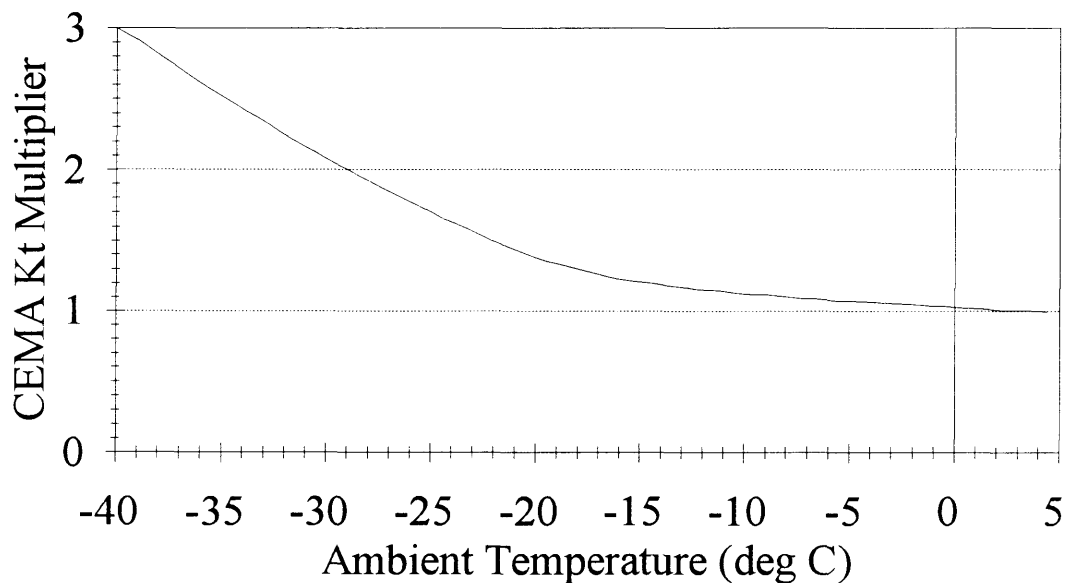


Figure 2.1 Variation of temperature correction factor, K_t , with temperature

K_x - Idler Friction Factor

The idler friction factor, K_x , accounts for the losses from idler bearing friction, seal friction, milling or churning of the grease in the bearings, and sliding resistance due to idler misalignment. Idlers are often intentionally misaligned to aid in belt training. Equation [2.2] shows the calculation of K_x .

$$K_x = 0.00068(W_b + W_m)g + \frac{A_i}{S_i} \quad (2.2)$$

where

S_i = carry idler spacing, m

A_i = force required to overcome frictional resistance and rotate idlers, N

= 6.67 for 15.24 cm (6 in) dia. idler rolls, CEMA C6, D6

= 8.01 for 12.70 cm (5 in) dia. idler rolls, CEMA A5, B5, C5, D5

= 10.23 for 10.16 cm (4 in) dia. idler rolls, CEMA A4, B4, C4

= 10.68 for 17.78 cm (7 in) dia. idler rolls, CEMA E7

= 12.45 for 12.24 cm (6 in) dia. idler rolls, CEMA E6

For regenerative declined conveyors, $A_i = 0$

If two roll vee return idlers are used, increase A_i value by 5%.

Close inspection of the terms of equation [2.1] shows that CEMA has lumped the idler friction for both the carry and return runs into K_x . This is acceptable when calculating the effective tension but introduces inaccuracies when finding tensions at any particular point in the conveyor.

K_y - Factor for Calculating the Force of Belt and Load Flexure Over the Idlers

As the belt and material deflects and changes shape passing over the idlers energy is lost. The losses due to belt flexure are a function of the belt construction, cover thickness and indentation by the idler rolls, type of rubber compound, idler roll diameter, temperature and other factors. The K_t coefficient accounts for the effects of temperature on K_y . Losses due to material flexure are a function of belt tension, type of material, shape of material cross section, and idler spacing. Research has shown that the belt tension is much more significant in determining K_y than changes in the material handled. The CEMA handbook contains various tables for K_y . The values are dependent on conveyor length, slope, and material and belt masses. Equation [2.3] is a general formula for calculating K_y where Table 2.1 provides A and B . The calculation of K_y is dependent on belt tension and may therefore take several iterations.

$$K_y = (W_m + W_b) \times A \times 10^{-4} + B \times 10^{-2} \tag{2.3}$$

where

A and *B* are determined from Table 2.1

Table 2.1 *A* and *B* values for equation [2.3]

Average belt tension		Idler Spacing, <i>A_i</i>									
		0.91 m (3.0 ft)		1.07 m (3.5 ft)		1.22 m (4 ft)		1.37 m (4.5 ft)		1.52 m (5 ft)	
kN	lbs	A	B	A	B	A	B	A	B	A	B
4.45	1000	1.4447	1.565	1.4753	1.925	1.4783	2.250	1.4825	2.584	1.4615	2.910
8.90	2000	1.2412	1.345	1.1186	1.744	1.0856	1.982	1.0511	2.197	1.0368	2.331
13.34	3000	1.0943	1.237	0.9856	1.593	0.9626	1.799	0.9538	1.991	0.9890	2.091
17.79	4000	0.9827	1.164	0.9085	1.465	0.8934	1.659	0.8903	1.825	0.9307	1.938
22.24	5000	0.8620	1.122	0.8014	1.381	0.7934	1.559	0.7937	1.714	0.8254	1.839
26.69	6000	0.7646	1.076	0.7217	1.318	0.7139	1.472	0.7164	1.627	0.7366	1.761
31.14	7000	0.6766	1.039	0.6349	1.256	0.6420	1.404	0.6576	1.549	0.6984	1.657
35.59	8000	0.6163	0.998	0.5747	1.194	0.5808	1.337	0.5964	1.472	0.6443	1.583
40.03	9000	0.5515	0.958	0.5376	1.120	0.5304	1.272	0.5603	1.388	0.5988	1.507
44.48	10000	0.4866	0.918	0.4947	1.066	0.4835	1.216	0.5255	1.314	0.5556	1.430
48.93	11000	0.4356	0.885	0.4460	1.024	0.4464	1.167	0.4956	1.238	0.5220	1.340
53.38	12000	0.3916	0.842	0.3916	0.992	0.4188	1.100	0.4536	1.180	0.4980	1.242
57.83	13000	0.3499	0.798	0.3522	0.938	0.3852	1.040	0.4152	1.116	0.4583	1.169
62.27	14000	0.3151	0.763	0.3232	0.897	0.3504	0.996	0.3743	1.069	0.4092	1.123
66.72	15000	0.2803	0.718	0.2977	0.841	0.3180	0.935	0.3480	1.006	0.3768	1.063
71.17	16000	0.2502	0.663	0.2665	0.780	0.2851	0.875	0.3084	0.958	0.3396	1.009

Equation [2.3] is valid for belt tensions under 71.2 kN (16,000 lbs). Above 71.2 kN a value of $K_y = 0.016$ is reasonably accurate. For return sections a value of $K_y = 0.015$ is used.

C_f - Friction Modification Factor for Declining Conveyors

If the net change in height from the tail to the head of a conveyor is negative the system has the possibility of becoming regenerative. When a conveyor becomes regenerative the drive acts as a brake. In sizing the motor it is imperative that the motor is large enough to absorb the energy being generated by the conveyor. Several of the parameters used in the tension calculations are non-conservative for regenerative conveyors. The friction modification factor, *C_f*, is set to 0.5 through 0.7 for regenerative conveyors. For non-regenerative conveyors *C_f* = 1.

2.2.2 Calculation of Tension at Any Point in a Belt

In practice it is desirable to know the tension at any point along the conveyor. A general section of a conveyor is shown in Figure 2.2.

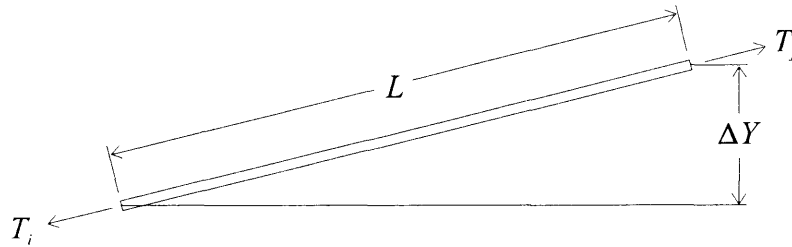


Figure 2.2 General section of a conveyor

The change in tension from point i to point j is given by equation [2.4].

$$\Delta T = T_j - T_i = LK_t(K_x + C_1K_yW_b g) + W_m g(C_1LK_y + \Delta Y) + W_b g\Delta Y \quad (2.4)$$

where

$$\Delta Y = Y_j - Y_i = \text{change in height of section}$$

The tension at any point in the belt is calculated by beginning at a known tension location, usually at the takeup, and working to the desired point in the belt using equation [2.4]. Localized forces resulting from skirtboard friction, pulley bearing friction, etc., are added to the tension in the belt as they are encountered. It is important to note that $K_x = 0$ and $C_1 = 1$ for sections of belt on the return run.

2.3 DIN 22101 Standard and ISO 5048 Standard

The German standard for conveyor belt design is DIN 22101, entitled *Belt conveyors for bulk materials, Bases for calculation and design (2)*. The current standard was updated in February of 1982. The International Standard, ISO 5048, entitled *Continuous mechanical handling equipment -- Belt conveyors with carrying idlers --*

Calculation of operating power and tensile force (3) was updated in 1989. These standards are discussed jointly in this section because they are essentially the same. This section focuses on the tension calculations.

2.3.1 Calculation of Belt Effective Tension

The overall resistance to motion of a belt conveyor, the effective tension, F , is subdivided into four groups; main resistances, F_H , secondary resistances, F_N , slope resistances, F_{St} , and special resistances, F_S .

$$F = F_H + F_N + F_{St} + F_S \quad (2.5)$$

Each of these components will be briefly discussed below.

F_H - Main Resistances

The main resistances account for (a) the resistances in idler bearings and seals, (b) the losses from the pressing down of the idlers into the belt and (c) the losses from belt and material flexure. The calculation of the main resistances assumes that there is a linear relationship between the resistance and the moving load. Equation [2.6] accounts for the main resistances in the carry and return section jointly.

$$F_H = L \cdot f \cdot g \cdot [m'_R + (2m'_G + m'_L) \cdot \cos \delta] \quad (2.6)$$

where

L = length of conveyor, (m)

f = hypothetical friction coefficient of the upper strand (carry side) and lower strand (return side) jointly

g = acceleration due to gravity, 9.81 (m/s²)

m'_R = effective mass of the upper and lower strand idlers combined per unit of belt length, (kg/m) of belt length

m'_G = belt mass, (kg/m) of belt length

m'_L = material mass, (kg/m) of belt length

δ = angle of inclination of the installation

Values of the hypothetical friction coefficient, f , for belts loaded between 70 and 110% of their nominal capacity are given in Table 2.2.

Table 2.2 Standard values for the coefficient f for belt conveyor installations with filling ratios in the range of 70 to 110%.

Horizontal installations, also installations conveying uphill and down gentle inclines (with electric motor drives):	
• favorable operating conditions, e.g. good belt alignment, easy running carrying idlers and material conveyed at low speeds, with low internal friction	0.017
• installations constructed and operated in normal (standard) manner	0.020
• adverse operating conditions, e.g. dust laden atmosphere, low temperatures, material conveyed exhibiting a very high internal friction, overloading, high speeds	0.023 to 0.027
• extremely low temperatures, but otherwise normally operated and conventionally constructed installations	up to 0.035
Installations conveying downhill at a steep incline ¹⁾ (drives operating as dynamos)	0.012 to 0.016
¹⁾ In the case of installations conveying downhill at a steep incline - drives operating as dynamos - the adoption of a somewhat lower value of f will result in a higher degree of safety in the design; in other cases - where the drives operate as motors - an enhanced safety of design is attained by adopting a higher value for f .	

F_N - Secondary Resistances

The secondary resistances result from the acceleration of the material, resistances between the feeder chutes and the material, belt cleaner resistances, and finally pulley bearing resistances. For conveyors longer than 80m the secondary resistances are calculated as a percentage of the main resistances.

$$C = 1 + \frac{F_N}{F_H} \quad (2.7)$$

where

C = coefficient for the all-inclusive consideration of the secondary resistances

Values of C for belts loaded between 70 and 110% of their nominal capacity are given in Table 2.3.

Table 2.3 Standard values for the coefficient C for belt conveyor installations with filling ratios in the range of 70 to 110%

L in m	80	100	150	200	300	400	500	600	700	800	900	1000	1500	≥ 2000
C	1.92	1.78	1.58	1.45	1.31	1.25	1.20	1.17	1.14	1.12	1.10	1.09	1.06	1.05

For installations shorter than 80 m or installations with more than one feeder point, it is necessary to calculate the individual components of F_N .

F_{St} - Slope Resistance

This force results from the lift or lowering of the material being conveyed. The slope resistance, F_{St} , is calculated as

$$F_{st} = H \cdot g \cdot m'_L \quad (2.8)$$

where

H = distance material is raised or lowered, m

($H > 0$ for uphill conveying, $H < 0$ for downhill conveying)

F_S - Special resistances

The special resistances account for idlers set at a tilt and for losses between the material conveyed and lateral chutes outside the feeder points. These forces occur only in special conveyor installations and are beyond the scope of this thesis.

2.3.2 Calculation of Tension at Any Point in a Belt

The ISO Standard does not include the calculation of the tensions at any point in the conveyor. The material in this section is taken from only the DIN Standard.

The calculation of the tensions at any point of the belt is carried out similarly to the discussion in section 2.2.2. The change in tension of a typical section of belt is given as

$$\Delta T = F_{H_i} + F_{N_i} + F_{St_i} + F_{S_i} \quad (2.9)$$

where the i subscript indicates the forces acting on the i^{th} section of belt.

The equation for the main resistances is given as

$$F_{Hi} = f_i \cdot g \cdot [m'_{Ri} + (m'_G + m'_{Li}) \cdot \cos \delta_i] \cdot l_i \quad (2.10)$$

At present it is not possible to give any precise values for f_i , but as an approximation one can use $f_i \approx f$.

The secondary resistances must be computed for each accessory adding tension to the belt. (ie. feeders, belt cleaners, etc.)

The slope resistances for a typical section of belt is given in equation [2.11].

$$F_{Sti} = g \cdot (m'_G + m'_{Li}) \cdot \sin \delta_i \cdot l_i \quad (2.11)$$

Again the special resistances are beyond the scope of this thesis.

2.4 Computer Models Used in Industry

This section briefly reviews the commercial computer programs available to do "static" conveyor belt tension calculations. The programs are in wide use by belt manufacturers, conveyor designers and end users.

2.4.1 PRO-BELT

This program is sold by Professional Designers & Engineers, Inc. of Boulder Colorado. The program does standard CEMA calculations, pulley shaft sizing, and rigid body estimations of the starting and stopping tensions based on CEMA. Metric (SI) and English units versions of the program are available.

2.4.2 Belt Conveyor Program

Creative Engineering of Bakersfield California have written a program simply entitled *Belt Conveyor Program*. The program does standard CEMA calculations in English and Metric (SI) units. The program also offers design optimization based on cost factors input by the user. Tensions based on the ISO 5048 standard can also be calculated. The manual which is provided with the demonstration version provides many tips and pitfalls that may occur while designing conveyors.

2.4.3 BELTSTAT

Conveyor Dynamics of Seattle Washington is a consulting company which does conveyor design and analysis. Their belt program, *BELTSTAT*, appears to be using a modified form of CEMA to calculate belt "static" tensions. The program output indicates that the idler friction coefficient, K_x , used in the CEMA calculations is being split between the carry and return runs. In CEMA, the idler friction coefficient for both the carry and return runs is lumped into the carry run of the belt. (See section 2.2.1.) The program also outputs the hypothetical friction factor, f , used in the ISO and DIN standards. Rigid body starting and stopping tensions are also calculated.

2.5 Conclusions

There are several standards worldwide which apply to conveyor belt design. The standards use friction factors to account for the losses from belt and material flexure, idler indentation losses, and idler bearing losses. These friction factors are determined from tables, equations, or simply from the experience of the person doing the calculations.

The standards have been written with the objective of calculating the "static" tensions in the belt by hand. The equations used are designed to keep the work required by the user within reasonable levels. The computer models presented in this section do nothing more than perform the hand calculations for the user. This type of computer usage decreases the time required to do the calculations but ignores the capabilities of the computers to do a more thorough analysis also in reasonable time.

Chapter 3.

DYNAMIC MODELING METHODS

Modeling techniques which simulate the conveyor behavior during transient periods (during starting and stopping) are referred to as dynamic models by the conveyor belt industry. Dynamic models are distinguished from static models by their use of something more advanced than rigid body motion to predict the behavior of the conveyor during transient periods.

3.1 Introduction

There are currently two techniques for dynamic modeling of conveyor belts. The first technique models the belt as a continuous medium and the exact solution of the wave equation for the given boundary conditions is solved. The second technique discretizes the belt into individual elements. The resulting system of equations are then solved using a numerical method. This chapter only introduces these methods, as little useful information has been published due to the commercial value. Analysis of the transient conditions provide a more accurate prediction of

- drive and brake slippage
- conveyor response to different types of drives and controls
- belt lift off in convex curves
- excessive belt tensions
- low belt tensions causing material spillage
- takeup motion

than simple rigid-body motion can provide. This information can then be used to calculate the load on the pulleys and conveyor structure. The starting curves used by the motor can

be changed to look at the effect on the conveyor system.

3.2 Wave-Equation Modeling

Several researchers (4,5,6) have proposed dynamic models based on the application of solutions to the wave equation. The conveyor belt is unfolded as shown in Figure 3.1. Harrison (4) is the only researcher who has developed this method in any detail.

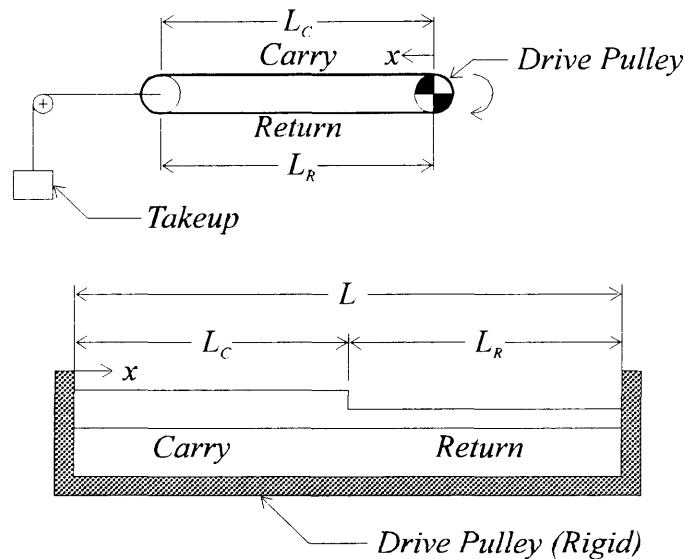


Figure 3.1 Unfolded belt for use with the wave equation

The carry section of the model is drawn thicker in Figure 3.1 to depict its increased weight due to the material on the belt. The wave equation is solved for the above model. The system shown is perturbed with a step function representing the belt stretch when beginning transients. The resulting waves are superimposed over the rigid-body motion of the system. The conveyor profile shown in Figure 3.2 has been solved using Harrison's wave model. The static tensions around the conveyor system are shown in Figure 3.3. Figure 3.4 and 3.5 show the transient tensions and velocities of the conveyor before and after the drive drum.

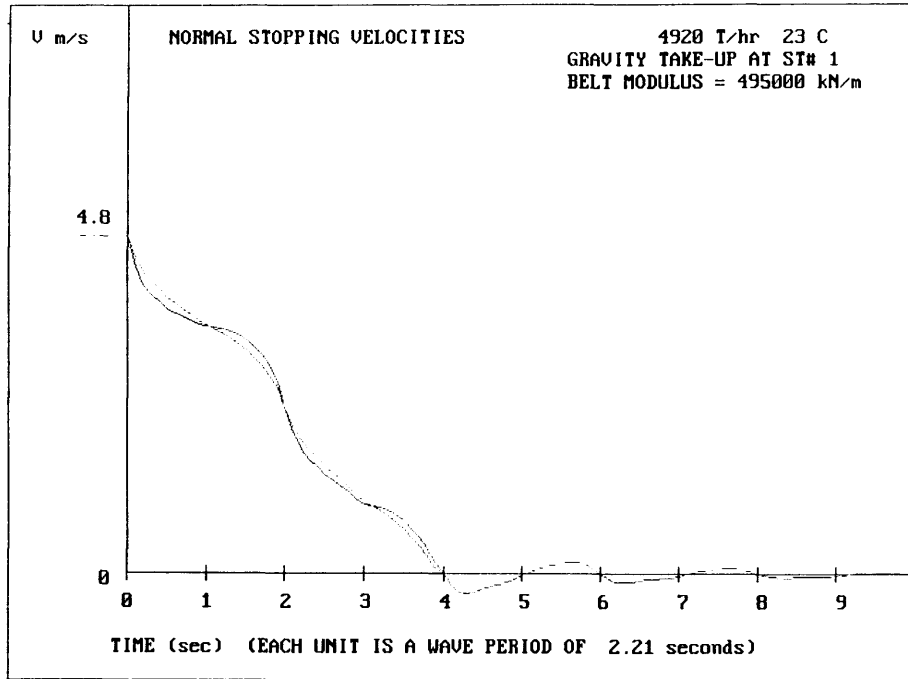


Figure 3.4 Belt velocities at T₁ and T₂ during stopping

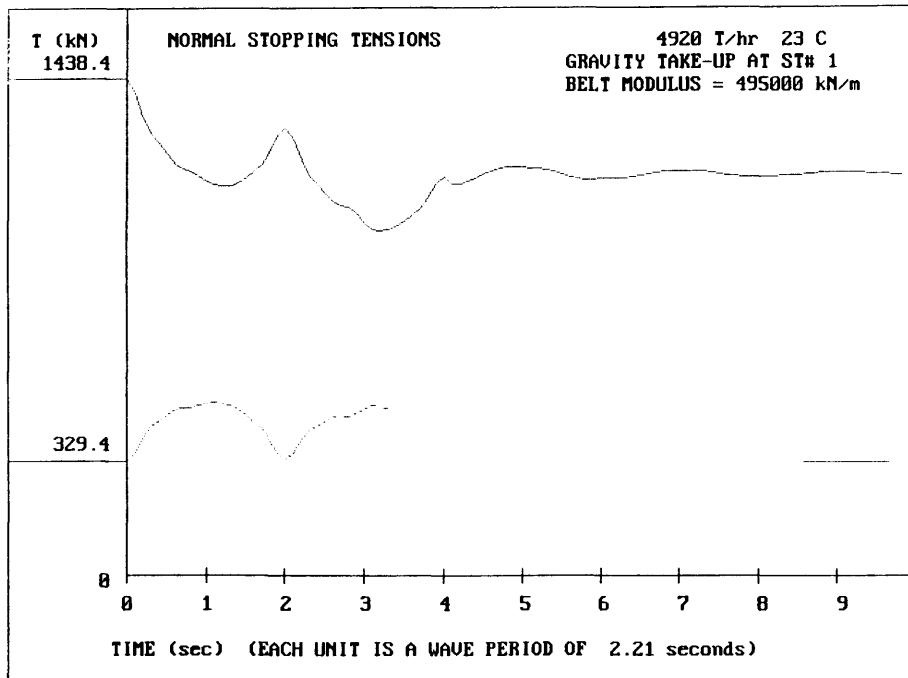


Figure 3.5 Belt tensions at T₁ and T₂ during stopping

The advantages of this method are the

- simplicity of the model
- speed in which solutions can be obtained

This method is however limited in that

- there is a difficulty in modeling various geometries
- different drive types cannot be easily modeled
- feedback control systems used in modern conveyor systems cannot be modeled
- booster drives cannot be modeled

3.3 Discrete Models

Discrete models break the belt up into spring-mass-damper discrete elements to model the characteristics of conveyor belts. Markusik (5), of Poland, has written *Dynamics of Setting in Motion Conveyors Belts with One or Two-Drum Frontal Propulsion*. Markusik's work includes derivations of the equations of motion for pulleys, electric drives with hydromatic clutches, gravity and winch takeups, along with two and three parameter belt elements. These components have been assembled and used to model several conveyor installations in the southern regions of Poland. The assembled differential equations, modeling parameters, modeling results and comparisons with measured data are all included in the book.

Winders, Barlow and Morrison (WBM) of Brisbane Australia and Denver Colorado has developed a dynamic conveyor belt program which they use commercially. Very little information has been published on the internal workings of the program. It is believed that it is similar to the work done by Markusik. WBM has however pioneered the use of 3-D graphs to show conveyor belt tensions and velocities around the whole belt. It is difficult to read specific values from these graphs but they are indispensable in determining if the belt model is numerically stable and where the maximum and minimum tensions occur around the belt.

Conveyor Dynamics Inc. (CDI) of Seattle Washington has also developed a dynamic conveyor belt program which they use commercially. Nordell, the president of

CDI, has published numerous papers on the general theory of the model, the belt element used in the model and results of the model (7,8). The modeling technique also appears to be similar to that presented by Markusik. The element used by CDI is shown in Figure 3.6.

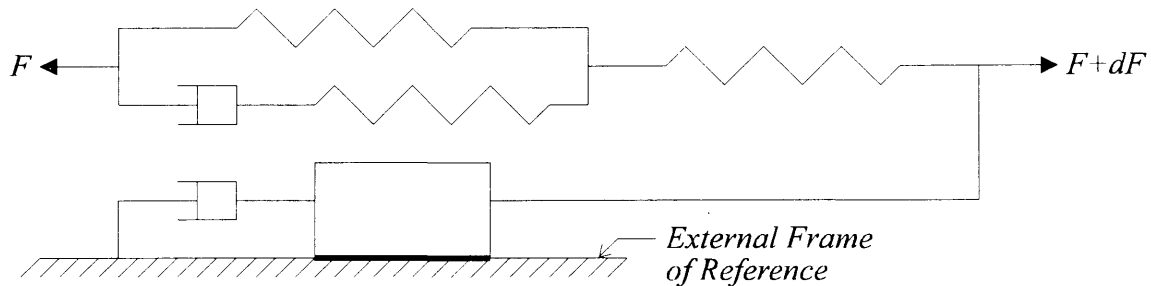


Figure 3.6 Conveyor Dynamics five-element composite model

The element is designed to incorporate the rheological properties of the belt, the elastic properties of the belt, and the belt idler interaction all into one element. It appears that the idler mass has been lumped into the belt mass and that the idler friction is simply applied as a sliding friction force. It is assumed that this sliding friction force also includes the forces acting on the belt from belt and material flexure. It is hard to imagine how all the values for the components of the element can be rationally determined when even the modulus of elasticity of the belt is only an estimate.

3.4 Conclusions

These methods have several inherent problems which have not been overcome to date. They include; mass dispersion, mass transport at booster drives, modeling of drive slip and belt lift off. These schemes also appear to employ an explicit time marching scheme whose time step is limited by the stiffest elements. If accurate drive and transmissions are modeled these will be considerably stiffer than the belt elements. A sub time marching scheme may be employed to increase the efficiency of this type of

modeling. These programs also seem to rely heavily on CEMA for the idler reactions.

All of the above methods are one-dimensional. This makes them inadequate to model belt lift off and other changes in geometry due to belt tensions. The above methods are also not easily expanded to two dimensions. They also do not truly model the idlers and pulleys which they essentially equate to an equivalent mass. The above methods also do not truly model the motion of the belt and the fact that it moves around the pulleys.

Chapter 4.

OBJECT MODELING METHOD

A unique new method for the dynamic modeling of conveyor belts is developed and presented in this section which eliminates many of the problems inherent with the previous models. The model is designed specifically for computer usage and makes use of object-oriented programming.

The conveyor is modeled in two dimensions. The belt, idlers, pulleys, drives and takeup are all idealized as discrete objects and their interactions are explicitly modeled. In two dimensions, belt lift off is modeled, the forces acting on individual idlers are calculated, and drive slip is modeled.

4.1 The Belt Object

The belting used with conveyors consists of three parts; the top cover, the belt carcass, and the bottom cover. The carcass carries the axial load in the belt while the covers protect the carcass from the environment and the material being conveyed. The covers are made of rubber or pvc (polyvinyl chloride plastic). The belt carcass consists of steel cables and/or one or more plies of woven fabric bonded together.

4.1.1 Belt Coefficients

The properties of the belting are orthotropic and considered to be rheological, meaning that the stress-strain relationship is time dependent. Several researchers have developed belt models which include rheological properties. The coefficients which must

be input into these models are not documented. Even the modulus of elasticity is not well defined for conveyor belts.

The author has considered the belt to be constructed of a series of Kelvin-Voigt elements consisting of a spring and dashpot in parallel (Figure 4.1).

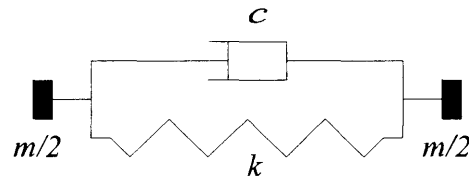


Figure 4.1 Kelvin-Voigt element

The axial stiffness of an element is

$$k = \frac{AE}{l^0} \quad (4.1)$$

where

A = cross sectional area of the belt

E = modulus of elasticity of the belt

l^0 = unstretched length of the element

The element mass, m , defined as

$$m = \rho A l^0 \quad (4.2)$$

where ρ = belt mass density

as shown in Figure 4.1, is split equally between the two end points.

The damping factor, c , is presented as a percentage (expressed as a ratio) of the value required for critical damping, c_{crit} , for the belt element.

$$c = 2m\zeta\sqrt{\frac{k}{m}} \tag{4.3}$$

where ζ = percentage of critical damping

The individual elements are connected in series with a pin joint and a rotational spring. A section of belt is shown in Figure 4.2. The last belt element connects to the first making the belt continuous. Belt shear is not modeled.

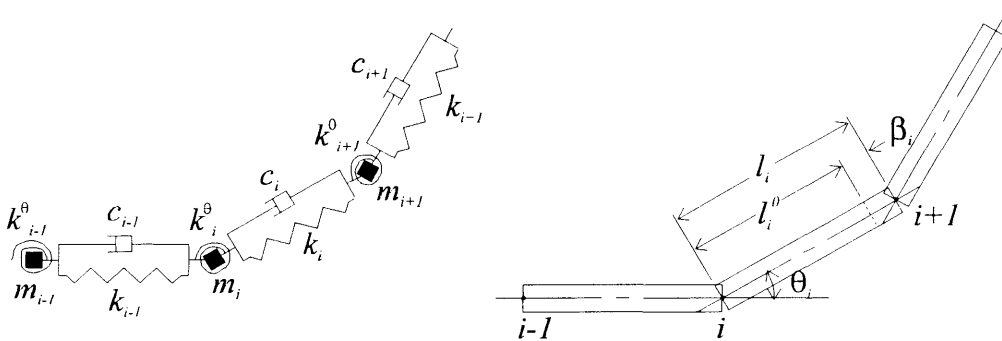


Figure 4.2 Typical belt section

When the elements are assembled the lumped mass, m_i , is the sum of half the mass from the element before and half after the node. The resulting expression for m_i is

$$m_i = \frac{\rho_{i-1} A_{i-1} l_{i-1}}{2} + \frac{\rho_i A_i l_i}{2} \tag{4.4}$$

The bending stiffness between elements is derived from beam theory. The virtual work for the continuous beam is,

$$\delta W = \int_{\Omega} \sigma_b \delta \varepsilon_b d\Omega - \delta \theta M = 0 \tag{4.5}$$

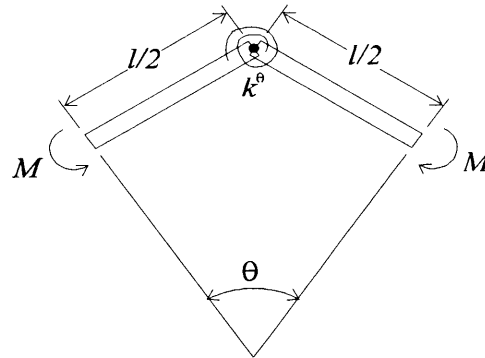


Figure 4.3 Bending beam

Using Hooke's law

$$\sigma_b = E\varepsilon_b \quad (4.6)$$

the strain at a point distance y from the center of the beam is

$$\varepsilon_b = \frac{y\theta}{l} \quad (4.7)$$

The incremental strain is further defined as

$$\delta\varepsilon_b = \frac{y\delta\theta}{l} \quad (4.8)$$

Substituting equations [4.6] through [4.8] into equation [4.5] and integrating over the volume yields

$$M\delta\theta = bl \int_{-h/2}^{h/2} \left(E \frac{y\theta}{l} \right) \frac{y\delta\theta}{l} dy = \frac{EI}{l} \theta\delta\theta$$

where

$$I = \frac{bh^3}{12}$$

The equation for virtual work for the element assemblage shown in Figure 4.3 is

$$\delta W = k^\theta \theta \delta \theta - M \delta \theta = 0 \quad (4.10)$$

Equating equation [4.9] and equation [4.10] one finds that

$$k^\theta = \frac{EI}{L} \quad (4.11)$$

It should be noted that a similar discrete element model that employs the same axial and rotational stiffness has been developed by Mustoe et al. (9) for the dynamic analysis of deep ocean pipelines.

4.1.2 Geometric Relations

Several variables are defined to simplify the equations in the following sections. Figure 4.4 shows how these variables relate to the belt elements.

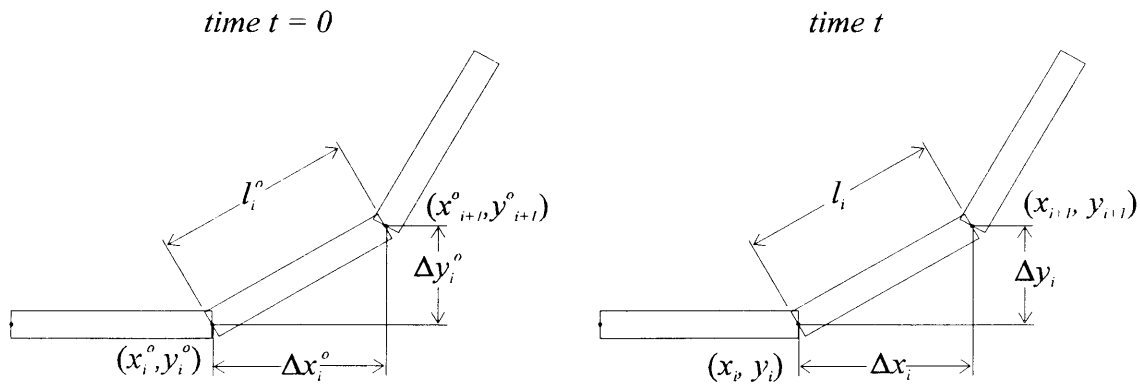


Figure 4.4 Belt variables in initial and current configurations

The nodal displacements in the x and y directions are defined as

$$u_i = x_i - x_i^o \quad (4.12 \text{ a})$$

and

$$v_i = y_i - y_i^o \quad (4.12 \text{ b})$$

Since x_i^o and y_i^o are constants the partial derivatives of equations [4.12] with respect to x_j and y_j respectively are

$$\frac{\partial u_i}{\partial x_j} = \delta_{i,j} \quad (4.13 \text{ a})$$

and

$$\frac{\partial v_i}{\partial y_j} = \delta_{i,j} \quad (4.13 \text{ b})$$

where the Kronecker delta is defined as

$$\delta_{i,j} = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases} \quad (4.14)$$

The horizontal and vertical components of element i are defined as

$$\Delta x_i = x_{i+1} - x_i \quad (4.15 \text{ a})$$

and

$$\Delta y_i = y_{i+1} - y_i \quad (4.15 \text{ b})$$

and the resulting length of element i is

$$l_i = \sqrt{(\Delta x_i)^2 + (\Delta y_i)^2} \quad (4.16)$$

The change in length, β_i , of a belt element is defined as

$$\beta_i = l_i - l_i^0 \quad (4.17)$$

or

$$\beta_i = \sqrt{(\Delta x_i)^2 + (\Delta y_i)^2} - l_i^0 \quad (4.18)$$

The tangent of the angle between elements i-1 and i is found using vector algebra. The vectors through element i and i-1 are

$$\bar{a} = \Delta x_i \hat{i} + \Delta y_i \hat{j} \quad (4.19)$$

$$\bar{b} = \Delta x_{i-1} \hat{i} + \Delta y_{i-1} \hat{j} \quad (4.20)$$

The tangent of the angle between the two vectors is

$$\tan \theta_i = \frac{\bar{a} \times \bar{b}}{\bar{a} \cdot \bar{b}} \quad (4.21)$$

The tangent of the angle between elements i and i-1 is therefore

$$\tan \theta_i = \frac{\Delta x_i \Delta y_{i-1} - \Delta x_{i-1} \Delta y_i}{\Delta x_i \Delta x_{i-1} + \Delta y_i \Delta y_{i-1}} \quad (4.22)$$

4.1.3 Equations of Motion for the Belt Object

The equations of motion are derived using the Lagrange approach. The detailed steps necessary to develop these equations are presented in this section. A dot placed over a symbol represents the derivative with respect to time.

The Lagrange equations, given here without proof, are

$$\frac{\partial}{\partial t} \left(\frac{\partial T}{\partial \dot{q}_i} \right) - \frac{\partial T}{\partial q_i} + \frac{\partial V}{\partial q_i} = Q_i \quad i = 1, 2, \dots, n \quad (4.23)$$

where

q_i = generalized displacements

\dot{q}_i = derivative of the general displacements with respect to time

T = total kinetic energy of the belt

V = total potential energy of the belt

Q = generalized non-conservative forces (damping, contact and friction forces)

n = number of degrees of freedom

There are two degrees of freedom at each belt node. The number of belt elements is equal to the number of belt nodes between the elements. For the belt model

$$q_i = (u_i, v_i) \quad (4.24)$$

and

$$\dot{q}_i = (\dot{u}_i, \dot{v}_i) \quad (4.25)$$

where

u_i = generalized displacement in the x direction for the i^{th} node

v_i = generalized displacement in the y direction for the i^{th} node

equation [4.23] becomes

$$\frac{\partial}{\partial t} \left(\frac{\partial T}{\partial \dot{u}_i} \right) - \frac{\partial T}{\partial u_i} + \frac{\partial V}{\partial u_i} = Q_i^u \quad i = 1, 2, \dots, N \quad (4.26 \text{ a})$$

and

$$\frac{\partial}{\partial t} \left(\frac{\partial T}{\partial \dot{v}_i} \right) - \frac{\partial T}{\partial v_i} + \frac{\partial V}{\partial v_i} = Q_i^v \quad i = 1, 2, \dots, N \quad (4.26 \text{ b})$$

where N = number of belt elements

The total kinetic energy (T) of the belt is

$$T = \frac{1}{2} \sum_{j=1}^N m_j (\dot{u}_j^2 + \dot{v}_j^2) \quad (4.27)$$

Taking the partial derivatives of the kinetic energy with respect to u_i and v_i one finds respectively that

$$\frac{\partial T}{\partial \dot{u}_i} = \sum_{j=1}^N m_j \dot{u}_j \frac{\partial \dot{u}_j}{\partial \dot{u}_i} = m_i \dot{u}_i \quad (4.28 \text{ a})$$

and

$$\frac{\partial T}{\partial \dot{v}_i} = \sum_{j=1}^N m_j \dot{v}_j \frac{\partial \dot{v}_j}{\partial \dot{v}_i} = m_i \dot{v}_i \quad (4.28 \text{ b})$$

The partial derivatives of equations [4.28] with respect to time are

$$\frac{\partial}{\partial t} \left(\frac{\partial T}{\partial \dot{u}_i} \right) = m_i \ddot{u}_i \quad (4.29 \text{ a})$$

and

$$\frac{\partial}{\partial t} \left(\frac{\partial T}{\partial \dot{v}_i} \right) = m_i \ddot{v}_i \quad (4.29 \text{ b})$$

Since the kinetic energy is not a function of u_i or v_i ,

$$\frac{\partial T}{\partial u_i} = \frac{\partial T}{\partial v_i} = 0 \quad (4.30)$$

The total potential energy (V) of the belt is

$$V = \frac{1}{2} \sum_{j=1}^N k_j \beta_j^2 + \frac{1}{2} \sum_{j=1}^N k_j^{\theta} \theta_j^2 + \sum_{j=1}^N m_j y_j g \quad (4.31)$$

Taking the partial derivatives of the potential energy with respect to u_i and v_i one finds that

$$\frac{\partial \mathcal{V}}{\partial u_i} = \sum_{j=1}^N k_j \beta_j \frac{\partial \beta_j}{\partial u_i} + \sum_{j=1}^N k_j^\theta \theta_j \frac{\partial \theta_j}{\partial u_i} \quad (4.32 \text{ a})$$

and

$$\frac{\partial \mathcal{V}}{\partial v_i} = \sum_{j=1}^N k_j \beta_j \frac{\partial \beta_j}{\partial v_i} + \sum_{j=1}^N k_j^\theta \theta_j \frac{\partial \theta_j}{\partial v_i} + \sum_{j=1}^N m_j g \frac{\partial y_j}{\partial v_i} \quad (4.32 \text{ b})$$

The first term in equations [4.32] represents the force components resulting from the axial spring. These are

$$S_{u_i} = \sum_{j=1}^N k_j \beta_j \frac{\partial \beta_j}{\partial u_i} \quad (4.33 \text{ a})$$

and

$$S_{v_i} = \sum_{j=1}^N k_j \beta_j \frac{\partial \beta_j}{\partial v_i} \quad (4.33 \text{ b})$$

The expression for the change in element length (equation [4.18]) when differentiated with respect to u_i and v_i yields

$$\frac{\partial \beta_j}{\partial u_i} = \frac{\Delta x_j}{l_j} \frac{\partial \Delta x_j}{\partial u_i} \quad (4.34 \text{ a})$$

and

$$\frac{\partial \beta_j}{\partial v_i} = \frac{\Delta y_j}{l_j} \frac{\partial \Delta y_j}{\partial v_i} \quad (4.34 \text{ b})$$

Remembering that

$$\Delta x_j = x_{j+1} - x_j \quad (4.35 \text{ a})$$

and

$$\Delta y_j = y_{j+1} - y_j \quad (4.35 \text{ b})$$

then

$$\frac{\partial \Delta x_j}{\partial u_i} = \frac{\partial x_{j+1}}{\partial u_i} - \frac{\partial x_j}{\partial u_i} \quad (4.36 \text{ a})$$

and

$$\frac{\partial \Delta y_j}{\partial v_i} = \frac{\partial y_{j+1}}{\partial v_i} - \frac{\partial y_j}{\partial v_i} \quad (4.36 \text{ b})$$

Substituting equations [4.13] into equations [4.36] yields

$$\frac{\partial \Delta x_j}{\partial u_i} = \delta_{i,j+1} - \delta_{i,j} \quad (4.37 \text{ a})$$

and

$$\frac{\partial \Delta y_j}{\partial v_i} = \delta_{i,j+1} - \delta_{i,j} \quad (4.37 \text{ b})$$

Hence

$$\frac{\partial \beta_j}{\partial u_i} = \frac{\Delta x_j (\delta_{i,j+1} - \delta_{i,j})}{l_j} \quad (4.38 \text{ a})$$

and

$$\frac{\partial \beta_j}{\partial v_i} = \frac{\Delta y_j (\delta_{i,j+1} - \delta_{i,j})}{l_j} \quad (4.38 \text{ b})$$

Thus the first terms become

$$S_{u_i} = \sum_{j=1}^N k_j \beta_j \frac{\Delta x_j (\delta_{i,j+1} - \delta_{i,j})}{l_j} \quad (4.39 \text{ a})$$

and

$$S_{v_i} = \sum_{j=1}^N k_j \beta_j \frac{\Delta y_j (\delta_{i,j+1} - \delta_{i,j})}{l_j} \quad (4.39 \text{ b})$$

Reapplying the Kronecker delta one finds that

$$S_{u_i} = \frac{k_{i-1}\beta_{i-1}\Delta x_{i-1}}{l_{i-1}} - \frac{k_i\beta_i\Delta x_i}{l_i} \quad (4.40 \text{ a})$$

and

$$S_{v_i} = \frac{k_{i-1}\beta_{i-1}\Delta y_{i-1}}{l_{i-1}} - \frac{k_i\beta_i\Delta y_i}{l_i} \quad (4.40 \text{ b})$$

Equation [4.40] represents the forces in the belt resulting from the axial spring. Now one examines the second terms in equations [4.32].

$$B_{u_i} = \sum_{j=1}^N k_j^\theta \theta_j \frac{\partial \theta_j}{\partial u_i} \quad (4.41 \text{ a})$$

and

$$B_{v_i} = \sum_{j=1}^N k_j^\theta \theta_j \frac{\partial \theta_j}{\partial v_i} \quad (4.41 \text{ b})$$

Differentiating equation [4.22] with respect to u_i and v_i respectively

$$(1 + \tan^2 \theta_j) \frac{\partial \theta_j}{\partial u_i} = \frac{\frac{\partial \Delta x_j}{\partial u_i} \Delta y_{j-1} - \frac{\partial \Delta x_{j-1}}{\partial u_i} \Delta y_j}{\Delta x_j \Delta x_{j-1} + \Delta y_j \Delta y_{j-1}} - \frac{(\Delta x_j \Delta y_{j-1} - \Delta x_{j-1} \Delta y_j) \left(\frac{\partial \Delta x_j}{\partial u_i} \Delta x_{j-1} + \frac{\partial \Delta x_{j-1}}{\partial u_i} \Delta x_j \right)}{(\Delta x_j \Delta x_{j-1} + \Delta y_j \Delta y_{j-1})^2} \quad (4.42 \text{ a})$$

and

$$(1 + \tan^2 \theta_j) \frac{\partial \theta_j}{\partial v_i} = \frac{\Delta x_j \frac{\partial \Delta y_{j-1}}{\partial v_i} - \Delta x_{j-1} \frac{\partial \Delta y_j}{\partial v_i}}{\Delta x_j \Delta x_{j-1} + \Delta y_j \Delta y_{j-1}} - \frac{(\Delta x_j \Delta y_{j-1} - \Delta x_{j-1} \Delta y_j) \left(\frac{\partial \Delta y_j}{\partial v_i} \Delta y_{j-1} + \frac{\partial \Delta y_{j-1}}{\partial v_i} \Delta y_j \right)}{(\Delta x_j \Delta x_{j-1} + \Delta y_j \Delta y_{j-1})^2} \quad (4.42 \text{ b})$$

Substituting the expression for $\tan \theta_j$ and simplifying, one finds that

$$\frac{\partial \theta_j}{\partial u_i} = \frac{\frac{\partial \Delta x_j}{\partial u_i} \Delta y_j (\Delta x_{j-1}^2 + \Delta y_{j-1}^2) - \frac{\partial \Delta x_{j-1}}{\partial u_i} \Delta y_{j-1} (\Delta x_j^2 + \Delta y_j^2)}{(\Delta x_{j-1}^2 + \Delta y_{j-1}^2)(\Delta x_j^2 + \Delta y_j^2)} \quad (4.43 \text{ a})$$

and

$$\frac{\partial \theta_j}{\partial v_i} = \frac{-\frac{\partial \Delta y_j}{\partial v_i} \Delta x_j (\Delta x_{j-1}^2 + \Delta y_{j-1}^2) + \frac{\partial \Delta y_{j-1}}{\partial v_i} \Delta x_{j-1} (\Delta x_j^2 + \Delta y_j^2)}{(\Delta x_{j-1}^2 + \Delta y_{j-1}^2)(\Delta x_j^2 + \Delta y_j^2)} \quad (4.43 \text{ b})$$

Substituting the lengths of element j and $j-1$ into the equation yields

$$\frac{\partial \theta_j}{\partial u_i} = \frac{\partial \Delta x_j}{\partial u_i} \frac{\Delta y_j}{l_j^2} - \frac{\partial \Delta x_{j-1}}{\partial u_i} \frac{\Delta y_{j-1}}{l_{j-1}^2} \quad (4.44 \text{ a})$$

and

$$\frac{\partial \theta_j}{\partial v_i} = -\frac{\partial \Delta y_j}{\partial v_i} \frac{\Delta x_j}{l_j^2} + \frac{\partial \Delta y_{j-1}}{\partial v_i} \frac{\Delta x_{j-1}}{l_{j-1}^2} \quad (4.44 \text{ b})$$

Transforming the differential using equation [4.37], one finds that

$$\frac{\partial \theta_j}{\partial u_i} = (\delta_{i,j+1} - \delta_{i,j}) \frac{\Delta y_j}{l_j^2} - (\delta_{i,j} - \delta_{i,j-1}) \frac{\Delta y_{j-1}}{l_{j-1}^2} \quad (4.45 \text{ a})$$

and

$$\frac{\partial \theta_j}{\partial v_i} = -(\delta_{i,j+1} - \delta_{i,j}) \frac{\Delta x_j}{l_j^2} + (\delta_{i,j} - \delta_{i,j-1}) \frac{\Delta x_{j-1}}{l_{j-1}^2} \quad (4.45 \text{ b})$$

Thus the second terms become

$$B_{u_i} = \sum_{j=1}^N \left[(\delta_{i,j+1} - \delta_{i,j}) \frac{k_j^\theta \theta_j \Delta y_j}{l_j^2} - (\delta_{i,j} - \delta_{i,j-1}) \frac{k_j^\theta \theta_j \Delta y_{j-1}}{l_{j-1}^2} \right] \quad (4.46 \text{ a})$$

and

$$B_{v_i} = \sum_{j=1}^N \left[-(\delta_{i,j+1} - \delta_{i,j}) \frac{k_j^\theta \theta_j \Delta x_j}{l_j^2} + (\delta_{i,j} - \delta_{i,j-1}) \frac{k_j^\theta \theta_j \Delta x_{j-1}}{l_{j-1}^2} \right] \quad (4.46 \text{ b})$$

Reapplying the Kronecker delta one finds that

$$B_{v_i} = \frac{k_{i-1}^\theta \theta_{i-1} \Delta y_{i-1}}{l_{i-1}^2} - \frac{k_i^\theta \theta_i \Delta y_i}{l_i^2} - \frac{k_i^\theta \theta_i \Delta y_{i-1}}{l_{i-1}^2} + \frac{k_{i+1}^\theta \theta_{i+1} \Delta y_i}{l_i^2} \quad (4.47 \text{ a})$$

and

$$B_{v_i} = -\frac{k_{i-1}^\theta \theta_{i-1} \Delta x_{i-1}}{l_{i-1}^2} + \frac{k_i^\theta \theta_i \Delta x_i}{l_i^2} + \frac{k_i^\theta \theta_i \Delta x_{i-1}}{l_{i-1}^2} - \frac{k_{i+1}^\theta \theta_{i+1} \Delta x_i}{l_i^2} \quad (4.47 \text{ b})$$

Equation [4.47] represents the bending forces in the belt represented by the rotational springs located between elements.

The final term in equation [4.32 b] represents the forces due to gravity acting on the belt.

$$G_{v_i} = \sum_{j=1}^N m_j g \frac{\partial y_j}{\partial v_i} \quad (4.48)$$

This term expands to

$$G_{v_i} = m_i g \quad (4.49)$$

The Q_i term of equation [4.26] represents damping forces that dissipate energy and external forces that add energy to the system. These types of forces are non-conservative. They are determined by considering the virtual work dW_i done by all of the non-conservative forces due to a virtual generalized displacement, dq_i , of a particular generalized coordinate, q_i . The virtual work dW_i , therefore, is expressed by the equation

$$dW_i = Q_i dq_i \quad (4.50)$$

In the belt model there are three non-conservative forces.

D_i = Forces from axial damping,

C_i = Forces from contact, and

F_i = Forces from friction

The forces resulting from contact and friction are determined later in this chapter and are presented in section 4.4. The axial damping force is calculated as follows.

$$D_i = -\sum_{j=1}^n c_j \dot{\beta}_j \frac{\partial \beta_j}{\partial \alpha_i} \quad (4.51)$$

The latter term is negative since the damping force opposes motion. The components of equation [4.51] in the x and y direction are

$$D_{u_i} = -\sum_{j=1}^N c_j \dot{\beta}_j \frac{\partial \beta_j}{\partial u_i} \quad (4.52 \text{ a})$$

and

$$D_{v_i} = -\sum_{j=1}^N c_j \dot{\beta}_j \frac{\partial \beta_j}{\partial v_i} \quad (4.52 \text{ b})$$

The partial derivative of the change in element length with respect to u_i and v_i from equations [4.38] are

$$\frac{\partial \beta_j}{\partial u_i} = \frac{\Delta x_j (\delta_{i,j+1} - \delta_{i,j})}{l_j} \quad (4.53 \text{ a})$$

and

$$\frac{\partial \beta_j}{\partial v_i} = \frac{\Delta y_j (\delta_{i,j+1} - \delta_{i,j})}{l_j} \quad (4.53 \text{ b})$$

The partial derivative of the change in element length with respect to time is

$$\dot{\beta}_j = \frac{\Delta x_j \Delta \dot{x}_j + \Delta y_j \Delta \dot{y}_j}{l_j} \quad (4.54)$$

Substituting equations [4.53] and [4.54] into equation [4.52] one finds that

$$D_{u_i} = - \sum_{j=1}^N c_j \frac{\Delta x_j \Delta \dot{x}_j + \Delta y_j \Delta \dot{y}_j}{l_j} \frac{\Delta x_j (\delta_{i,j+1} - \delta_{i,j})}{l_j} \quad (4.55 \text{ a})$$

and

$$D_{v_i} = - \sum_{j=1}^N c_j \frac{\Delta x_j \Delta \dot{x}_j + \Delta y_j \Delta \dot{y}_j}{l_j} \frac{\Delta y_j (\delta_{i,j+1} - \delta_{i,j})}{l_j} \quad (4.55 \text{ b})$$

The forces acting on the belt elements due to the axial dashpots are therefore

$$D_{u_i} = \frac{-c_{i-1} \Delta x_{i-1} (\Delta x_{i-1} \Delta \dot{x}_{i-1} + \Delta y_{i-1} \Delta \dot{y}_{i-1})}{l_{i-1}^2} + \frac{c_i \Delta x_i (\Delta x_i \Delta \dot{x}_i + \Delta y_i \Delta \dot{y}_i)}{l_i^2} \quad (4.56 \text{ a})$$

and

$$D_{v_i} = \frac{-c_{i-1} \Delta y_{i-1} (\Delta x_{i-1} \Delta \dot{x}_{i-1} + \Delta y_{i-1} \Delta \dot{y}_{i-1})}{l_{i-1}^2} + \frac{c_i \Delta y_i (\Delta x_i \Delta \dot{x}_i + \Delta y_i \Delta \dot{y}_i)}{l_i^2} \quad (4.56 \text{ b})$$

The resulting equations of motion are

$$m_i \ddot{u}_i = S_{u_i} + B_{u_i} + D_{u_i} + C_{u_i} + F_{u_i} \quad (4.57 \text{ a})$$

and

$$m_i \ddot{v}_i = S_{v_i} + B_{v_i} + G_{v_i} + D_{v_i} + C_{v_i} + F_{v_i} \quad (4.57 \text{ b})$$

where

S_i = Force from axial spring (Equation [4.40])

B_i = Force from rotational spring (Equation [4.47])

G_i = Force from gravity (Equation [4.49])

D_i = Force from axial damper (Equation [4.56])

C_i = Force from external contact (Section 4.4)

F_i = Force from external friction (Section 4.4)

The equations of motion are solved using a central explicit time marching scheme. The accelerations at the belt nodes are calculated from equations [4.57] as

$$\ddot{u}_i^t = \frac{S_{u_i} + B_{u_i} + D_{u_i} + C_{u_i} + F_{u_i}}{m_i} \quad (4.58 \text{ a})$$

and

$$\ddot{v}_i^t = \frac{S_{v_i} + B_{v_i} + G_{v_i} + D_{v_i} + C_{v_i} + F_{v_i}}{m_i} \quad (4.58 \text{ b})$$

The velocities are updated as

$$\dot{u}_i^{t+\Delta t/2} = \dot{u}_i^{t-\Delta t/2} + \ddot{u}_i^t \Delta t \quad (4.59 \text{ a})$$

and

$$\dot{v}_i^{t+\Delta t/2} = \dot{v}_i^{t-\Delta t/2} + \ddot{v}_i^t \Delta t \quad (4.59 \text{ b})$$

and then the nodal displacements are updated as

$$u_i^{t+\Delta t} = u_i^t + \dot{u}_i^{t+\Delta t/2} \Delta t \quad (4.60 \text{ a})$$

and

$$v_i^{t+\Delta t} = v_i^t + \dot{v}_i^{t+\Delta t/2} \Delta t \quad (4.60 \text{ b})$$

4.2 The Pulley, Takeup, Drive and Idler Objects

The pulley, takeup, drive and idler objects are considered jointly in this section because they are very similar. The takeup object is considered to be a gravity takeup. It is modeled as a pulley which is allowed to translate with a large mass attached to it. A drive object is a pulley with an external torque applied to it. Idlers can be modeled as small pulleys.

4.2.1 The Pulley Object

Pulleys rotate about a fixed position on several different types of bearings. There are several forces which act on the pulley object. They include; the weight of the pulley, forces resulting from belt contact, friction forces between the belt and the pulley shell, and pulley bearing friction . The bearing friction force has not been included in the model for the sake of simplicity. Since the pulley is not allowed to translate, only the rotational degree of freedom needs to be considered. The resulting equation of motion for the pulley is

$$I_p \dot{\omega}_p = T_p \quad (4.61)$$

where

$\dot{\omega}_p$ = the angular acceleration of the pulley

I_p = the polar moment of inertia of the pulley

T_p = the total moment generated from friction between the belt and pulley shell

Equation [4.61] is solved using an explicit time marching scheme. The angular velocity is updated as

$$\omega_p^{t+\Delta t/2} = \omega_p^{t-\Delta t/2} + \frac{T_p \Delta t}{I_p} \quad (4.62)$$

Note, that there is no need to update the pulley angle.

4.2.2 The Takeup Object

There are three types of takeup systems used with conveyor belts; gravity, screw, and active winch. This section presents the additional equations of motion required for a gravity takeup that is allowed to translate in the vertical direction (See Figure 4.5). It is assumed that the connection between the pulley and the counterweight is rigid and that

there is no friction in the takeup guides. These assumptions are made for the sake of simplicity.

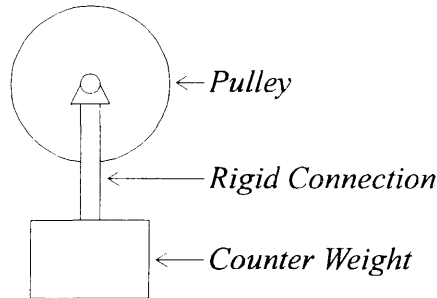


Figure 4.5 Gravity takeup object

Since there is no difference in the polar moment of inertia between the pulley and takeup objects equation [4.61] also applies for the rotational motion of the takeup pulley. The takeup is not allowed to move horizontally (in the x direction) so the only additional equation of motion for the takeup is for its motion in the vertical, y, direction. The equation of motion for the gravity takeup in the y direction is

$$(m_p + m_{TU})\ddot{y} = F_{y_p} \quad (4.63)$$

where

m_p = pulley mass

m_{TU} = takeup mass

F_{y_p} = the total force acting on the takeup in the y direction from gravity, belt contact, and friction between the belt and the pulley

Equation [4.63] is solved using an explicit time marching scheme. The velocity in the y direction is updated as

$$\dot{y}_P^{t+\Delta t/2} = \dot{y}_P^{t-\Delta t/2} + \left(\frac{F_{y_p}^t}{m_p + m_{TU}} \right) \Delta t \quad (4.64)$$

and the location of the center of the pulley is updated as

$$y_P^{t+\Delta t} = y_P^t + \dot{y}_P^{t+\Delta t/2} \Delta t \quad (4.65)$$

4.2.3 The Drive and Brake Objects

There are several different types and numerous configurations of drives, transmissions, clutches, flywheels and brakes. The detailed modeling of these components is beyond the scope of this work. Currently these objects are modeled by applying an external torque to the pulley which may be a function of pulley speed and/or time. The polar moment of inertia of the accessories are added to the polar moment of inertia of the pulley. Since these objects do not translate only the rotational equation of motion needs to be solved. The equation of motion for a drive object is therefore

$$(I_P + I_D) \dot{\omega}_P = T_P + T_D \quad (4.66)$$

where

I_D = the polar moment of inertia of the accessories attached to the pulley

T_D = the torque applied to the pulley from the drive or brake

Equation [4.66] is solved using an explicit time marching scheme. The angular velocity is updated as

$$\omega_P^{t+\Delta t/2} = \omega_P^{t-\Delta t/2} + \frac{(T_P + T_D) \Delta t}{I_P + I_D} \quad (4.67)$$

Note, that there is no need to update the drive angle.

4.2.4 The Idler Object

The idlers can be thought of as small pulleys. They do not translate and only the rotational equation of motion needs to be solved. The equation of motion for an idler is therefore

$$I_I \dot{\omega}_I = T_I \quad (4.68)$$

where

$\dot{\omega}_I$ = the angular acceleration of the pulley

I_I = the polar moment of inertia of the idler

T_I = the total moment generated from friction between the belt and idler shell

Equation [4.68] is solved using an explicit time marching scheme. The angular velocity is updated as

$$\omega_I^{t+\Delta t/2} = \omega_I^{t-\Delta t/2} + \frac{T_I \Delta t}{I_I} \quad (4.69)$$

Note, that there is no need to update the idler angle.

4.3 Contact Checking

As the belt moves various segments of the belt come in contact with both pulleys and idlers. The contact and friction algorithms for pulleys presented in this section also apply to takeup and drive pulleys. Three assumptions have been made with regard to contact checking. First, an idler can only be in contact with one belt element. Second, the belt will never come in contact with itself, and finally, pulleys, the take-up pulley, drive pulleys and idlers will never come in contact with each other.

Friction is required to transmit power from the drive to the belt. It is also required to rotate the pulleys and idlers. Hence, the more accurately that friction between the belt

and pulleys and idlers is modeled, the more accurate the model should predict the dynamics of the conveyor system.

4.3.1 Pulley Contact

The contact between the belt and pulleys is checked at the nodes of the belt elements. A typical contact situation between several belt elements and a pulley is shown in Figure 4.6. Note that the length chosen for the belt elements is small compared to the radius of the pulley involved. In this model a length corresponding to a cord of 30° has been chosen.

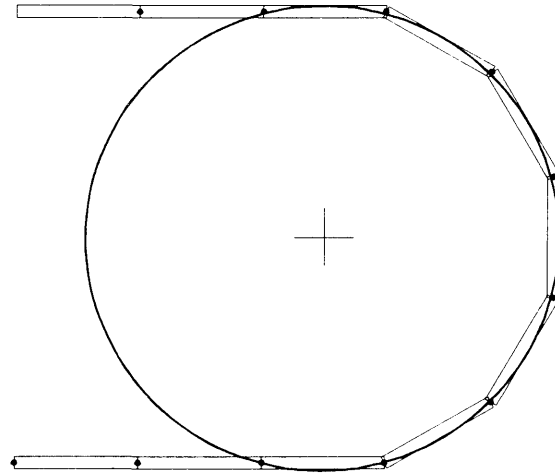


Figure 4.6 Contact between the belt and a pulley

If a belt element ever becomes longer than the pulley diameter the belt would pass through the pulley. The notation used for contact checking between the belt and a pulley is shown in Figure 4.7.

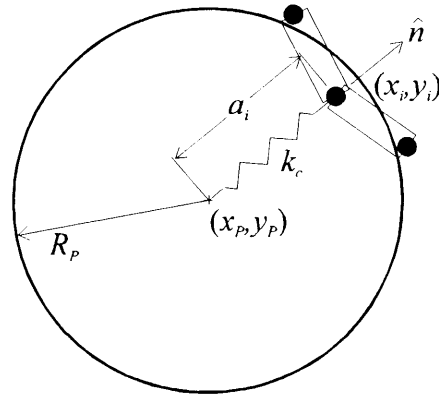


Figure 4.7 Notation for contact between the belt and a pulley

The distance between a belt node and the pulley center is given as:

$$a_i = \sqrt{(x_i - x_p)^2 + (y_i - y_p)^2} \quad (4.70)$$

If the distance, a_i , is less than the pulley radius, R_p , the belt node is considered to be in contact. The contact between the node and the pulley is considered as a penetration problem. The interface is being modeled as a spring of length R_p with a spring constant k_c . The resulting contact force is then:

$$f_c = k_c \frac{R_p - a_i}{R_p} \quad (4.71)$$

Physically the "stiffness" reflects a thinning of the belt and a compression of the pulley material. Although, in theory, the spring constant k_c can be calculated, in the current model k_c is user selected based on observations of the model. Its value is dependent on the maximum allowable penetration, the maximum expected normal velocity and the maximum expected force acting on the node. The stiffness of the contact spring affects the maximum time step of the explicit time marching scheme used. If very small penetration is allowed, then the contact spring is stiff, and the time step is small. Therefore some consideration of accurate contact modeling and computational efficiency needs to be taken when determining the contact spring stiffness value. The contact force between the belt

node and the pulley may be written with respect to each as

$$\bar{C}_i = f_c \hat{n}_i \quad (4.72)$$

and

$$\bar{C}_p = -f_c \hat{n}_i \quad (4.73)$$

where

\bar{C}_i = the contact force acting on belt node i

\bar{C}_p = the contact force acting on the pulley from node i

\hat{n}_i = the unit normal vector at the point where the pulley and belt node i are in contact and is defined as

$$\hat{n}_i = \frac{x_i - x_p}{a_i} \hat{i} + \frac{y_i - y_p}{a_i} \hat{j} \quad (4.74)$$

The belt node force components become

$$C_{u_i} = k_c \left(\frac{R_p - a_i}{R_p} \right) \left(\frac{x_i - x_p}{a_i} \right) \quad (4.75 \text{ a})$$

and

$$C_{v_i} = k_c \left(\frac{R_p - a_i}{R_p} \right) \left(\frac{y_i - y_p}{a_i} \right) \quad (4.75 \text{ b})$$

in the x and y directions respectively.

When all of the belt elements in contact with a pulley have been identified and the contact forces applied, the friction forces between the belt and the pulley are calculated. The method used is based on the friction model presented by Taylor and Preece (10). Their model involves the calculation of a stick force. The stick force is the force required to bring the objects which are in contact to the same tangential velocity by the end of the time step. In Taylor and Preece's derivation it is assumed that (a) the contact forces are the only forces acting upon the objects and (b) that each object is only in contact with one other object at any moment in time. In this thesis the method has been expanded to include external forces acting on the objects and multiple contact. The nomenclature used for the

friction calculation between the belt and a pulley is shown in Figure 4.8.

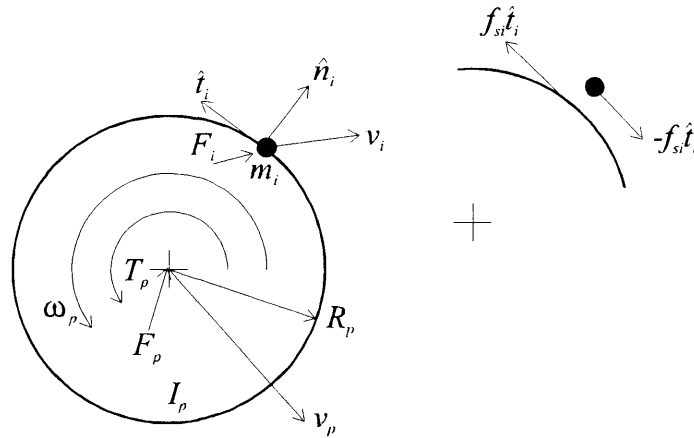


Figure 4.8 Friction between the belt and a pulley

Since at the point of contact the velocity of the node is given by node velocity \bar{v}_i and the velocity of the same point on the pulley is given by

$$\bar{v}_i = \bar{v}_p + R_p \omega_p \hat{t}_i \tag{4.76}$$

the relative velocity \bar{v}_r , between these two points is

$$\bar{v}_r = (\bar{v}_i - \bar{v}_p) - R_p \omega_p \hat{t}_i \tag{4.77}$$

The relative tangential velocity v_{n_i} , at point i is found by taking the dot product of equation [4.77] with \hat{t}_i

$$v_{n_i} = \hat{t}_i \cdot (\bar{v}_i - \bar{v}_p) - R_p \omega_p \tag{4.78}$$

The accelerations of the pulley and nodes in contact with the pulley resulting from the stick force, f_s , and any external forces are as follows

Pulley Translation

$$m_p \dot{\bar{v}}_p = \sum_{j=1}^{NC} f_{s_j} \hat{t}_j + \bar{F}_p \quad (4.79)$$

Pulley Rotation

$$I_p \dot{\omega}_p = R_p \sum_{j=1}^{NC} f_{s_j} + T_p \quad (4.80)$$

Nodal Translation

$$m_i \dot{\bar{v}}_i = -f_{s_i} \hat{t}_i + \bar{F}_i \quad (4.81)$$

where NC is the number of belt nodes in contact with the pulley.

Taking the time derivative of equation [4.78] to find the acceleration one finds that

$$\dot{\hat{t}}_i \cdot (\bar{v}_i - \bar{v}_p) + \hat{t}_i \cdot (\dot{\bar{v}}_i - \dot{\bar{v}}_p) - R_p \dot{\omega}_p = \dot{v}_{n_i} \quad (4.82)$$

Since the derivative, $\dot{\hat{t}}_i$, of the tangential vector is small this term is dropped from the equation. Substituting equations [4.79], [4.80] and [4.81] into equation [4.82] yields

$$\frac{-f_{s_i} + \hat{t}_i \cdot \bar{F}_i}{m_i} - \frac{\hat{t}_i \cdot \sum_{j=1}^{NC} f_{s_j} \hat{t}_j + \hat{t}_i \cdot \bar{F}_p}{m_p} - \frac{R_p^2 \sum_{j=1}^{NC} f_{s_j} + R_p T_p}{I_p} = \dot{v}_{n_i} \quad (4.83)$$

Equation [4.83] is a set of linear equations which need to be solved for f_{s_i} . In most cases a numerical method such as LU decomposition would be used. However because of the large number of time steps involved, it is very computer intensive and time consuming.

An alternative solution has been developed which greatly speeds up the process. By making the assumption that the frictional forces will have little impact on the translation of the pulley (i.e. motion in the x or y direction) equation [4.83] simplifies to:

$$\frac{-f_{s_i} + \hat{t}_i \cdot \bar{F}_i}{m_i} - \frac{R_p^2 \sum_{j=1}^{NC} f_{s_j} + R_p T_p}{I_p} = \dot{v}_{n_i} \quad (4.84)$$

This assumption will only introduce error for the take-up pulley because the other pulleys in the system do not translate. The stick force f_{s_i} , written in terms of $\sum_{j=1}^{NC} f_{s_j}$ becomes

$$f_{s_i} = -\dot{v}_{n_i} m_i + \hat{t}_i \cdot \bar{F}_i - \frac{\left(R_p^2 \sum_{j=1}^{NC} f_{s_j} + R_p T_p \right) m_i}{I_p} \quad (4.85)$$

Expression [4.85] applies to each node in contact with the pulley. Solving for f_{s_i} still requires iteration. If one considers all of the nodes in contact at once as a group, then equation [4.84] can be written as,

$$\frac{-\sum_{j=1}^{NC} f_{s_j} + \sum_{j=1}^{NC} \hat{t}_j \cdot \bar{F}_j}{\sum_{j=1}^{NC} m_j} - \frac{R_p^2 \sum_{j=1}^{NC} f_{s_j} + R_p T_p}{I_p} = \frac{\sum_{j=1}^{NC} \dot{v}_{n_j} m_j}{\sum_{j=1}^{NC} m_j} \quad (4.86)$$

Solving equation [4.86] for $\sum_{j=1}^{NC} f_{s_j}$ one finds that

$$\sum_{j=1}^{NC} f_{s_j} = \frac{-\sum_{j=1}^{NC} \dot{v}_{n_j} m_j + \sum_{j=1}^{NC} \hat{t}_j \cdot \bar{F}_j - \frac{R_p T_p \sum_{j=1}^{NC} m_j}{I_p}}{1 + \frac{R_p^2 \sum_{j=1}^{NC} m_j}{I_p}} \quad (4.87)$$

Multiplying both sides of equation [4.87] by R_p^2 yields

$$R_p^2 \sum_{j=1}^{NC} f_{s_j} = \frac{R_p^2 \left(-\sum_{j=1}^{NC} \dot{v}_{n_j} m_j + \sum_{j=1}^{NC} \hat{t}_j \cdot \bar{F}_j \right) - \frac{R_p^3 T_p \sum_{j=1}^{NC} m_j}{I_p}}{1 + \frac{R_p^2 \sum_{j=1}^{NC} m_j}{I_p}} \quad (4.88)$$

Adding $R_p T_p$ to both sides one obtains

$$R_p^2 \sum_{j=1}^{NC} f_{s_j} + R_p T_p = \frac{R_p^2 \left(-\sum_{j=1}^{NC} \dot{v}_{n_j} m_j + \sum_{j=1}^{NC} \hat{t}_j \cdot \bar{F}_j \right) - \frac{R_p^3 T_p \sum_{j=1}^{NC} m_j}{I_p} + R_p T_p + \frac{R_p^3 T_p \sum_{j=1}^{NC} m_j}{I_p}}{1 + \frac{R_p^2 \sum_{j=1}^{NC} m_j}{I_p}} \quad (4.89)$$

Simplifying and dividing both sides of equation [4.89] by I_p gives

$$\frac{R_p^2 \sum_{j=1}^{NC} f_{s_j} + R_p T_p}{I_p} = \frac{R_p^2 \left(-\sum_{j=1}^{NC} \dot{v}_{n_j} m_j + \sum_{j=1}^{NC} \hat{t}_j \cdot \bar{F}_j \right) + R_p T_p}{I_p + R_p^2 \sum_{j=1}^{NC} m_j} \quad (4.90)$$

Equation [4.90] is now in a form which can be substituted into equation [4.85]. The resulting equation for f_{s_i} is

$$f_{s_i} = -\dot{v}_{n_i} m_i + \hat{t}_i \cdot \bar{F}_i - \left[\frac{R_p^2 \left(-\sum_{j=1}^{NC} \dot{v}_{n_j} m_j + \sum_{j=1}^{NC} \hat{t}_j \cdot \bar{F}_j \right) + R_p T_p}{I_p + R_p^2 \sum_{j=1}^{NC} m_j} \right] m_i \quad (4.91)$$

The values for f_{s_i} can now be solved directly without iteration. Since the objects will be accelerated in one time step the acceleration \dot{v}_{n_i} , is replaced by

$$\dot{v}_{n_i} = \frac{-v_{n_i}}{\Delta t} \quad (4.92)$$

The values of f_{s_i} determined using equation [4.91] must now be compared to the maximum allowable static friction forces at each node. The static friction force is calculated as

$$f_{s_i} = \mu_s f_{n_i} \quad (4.93)$$

where μ_s is the coefficient of static friction and f_{n_i} is the normal force acting on node i . The normal force is calculated during the contact checking step. If the stick friction force is greater than the static friction force then the dynamic friction force

$$f_{d_i} = \mu_d f_{n_i} \quad (4.94)$$

is applied to the node and the pulley. This alters the force balance. The known forces on all slipping nodes are now applied. These nodes are no longer considered to be in the quantity $\sum_{j=1}^{NC} f_{s_j}$ and are removed as appropriate. The forces on the remaining nodes are now recalculated accordingly. The algorithm for calculating the friction forces between the belt and a pulley is shown in Figure 4.9.

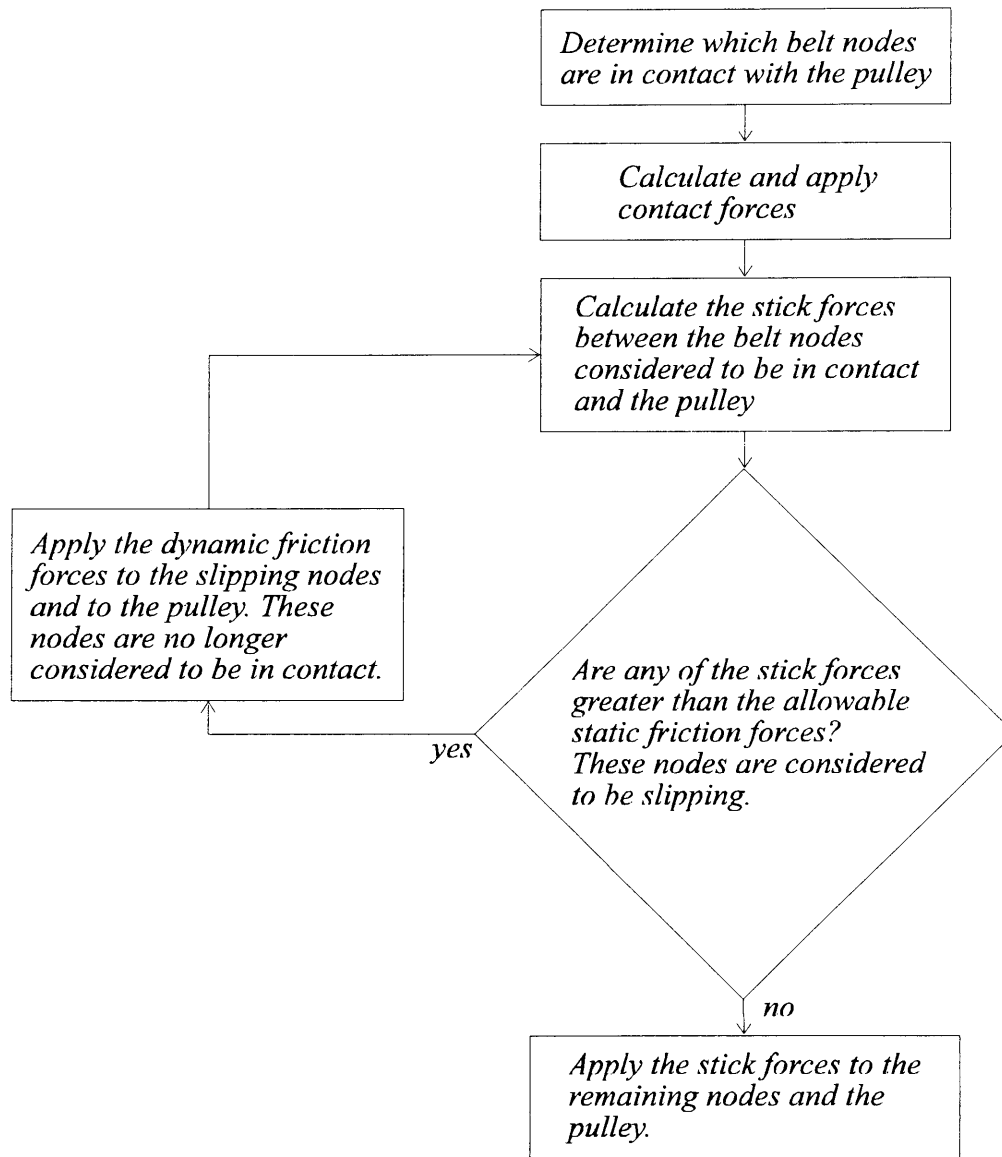


Figure 4.9 Belt/Pulley friction algorithm

4.3.2 Idler Contact

The contact between the belt and idlers uses a different approach. Contact is checked with the element centerline and not at the nodes. If the element nodes were used to check contact with the idlers the belt element length would have to be shorter than the idler diameters. This would significantly shorten the belt element lengths and greatly decrease the computational efficiency of the program. The notation used for contact checking between the belt and an idler is shown in Figure 4.10.

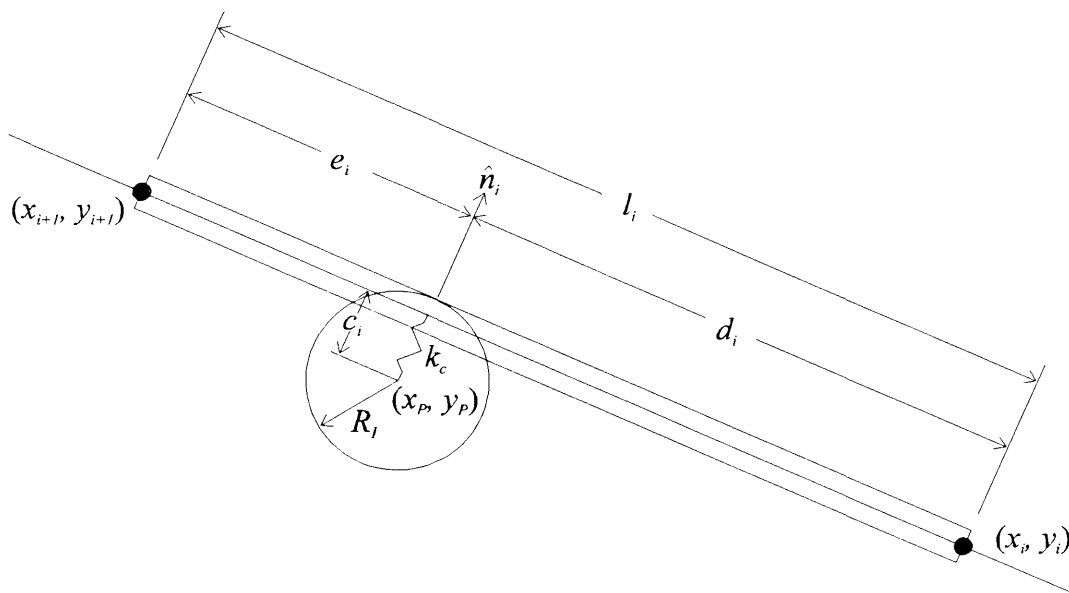


Figure 4.10 Contact between an idler and a belt element

The tangential distance between the belt element and the idler center is found using vector algebra

$$c_i = \frac{|\bar{a}_i \times \bar{b}_i|}{|\bar{a}_i|} \quad (4.95)$$

where

$$\begin{aligned} \bar{a}_i &= (x_{i+1} - x_i)\hat{i} + (y_{i+1} - y_i)\hat{j} \\ \bar{b}_i &= (x_p - x_i)\hat{i} + (y_p - y_i)\hat{j} \end{aligned}$$

If the distance c_i is less than the idler radius, R_I , then the belt element may be in contact. An additional test must be performed to determine if the point of tangency falls within the element length. The distance between end points and the point of tangency must be less than the element length for both end points. The distances between the endpoints and the point of tangency are found using the following equations.

$$d_i = \sqrt{|\bar{b}_i|^2 - c_i^2} \quad (4.96)$$

$$e_i = \sqrt{|\bar{b}_{i+1}|^2 - c_i^2} \quad (4.97)$$

If there is contact between the belt element and the idler, a spring is inserted between the point of tangency on the belt element and the idler center. The contact force is then calculated as

$$f_c = k_c \frac{R_I - a_i}{R_I} \quad (4.98)$$

The contact force is applied to the belt and the idler in the normal direction. The normal direction is more complicated to calculate than with contact with a pulley. The difficulty is in determining which side of the belt element the idler is in contact. Fortunately the sign of the cross product used to determine the tangential distance between the belt element and the idler center indicates which side the idler is on. Upon using this logic the normal direction is

$$\hat{n}_i = \frac{\bar{a} \times \bar{b}}{|\bar{a} \times \bar{b}|} [(y_{i+1} - y_i)\hat{i} + (x_{i+1} - x_i)\hat{j}] \quad (4.99)$$

The contact force cannot be simply applied to the belt element. The force must be divided between the two nodes at the ends of the element. The ratio of the division is determined knowing the distance from the endpoint to the point of tangency. The force applied to the belt element nodes are

$$\bar{C}_i = \frac{f_c d_i}{l_i} \hat{n}_i \quad (4.100)$$

and

$$\bar{C}_{i+1} = \frac{f_c e_i}{l_i} \hat{n}_i \quad (4.101)$$

The force applied to the idler is

$$\bar{C}_I = -f_c \hat{n}_i \quad (4.102)$$

The friction between a belt element and an idler is calculated using the same logic as was used in calculating the friction between the belt and a pulley. Since the velocity of the belt at the point of contact is

$$\bar{v}_{Bc} = (1 - \gamma_i) \bar{v}_i + \gamma_i \bar{v}_{i+1} \quad (4.103)$$

where

$$\gamma_i = \frac{d_i}{l_i}$$

and the velocity of the idler at the point of contact is

$$\bar{v}_{Ic} = \omega_I R_I \hat{t} \quad (4.104)$$

the relative velocity, \bar{v}_r , between these two points is

$$\bar{v}_r = \bar{v}_{Bc} - \bar{v}_{Ic} \quad (4.105)$$

or

$$\bar{v}_r = (1 - \gamma_i) \bar{v}_i + \gamma_i \bar{v}_{i+1} - \omega_I R_I \hat{t} \quad (4.106)$$

The relative tangential velocity, v_{rt} , at the point of contact is found by taking the dot product of equation [4.106] with \hat{t}

$$\mathbf{v}_n = \hat{t} \cdot [(1 - \gamma_i)\bar{\mathbf{v}}_i + \gamma_i\bar{\mathbf{v}}_{i+1}] - \omega_I R_I \quad (4.107)$$

Taking the derivative of equation [4.107] with respect to time to find the required acceleration one finds that

$$\begin{aligned} \dot{\mathbf{v}}_n = \dot{\hat{t}} \cdot [(1 - \gamma_i)\bar{\mathbf{v}}_i + \gamma_i\bar{\mathbf{v}}_{i+1}] \\ + \hat{t} \cdot [(1 - \gamma_i)\dot{\bar{\mathbf{v}}}_i + \gamma_i\dot{\bar{\mathbf{v}}}_{i+1} - \dot{\gamma}_i(\bar{\mathbf{v}}_i - \bar{\mathbf{v}}_{i+1})] - \dot{\omega}_I R_I \end{aligned} \quad (4.108)$$

The derivative of the tangential vector, $\dot{\hat{t}}$, is small and therefore dropped from the equation. By assuming that the belt element is rigid while calculating the friction forces the derivative of the contact length ratio, $\dot{\gamma}_i$, is also equal to zero. Equation [4.108] then reduces to

$$\dot{\mathbf{v}}_n = \hat{t} \cdot [(1 - \gamma_i)\dot{\bar{\mathbf{v}}}_i + \gamma_i\dot{\bar{\mathbf{v}}}_{i+1}] - \dot{\omega}_I R_I \quad (4.109)$$

The acceleration of the idler and belt nodes of the element in contact with the idler resulting from the stick force, f_s , and any external forces are as follows

Idler Rotation

$$I_I \dot{\omega}_I = f_s R_I + T_I \quad (4.110)$$

Translation of Belt Element Node i

$$m_i \dot{\bar{\mathbf{v}}}_i = -f_s (1 - \gamma_i) \hat{t} + \bar{\mathbf{F}}_i \quad (4.111)$$

Translation of Belt Element Node i+1

$$m_{i+1} \dot{\bar{\mathbf{v}}}_{i+1} = -f_s \gamma_i \hat{t} + \bar{\mathbf{F}}_{i+1} \quad (4.112)$$

The friction force has been divided between the two belt nodes based on a linear relationship of the distance from the particular node to the point of contact. Substituting equation [4.110], [4.111], and [4.112] into equation [4.109] one obtains

$$\dot{v}_n = \hat{t} \cdot \left[(1 - \gamma_i) \frac{-f_s(1 - \gamma_i)\hat{t} + \bar{F}_i}{m_i} + \gamma_i \frac{-f_s\gamma_i\hat{t} + \bar{F}_{i+1}}{m_{i+1}} \right] - \frac{R_I^2 f_s + T_I R_I}{I_I} \quad (4.113)$$

Solving equation [4.113] for f_s one obtains

$$f_s = \frac{\hat{t} \cdot m_{i+1} I_I (1 - \gamma_i) \bar{F}_i + \hat{t} \cdot m_i I_I \gamma_i \bar{F}_{i+1} - m_i m_{i+1} T_I R_I}{m_{i+1} I_I (1 - \gamma_i)^2 + m_i I_I \gamma_i^2 - m_i m_{i+1} R_I^2} - \dot{v}_n m_i m_{i+1} I_I \quad (4.114)$$

Since the objects will be accelerated in one time step the acceleration \dot{v}_n , is replaced by

$$\dot{v}_n = \frac{-v_n}{\Delta t} \quad (4.115)$$

The values of f_s determined using equation [4.115] must now be compared to the maximum allowable static friction force at the point of contact. The static friction force is calculated as

$$f_s = \mu_s f_n \quad (4.116)$$

where μ_s is the coefficient of static friction and f_n is the normal force acting on node i . The normal force is calculated during the contact checking step. If the stick friction force is greater than the static friction force then the dynamic friction force

$$f_d = \mu_d f_n \quad (4.117)$$

is applied to the belt element nodes and the idler.

Chapter 5.

OBJECT ORIENTED PROGRAMMING CONSIDERATIONS

The object based conveyor model presented in chapter 4 is designed to make use of the power of object-oriented programming (OOP). This section introduces object-oriented programming, describes the objects or classes used by the object model, and finally shows how the model can be expanded to include new features with minimal effort by using object-oriented techniques.

5.1 Introduction to Object Oriented Programming

Object-oriented programming is a style of programming that offers several advantages over traditional styles. These advantages include better code maintainability, better data protection, and ease of expandability.

The system being modeled is considered as a collection of objects or classes. These classes are similar to structures which contain related data in traditional programming. Classes take structures one step further by including functions which control the data and work with the data in the class.

The best way to appreciate the power of this method is to see how the programming structure actually conforms to the physical system which is being modeled.

5.2 The Conveyor Class

When designing the computer program the programmer must first decide what is being modeled. In our case we are modeling a conveyor. This will therefore be our

controlling class or the class the user interface will deal with. Figure 5.1 shows the structure of the conveyor class with an example conveyor input.

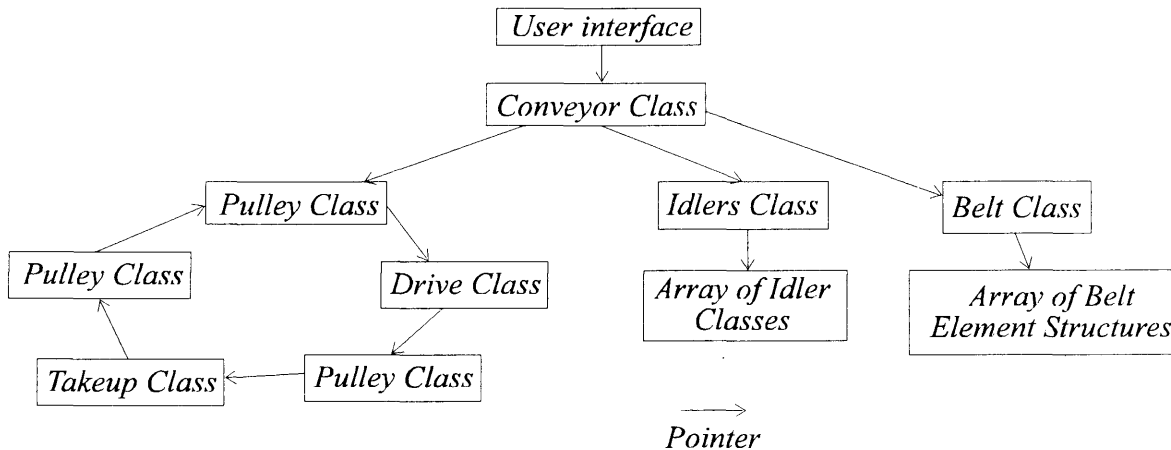


Figure 5.1 Structure of conveyor class

The first step in defining a conveyor is defining the parameters of the conveyor geometry. This is done by adding pulleys, a takeup and a drive object to the conveyor. The classes of these objects are defined in section 5.3. All the user interface must know is that it is adding one of these objects to the conveyor system. The conveyor class contains a circular dynamically linked list of the pulleys, takeup and drive(s). The functions of the conveyor class take care of adding the objects to the list. The user interface does not need to know how the conveyor class takes care of these objects, it just needs to know that a pulley, takeup or drive object has been added to the list. A circular dynamically linked list has been chosen to maintain the pulley, takeup, and drive objects for two reasons. First the conveyor geometry may change requiring a link (object) within the list to be added or subtracted. The second reason for using a dynamically linked list is that at compile time the program does not know how large the conveyor system will be and more importantly the order in which conveyor objects will be arranged in the list. Each link (object) of the list contains a pointer to the following link. At compile time it is not known whether the following link will be a pulley, takeup, drive or whatever may be defined in the future. The compiler only knows that the next link will be some object derived from the pulley base

class.

As the geometry is being input the conveyor class locates the idler objects and the initial location of the belt object. These objects are defined in sections 5.4 and 5.5 respectively. Again the user interface does not need to know how the conveyor object initializes them and it also does not need to tell conveyor class to initialize them. The conveyor class knows that if the geometry of the conveyor class changes, the belt and idlers may have to be reset. The conveyor class does allow the user interface to change the properties of the belt and idlers. The conveyor class contains a pointer to a belt object, which controls the belt, and a pointer to an idlers object, which controls the idlers.

Now that the conveyor geometry has been input and the conveyor has pulleys, a takeup and a drive or drives, and a belt and some idlers, the class needs to be able to do something with all these objects. To check the input visually, the conveyor class has a function to draw the conveyor to a specified output. All the user interface must do is tell the conveyor class to draw itself. The conveyor class itself does not know how to draw all of the objects but it tells the objects in the class to draw themselves. The belt is told to draw itself, and the idlers, and all the components which make up the conveyor.

There are several other functions which the user interface can tell the conveyor to do. They include stabilizing the conveyor, as described in section 6.2, moving the conveyor one time step, and zeroing the kinetic energy of the system. The conveyor class includes functions for the output of data to files and for saving the input conveyor to a file. All of these functions behave similarly to the draw function. The conveyor class does not know how to perform all the operations, but instead tells the objects making up the class to perform the operation.

The conveyor class contains several other functions and variables which control the inner workings of the class. These are not discussed because they deal with how object code is programmed and note the finer details of object-oriented programming.

5.3 The Pulley, Takeup, and Drive Classes

The conveyor belt geometry is defined by the locations of the pulleys. The takeup and drive pulleys are derived from a base pulley class and are discussed later in this

section.

The pulley class contains the data and functions required for a pulley. The data includes the x and y location of the pulley center, the pulley radius, mass, and rotational inertia. Several functions also make up the pulley class. Many of the functions of the conveyor class are also found in the pulley class. An example is when the user interface asks the conveyor class to draw itself, the conveyor class then asks each pulley to draw themselves. Therefore the pulley class must have a function which draws the pulley to a selected output device. The pulley class also has functions for moving the pulley, zeroing the kinetic energy, and for file output. Like the conveyor class, there are several functions and data members of the pulley class which have not been discussed.

The introduction of the takeup class shows where the power of object-oriented programming for code expandability comes in. A gravity takeup is really a standard pulley with a large mass attached to it. In traditional programming methods the code written for the pulley would simply be copied and then modified to model a takeup. In object-oriented programming all of the data and functions of the pulley are inherited into the takeup object. This means that the computer code for the takeup object can use all the functions and data available for the pulley. However, the takeup object requires some additional data and some of the pulley functions may not be completely correct for the takeup. The data can simply be added to the takeup class and the required function rewritten. The additional data member required for the takeup is the mass of the counterweight attached to the pulley. The only pulley function that must be redefined for the takeup is the function which moves the takeup because the takeup is allowed to translate. Therefore, to add the takeup model to the program the only code required is the addition of the takeup mass to the takeup class and the redefinition of the function which moves the takeup. The definitions of the pulley and takeup classes are shown in Figure 5.2 and Figure 5.3 to show how inheritance is implemented in C++.

```

class AHPulley {
protected: double x, y, vx, vy, ax, ay, theta, omega, alpha;
           double forcex, forcey, torque, mass, inertia, radius;
           double mustatic, mudynamic, distance, theta_in, theta_out, spacing;
           int type, start, rotation;
public:    AHPulley *next;
          AHPulley();
// functions which do things
          AHPulley( PulleyTB *tb);
          void Initialize();
          void Draw(HDC hDC);
          virtual void Move(double dt);
// functions which set things
          void Start(int i);
          void ForceX ( double a);
          void ForceY ( double a);
          void Torque ( double a);
          virtual void DriveForces();
          virtual void ZeroForces();
          virtual void ZeroKineticEnergy();
// functions returning things
          double Length();
          double X();
          double Y();
          virtual double X0();
          virtual double Y0();
          double Vx();
          double Vy();
          double Ax();
          double Ay();
          double Radius();
          double Inertia();
          double Omega();
          double Torque();
          int Rotation();
          double Mustatic();
          double Mudynamic();
          double Theta_in();
          double Theta_out();
          double Distance();
          double Spacing();
          int Start();
          int NumberOfIdlers();
          virtual double KineticEnergy();
          virtual double PotentialEnergy();
};

```

Figure 5.2 Pulley class definition

```
class AHTakeup : public AHPulley {
protected:
    double tumass, x0, y0;
public:
    AHTakeup( TakeupTB *tb);
    AHTakeup();
    virtual void ZeroForces();
    virtual void Move(double dt);
    virtual double KineticEnergy();
    virtual double PotentialEnergy();
    virtual double X0();
    virtual double Y0();
    virtual void ZeroKineticEnergy();
};
```

Figure 5.3 Takeup class definition

A simple drive object has been defined for the object code. Like the takeup, the drive class is derived from the pulley class. All of the data and functions of the pulley class are inherited by the drive class. Additional data must be added to the drive class to define the speed torque curve of the drive. The function which moves the takeup must also be changed to account for the speed torque curve of the drive. This is a very simple drive model, more advanced drives are discussed in the section 7.2.

Once a base class has been written it is very easy to expand the computer code to include objects derived from this base class. This makes object oriented code very easy to expand and also easy to maintain. As an example, suppose that at some point the programmer wants to change the way that the pulleys, takeup, and drives are drawn. Using traditional programming techniques all of the code that had been copied for the different objects would have to be updated. This often leads to programming errors and difficult program maintenance. With object oriented code, only the function in the base object has to be changed. In the conveyor model only the drawing function in the pulley class has to be modified, for example.

5.4 The Idlers Class

The idlers class controls all of the idlers in the conveyor system. Each idler is further controlled by an idler class. (Notice the distinction between the singular and plural use of the word idler.) The idler class contains the x and y location, radius and rotational inertia of the idler. The idler class also has functions to draw and move the idler. A dynamically allocated array of idler objects is used by the idlers class to hold the idlers. The functions of the idlers class take care of telling each idler what to do. Returning to the example of when the user interface instructs the conveyor to draw itself, first the conveyor class instructs all of the pulleys, and therefore, the takeup and drive objects, to draw themselves and then it instructs the idlers object to draw itself. The idlers object takes the command one step further by looping through the array of idler objects and instructing each of them to draw themselves. A similar process occurs when the conveyor is asked to move itself and perform other functions.

5.5 The Belt Class

The belt class is the most complex used in the conveyor model. The data required for each belt element are placed in a structure. The belt element structures are contained in a dynamically allocated array. The belt elements have not been modeled as an individual class because the functions for controlling the belt can be handled more efficiently by letting the belt class handle them. The functions for drawing and moving the belt are like those of the pulleys and the idlers. The complexity of the belt class comes from a break from object-oriented programming structure. The contact and friction checking functions have been included as part of the belt class. In true object-oriented code these functions would be written as friend functions or functions which have access to the data of the belt class, the idlers class, and the pulley class. The functions have been included in the belt class because they can get faster access (and hence, CPU time) to the data of the belt class than if they were written as friend functions. This is important in the modeling scheme used where speed of program execution is important.

5.6 Using Object-Oriented Techniques to Expand the Model

Good examples which illustrate the idea of inheritance are the takeup and drive objects which are derived from the pulley class. Minimal effort is required to derive these additional objects. There is no need to redefine the contact algorithms or any other part of the conveyor code when objects are added. The only additional work required are the changes to the user interface that may be necessary.

In the same manner that the simple drive was derived from the pulley class it is possible to derive a new object, like a more complicated drive, from the simple drive class. The new class inherence all of the data and functions from the simple drive class. All of the functions which had previously been written and tested for the drive class are available to the new drive class. This allows the programmer to concentrate on the new functions and functions which must be updated for the drive class.

5.7 Conclusions

This chapter has introduced the way object-oriented programming techniques are used to program the object model. The way the object model has been set up lends itself intuitively to object-oriented programming. With object-oriented programming computer code is not rewritten or copied when defining similar objects. This leaves the programmer free to concentrate on the differences between the objects which are what are important.

Chapter 6.

TESTING THE OBJECT MODEL

In this section a simple model is used to demonstrate the operation and check the results of the object model. The system shown in Figure 6.1 is modeled.

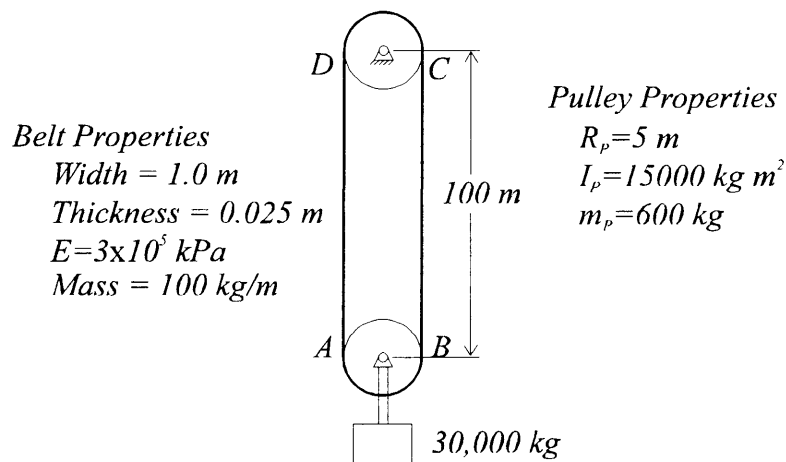


Figure 6.1 System modeled using object model

A contact stiffness of $k_c = 10^{12}$ N/m has been selected. This value was chosen to prevent excessive penetration of the belt into the pulley. The time step is limited by the central explicit time stepping scheme used. The time step is calculated as

$$\Delta t_{\max} = \frac{2}{\sqrt{k/m}} \quad (6.1)$$

where Δt_{\max} is the maximum allowable time step. In this example the time step used is half the maximum allowable time step and is equal to 0.00036 seconds. The belt is divided into 89 elements, each of 2.6 m length.

6.1 Initial Conditions

The object model selects a belt element size no larger than a 30° arc of the smallest pulley. All of the belt elements have the same unstretched length. The initialization process of the model locates the belt elements in integer numbers in each section of the belt profile. The straight distance between each pulley and the arc of the belt contact around each pulley are considered to be separate sections of the belt profile. This process results in an initial tension distribution in the belt as shown in Figure 6.2.

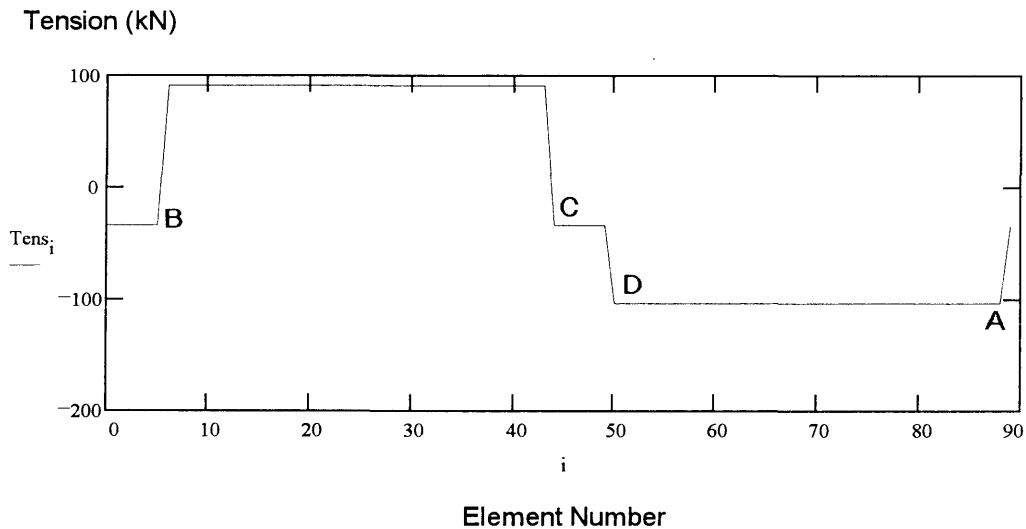


Figure 6.2 Initial tension distribution in the belt

The discontinuous steps in tension between the different sections of the belt are a result of the initial belt elements' locations. Ideally the initial belt tensions would reflect the static equilibrium belt tensions, but this requires a more involved and complicated belt locating algorithm. As the size of the system being modeled becomes larger, these tension steps become smaller. Using a smaller element size also reduces these discontinuities in tension.

6.2 Stabilization

The next step in the modeling process is to bring the system to a static equilibrium position. The model can simply be allowed to run dynamically and let the axial damping dashpot gradually remove the system kinetic energy. This takes a considerable amount of computational time. To speed up this process, kinetic energy damping is introduced. The kinetic energy of the system is monitored. When the kinetic energy of the system attains a maximum, the kinetic energy of the system is set to zero. This is performed by setting the velocities of all the objects to zero. This procedure of zeroing the kinetic energy is repeated until the total energy of the system is constant. The total kinetic and potential energies of the example system (Figure 6.1) are shown versus time during the stabilization process in Figure 6.3. It should be noted that time is a non-physical iteration (or relaxation) parameter during the stabilization process. This process takes approximately 1/4 hour of computer time on a 486 DX 66 MHz personal computer for the example system.

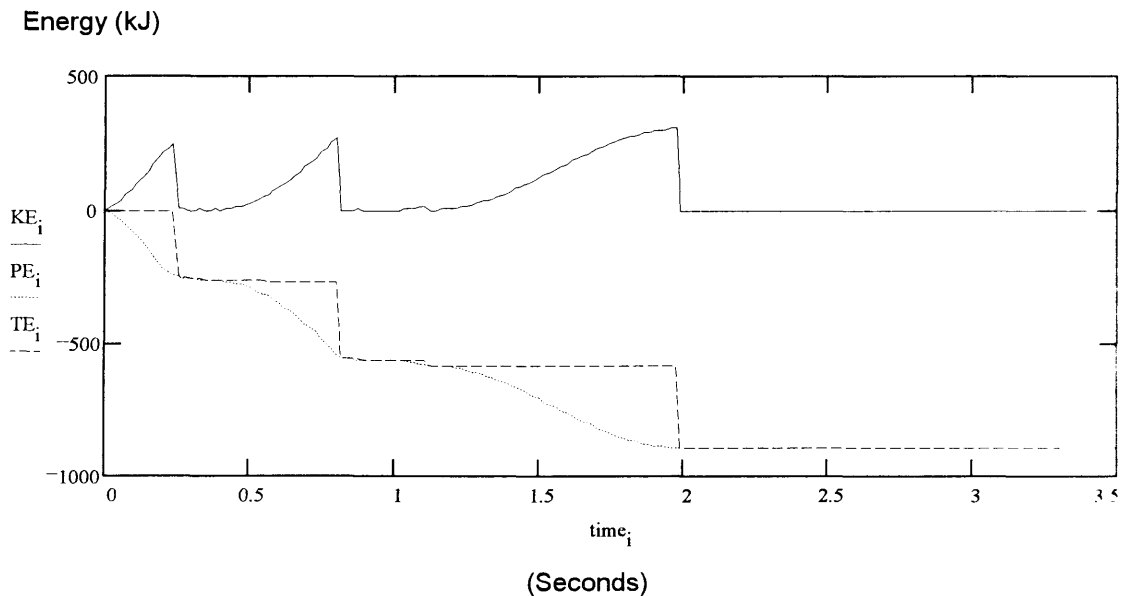


Figure 6.3 Total kinetic and potential energy during the stabilization process

During the stabilization process, the friction between the belt and the pulleys is not modeled. The rationale for this are explained in section 6.3. The tensions in the belt at the end of the stabilization process are shown in Figure 6.4.

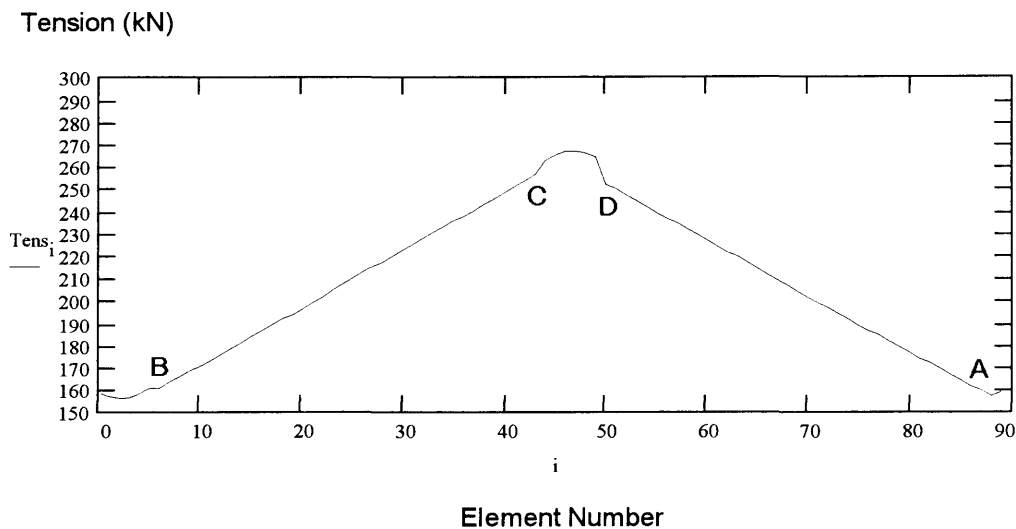


Figure 6.4 Belt tensions after stabilization with zero contact friction

6.3 Belt Tensions at Static Equilibrium

Once the system being modeled has been stabilized, the tensions in the belt represent the static equilibrium conditions. During the stabilization process the friction between the belt and the pulley is set to zero. To ensure that adding friction after the stabilization process does not change the tension profile the model is run for another 2000 time steps. The resulting belt tensions are shown in Figure 6.5.

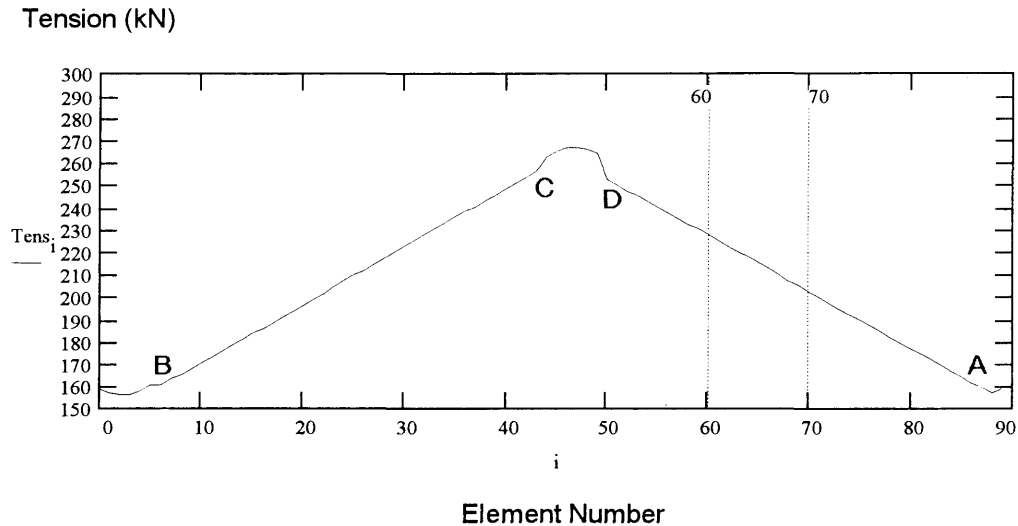


Figure 6.5 Belt tensions after 2000 time steps
with contact friction

Calculations by hand show that the belt tensions at points (A) and (B) are 158 kN and the tensions at points (C) and (D) are 256 kN. The belt tensions in the numerical model agree with these values. The displacement of the takeup, the lower pulley, is calculated by hand to be -3.14 m. The model predicts that the takeup has moved -3.39 m.

Inspection of the tension graphs (Figure 6.4 and 6.5) show a slight step in the tensions at the locations where the belt first comes in contact with the pulley and again when it loses contact with the pulley. These steps in the tension are a result of using the nodes of the belt elements to determine contact, which leads to small errors in modeling contact in these regions. The contact forces (See section 4.4.2) are applied in an outward normal direction to the belt nodes. This results in adding a small amount of tension to the first and last elements coming into contact with the pulley. These steps can be made smaller by decreasing the belt element length. There is a possibility of eliminating these tension steps by using a contact checking scheme similar to the one used between the idlers and the belt. However, this scheme introduces major difficulties when attempting to model the friction between the belt and the pulley.

During the stabilization process the friction between the belt and the pulleys is set to zero. If the friction is not set to zero, the resulting tensions in the belt are shown in Figure 6.6 at the end of the stabilization process.

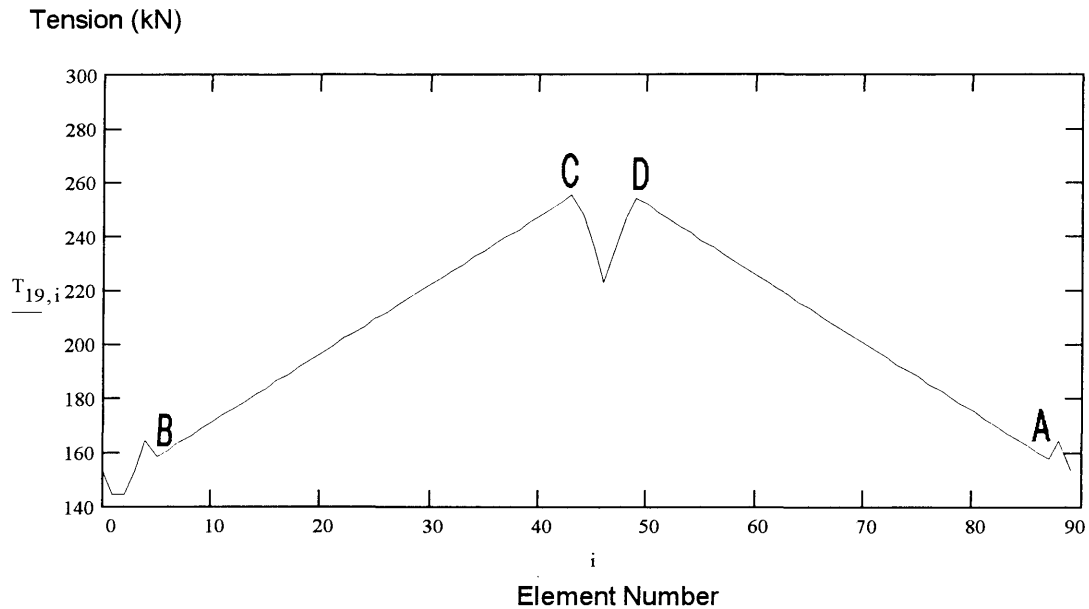


Figure 6.6 Belt tensions after stabilization with friction

The decrease in tension around the pulley face is a result of the lower tensions that were "locked" in at an early time within the system. To clarify this explanation, a 3-D graph of the tensions leading up to the stabilized system are shown in Figure 6.7.

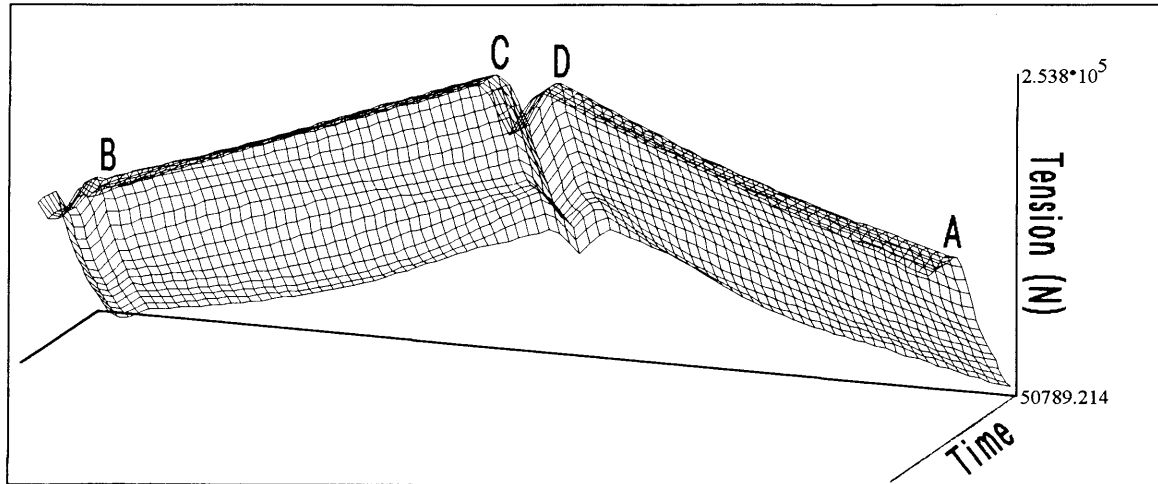


Figure 6.7 Evolution of belt tensions during stabilization process with friction

It is seen that the tensions of the whole system are rising as the takeup increases the tensions in the system. The friction between the belt and the pulley removes some of the tension between the points where the belt comes into contact and the center of contact between the belt and the pulley. This change in tension is physically correct and is accurately modeled. However, in actual conveyor systems, the takeup tension is not applied in this fashion and the tension around a pulley should be nearly constant in static equilibrium. Note, that this is not the case for a pulley which has an external torque applied to it.

6.4 The Dynamic Response of the System

To observe the dynamic characteristics of the example system, two tests have been run. These tests are designed so that their results can be compared with analytical solutions.

6.4.1 Dynamic Response of Increasing Takeup Mass

In this test the takeup mass is instantaneously increased by 10% after the system has been stabilized. This causes the system to oscillate around a new static equilibrium position at the natural frequency of the system. Both the static equilibrium position and the natural frequency of the system can be calculated by hand. The dynamic tensions in the system are shown in Figure 6.8.

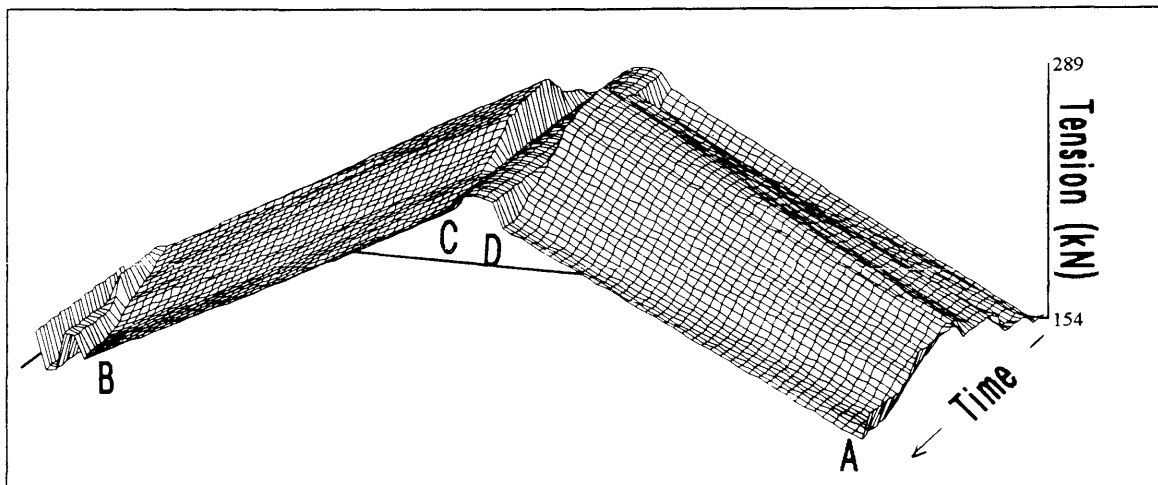


Figure 6.8 Dynamic tensions resulting from an instantaneous 10% increase in takeup mass

The translational response of the takeup is shown in Figure 6.9.

Takeup Displacement (m)

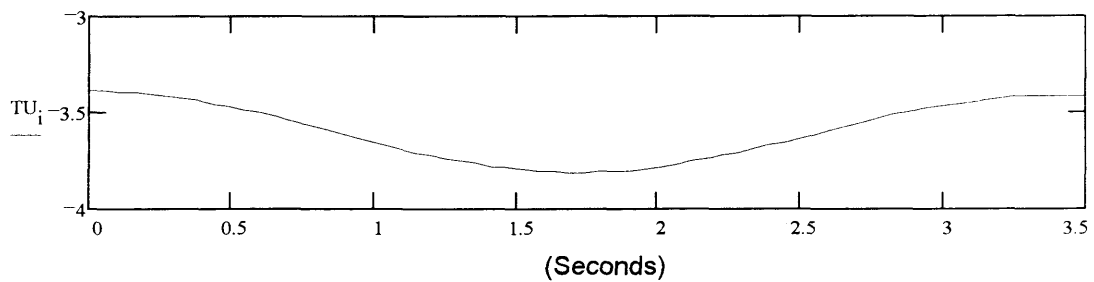


Figure 6.9 Takeup response from an instantaneous 10% increase in takeup mass

The systems natural frequency is calculated as

$$freq = \frac{\sqrt{\frac{0.0254m^2 \cdot 3 \times 10^5 kPa}{100m}}}{2\pi \sqrt{15000kg + 300kg + \frac{5m\pi}{2} \frac{100kg}{m} + \frac{100m \times 100kg/m}{3}}} = 0.315Hz \quad (6.2)$$

The frequency of the takeup oscillation shown in Figure 6.9 is 0.294 Hz. The change in the static equilibrium position of the takeup is calculated by hand to be -0.197 m. Figure 6.9 shows that the takeup motion has an amplitude of twice this amount which is correct.

6.4.2 Dynamic Response to a 400 kN-m Torque

In this test a constant torque of 400 kNm is applied to the upper pulley for 0.07 seconds (200 time steps). The dynamic response of the system is shown in Figure 6.10.

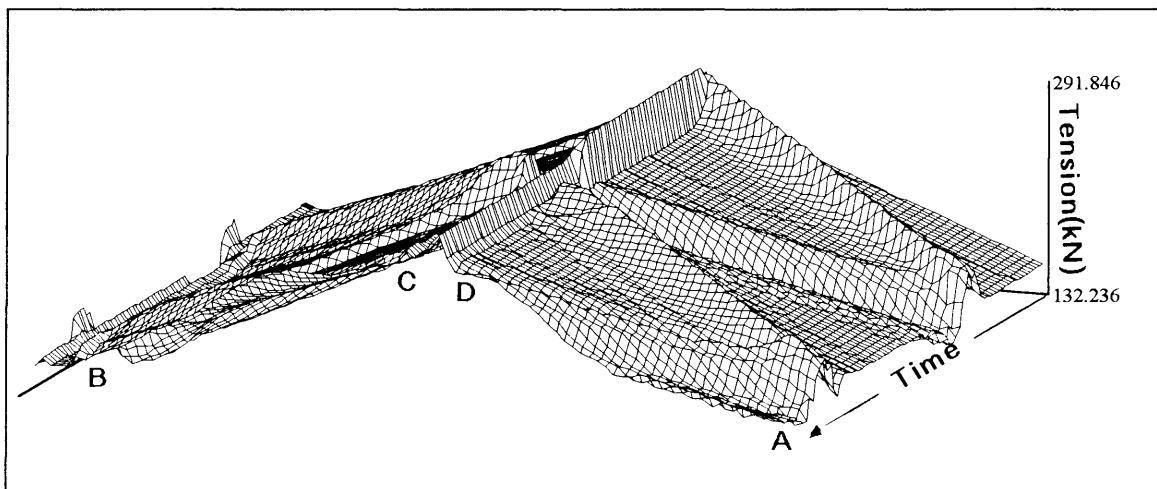


Figure 6.10 Belt tensions resulting from a 400 kN-m torque applied to the upper pulley for 0.07 seconds

In Figure 6.10 the tension and compression waves can be seen propagating around the belt. By looking at the tensions at two points in the belt versus time the speed of sound in the belt can be determined. The tensions at element nodes 60 and 70 (See Figure 6.5 for nodal locations) are shown in Figure 6.11.

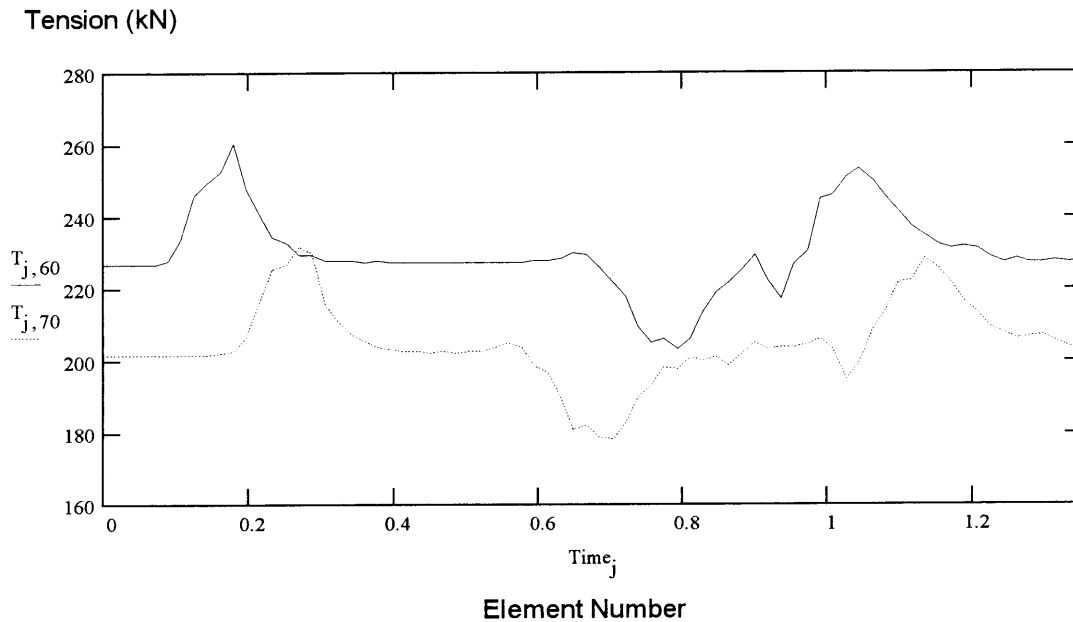


Figure 6.11 Tensions at two points in the belt versus time

The points shown in Figure 6.11 are approximately 26 m apart. The wave speed is calculated from Figure 6.11 as approximately 275 m/s. The analytical wave speed is calculated as 274 m/s.

6.4.3 Dynamic Response to a 400 kN-m Torque With Axial Belt Damping

In this test a constant torque of 400 kNm is applied to the upper pulley for 0.07 seconds (200 time steps). The belt elements have a damping coefficient of 10% of critical damping. The dynamic response of the system is shown in Figure 6.12.

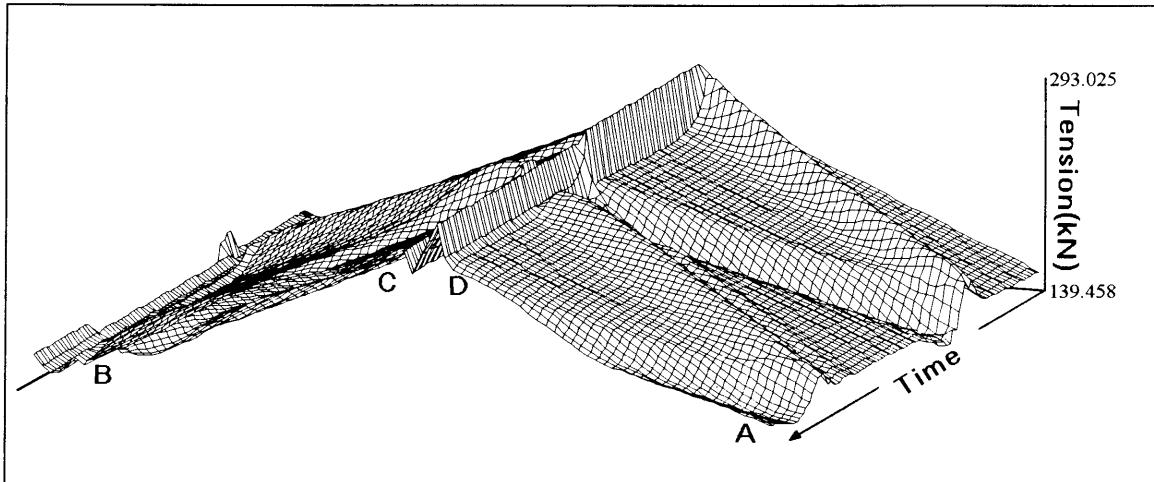


Figure 6.12 Belt tensions resulting from a 400 kN-m torque applied to the upper pulley for 0.07 seconds with axial belt damping of 10% of critical

In Figure 6.12 the tension and compression waves can be seen propagating around the belt. The waves appear to be softer from the damping in the belt. By looking at the tensions at two points in the belt versus time the speed of sound in the belt can be determined. The tensions at element nodes 60 and 70 (See Figure 6.5 for nodal locations) are shown in Figure 6.13.

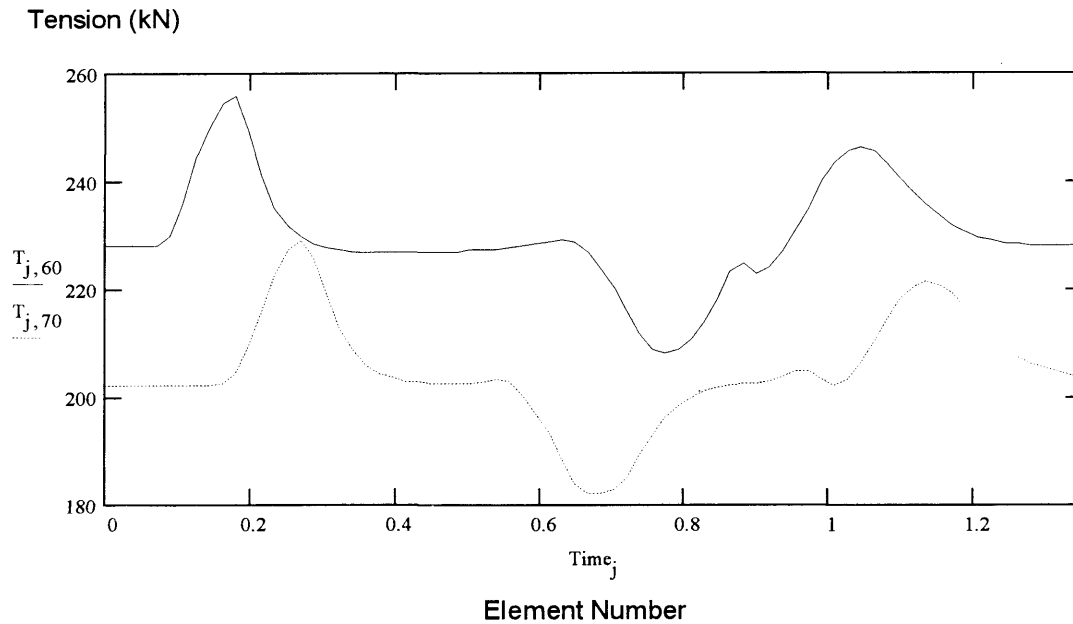


Figure 6.13 Tensions at two points in the belt versus time
with belt damping of 10% of critical

The points shown in Figure 6.13 are approximately 26 m apart. The wave speed calculated from Figure 6.13 as approximately 300 m/s. The wave speed is slightly faster than that of section 4.2 because adding the damping increases the modeling stiffness of the belt elements which in turn increases the wave propagation speed.

Chapter 7.

CONCLUDING REMARKS

This thesis has presented an exciting new transient dynamic discrete model for conveyor belts. The usage of object-oriented programming techniques to construct this model will enable it to be easily customized to meet future needs. This chapter includes the conclusions drawn from this investigation and suggestions for further work.

7.1 Conclusions

The following conclusions can be drawn from this investigation.

1. The standards relating to conveyor belts provide a good estimation of the "static" running tensions of a belt conveyor.
2. Dynamic conveyor belt models currently available to design engineers are limited in several ways: (a) Belt/idler interactions are modeled by including the effective mass of the idlers in with the belt mass and applying rubber hysteresis losses as external friction forces; (b) Belt and material flexure losses are also modeled as external friction forces; (c) There is difficulty in incorporating laboratory data into working models.
3. The transient dynamic discrete model presented in this thesis avoids the above limitations by explicitly modeling the contact between the various conveyor components and the transient-dynamics of each individual conveyor component.

4. The verification tests solved using the new numerical model are in close agreement with analytical solutions.
5. The new model has been developed using object-oriented programming techniques that will allow the model to be easily adapted to include more complex conveyor components.
6. The new modeling method is very compute intensive. This computational expense, however, should not limit the future development of this model because desktop/personal computers will be fast enough to handle the computations necessary in reasonable time in the near future. The computer time required for 3.6 seconds of modeling time of a conveyor system discretized into 90 belt elements on a 486 DX 66 MHz personal computer is approximately 1/4 hour.

7.2 Suggestions for Further Work

There are several areas in which further research is justified by this current investigation. Some of these areas include:

1. Porting the existing computer code to an advanced operating system which can handle significant levels of memory allocation, parallel processing, and multitasking. Windows NT is one possibility.
2. Additional refinements in modeling individual conveyor components, such as:
 - a) Inclusion of pulley and idler bearing friction into the model.
 - b) Addition of a rotational dashpot damper to model internal belt and material damping at the belt nodes from energy losses due to flexure.
 - c) Drives and drive accessories should be developed as they are needed in the usage of the model. These drives may need to include the feedback control. If the drive models are significantly stiffer than the rest of the conveyor model a sub-time

marching scheme may be used (13, 14).

- d) Additional types of takeups and configurations should be developed as needed.
3. Modeling of material transport by including it with the belt nodal masses.
 4. Introduce a coefficient of restitution into the contact between the belt, pulleys and idlers. The laboratory work done by Spaans (11) and Jonkers (12) may be used to obtain realistic coefficients of restitution for belt/idler interactions.
 5. The new transient dynamic model should be comprehensively tested and calibrated using field measurements from operating conveyor systems.
 6. The localized phenomenon of the contact between the belt and pulleys should be studied.
 7. Development of a 3-D model using the same logic as used in the current 2-D model. A 3-D model could include horizontal curves and modeling of pipe conveyors.

REFERENCES CITED

1. CEMA. *Belt Conveyors for Bulk Materials*. Third Edition. Conveyor Equipment Manufacturers Association, 1988.
2. DIN 22101, *Belt Conveyors for Bulk Materials*, 1982.
3. International Standard ISO 5048, *Continuous Mechanical Handling Equipment*, Second Edition, 1989.
4. Harrison, Alex. *Dynamic Measurement and Analysis of Steelcord Conveyor Belts*. Ph.D Thesis, University of Newcastle, Australia, 1984.
5. Markusik, Sylwester. *Dynamika Rozruchu Przenosnikow Tasmowych Z Napędem Jedno-Lub Dwubębnowym Czołowym*. Poland: Zakładzie Graficznym Politechniki Śląskiej, 1982.
6. Funke, H., "The Dynamic Stress of Conveyor Belt Systems When Starting and Stopping." *Braunkohle*, Vol. 26, 1975, No. 3, pp. 64-73.
7. Nordell, L. K. and Z. P. Ciozda. "Transient Belt Stresses During Starting and Stopping: Elastic Response Simulated by Finite Element Methods." *Bulk Solids Handling*, March 1984, pp. 93-98.
8. Nordell, L. K. "The Analysis of Starting and Stopping High Strength, High Capacity Belts Using Modern Engineering Tools." Society of Mining Engineers of AIME, PREPRINT, 1984.

9. Mustoe, G.G.W., Huttelmaier, and Chung. "Assessment of Dynamic Coupled Bending-Axial Effects for Two-Dimensional Deep-Ocean Pipes by the Discrete Element Method." *International Journal of Offshore and Polar Engineering*, Vol. 2, No. 4, December 1992.
10. Taylor, M. and D. Preece. "Simulation of Blasting Induced Rock Motion Using Spherical Element Models." *DEM 1st U.S. Conference*, CSM Press, 1989.
11. Spaans, C., "The Indentation Resistance of Belt Conveyors." Delft Technical University, Dept. of Mech. Eng. WTHD Nr. 103, Jan. 1978.
12. Jonkers, C., "The Indentation Rolling Resistance of Belt Conveyors." *Fördern und Heben*, Vol. 30, 1980, pp. 312-318.
13. Smolinski, Patrick and Shelly Sleith. "Explicit Multi-Time Step Methods for Structural Dynamics." *New Methods in Transient Analysis*, ASME, PVP-Vol. 246/AMD-Vol. 143, 1992, pp. 1-6.
14. Smolinski, Patrick. "An explicit multi-time step integration method for second order equations." *Computer Methods in Applied Mechanics and Engineering*, Vol. 94, No. 1, January 1992, pp. 25-34.

APPENDIX

convpart.h

```
#include <owl.h>
#include <stdio.h>
#include <math.h>

#define PI 3.141592
#define g 9.81

double sqr(double a);

struct PulleyTB {
    char x[10], y[10], radius[10], mass[10];
    WORD ClockWise, CClockWise;
    char mustatic[10], mudynamic[10];
    WORD Carry, Return;
    char spacing[10];
};

class AHPulley {
protected:
    double x, y, vx, vy, ax, ay, theta, omega, alpha;
    double forcex, forcey, torque;
    double mass, inertia;
    double radius;
    double mustatic, mudynamic;
    double distance, theta_in, theta_out;
    double spacing;
    int type;
    int start;
    int rotation;
public:
    AHPulley *next;
```

```

    AHPulley();
// functions which do things
    AHPulley( PulleyTB *tb);
    void Initialize();
    void Draw(HDC hDC);
    virtual void Move(double dt);
    virtual void Name( HFILE hf) { _lwrite(hf, "PULLEY", 7);}
    virtual void AddMass() {}
// functions which set things
    void Start(int i) {start = i;}
    void ForceX ( double a) {forcex += a;}
    void ForceY ( double a) {forcey += a;}
    void Torque ( double a) {torque += a;}
    virtual void DriveForces(int z) {};
    virtual void ZeroForces() {forcex = 0.0; forcey = -mass*g; torque = 0.0;};
    virtual void ZeroKineticEnergy() {omega = 0.0;}
// functions returning things
    double Length();
    double X() { return x;}
    double Y() { return y;}
    virtual double X0() { return x;}
    virtual double Y0() { return y;}
    double Vx() {return vx;}
    double Vy() {return vy;}
    double Ax() {return ax;}
    double Ay() {return ay;}
    double Radius() { return radius;}
    double Inertia() { return inertia;}
    double Omega() { return omega;}
    double Alpha() { return alpha;}
    double Torque() { return torque;}
    int Rotation() {return rotation;}
    double Mustatic() {return mustatic;}
    double Mudynamic() {return mudynamic;}
    double Theta_in() {return theta_in;};
    double Theta_out() {return theta_out;};
    double Distance() {return distance;};
    double Spacing() {return spacing;}
    int Start() {return start;}
    int NumberOfIdlers() {return (int)distance/spacing;}

```

```

        virtual double KineticEnergy() {return 0.5*inertia*sqr(omega);}
        virtual double PotentialEnergy() {return 0.0;}
        virtual double Size() {return sizeof(AHPulley);}
};

struct TakeupTB {
    char x[10], y[10], radius[10], mass[10];
    WORD ClockWise, CClockWise;
    char mustatic[10], mudynamic[10];
    WORD Carry, Return;
    char spacing[10], tumass[10];
};

class AHTakeup : public AHPulley {
protected:
    double tumass, x0, y0;
public:
    AHTakeup( TakeupTB *tb);
    AHTakeup();
    virtual void Name( HFILE hf) { _lwrite(hf, "TAKEUP", 7);}
    virtual double Size() {return sizeof(AHTakeup);}
    virtual void ZeroForces() {forcex = 0.0; forcey = -(tumass+mass)*g;
        torque = 0.0;}
    virtual void Move(double dt);
    virtual double KineticEnergy() {return
        0.5*(mass+tumass)*(sqr(vx)+sqr(vy)) + 0.5*inertia*sqr(omega);}
    virtual double PotentialEnergy() {return (mass+tumass)*g*(y-y0);}
    virtual double X0() {return x0;}
    virtual double Y0() {return y0;}
    virtual void ZeroKineticEnergy() {vy = 0.0; omega = 0.0;}
    virtual void AddMass() {tumass *= 1.1;}
};

struct Drive1TB {
    char x[10], y[10], radius[10], mass[10];
    WORD ClockWise, CClockWise;
    char mustatic[10], mudynamic[10];
    WORD Carry, Return;
    char spacing[10];
    char maxtorque[10], maxrpm[10];
};

```

};

```

class AHDrive1 : public AHPulley {
    protected:
        double maxtorque, maxrpm;
    public:
        AHDrive1( Drive1TB *tb);
        virtual void ZeroForces() { forcex = 0.0; forcey = -mass*g; torque = 0.0;}
        virtual void DriveForces(int z) {if (z) torque += 400000.0;}
        virtual void Name( HFILE hf) { _lwrite(hf, "DRIVE1", 7);}
        virtual double Size() {return sizeof(AHDrive1);}
};

```

```

double Length( AHPulley *Tail); // This function returns the length of the conveyor
double MinRadius( AHPulley *Tail); // Returns the minimum pulley radius in a conveyor
int NumberOfIdlers( AHPulley *Tail); // Returns the number of idlers in the belt
double Potential( AHPulley *Tail);
double Kinetic( AHPulley *Tail);

```

```

// *****
// *** Class Definition of Idlers
// *****

```

```

class AHidler {
    private:
        double x, y;
        double forcex, forcey;
        int contact;
    public:
        AHidler() { contact = 0;}
        double X() {return x;}
        void X(double a) {x = a;}
        void Y(double a) {y = a;}
        double Y() {return y;}
        double Radius() {return 2.0;}
        int Contact() {return contact;}
        void Contact(int a) {contact = a;}
};

```

```

class AHIdlers {

```

```

private:
    double radius;
    int number;
public:
    AHIdler *idler;
    AHIdlers();
    int Number() {return number;}
    void Initialize( AHPulley *tail);
    void Draw( HDC hDC);
};

// *****
// *** Class Definition of Belt
// *****

typedef struct Elementtag {
    double x, y;
    double vx, vy;
    double forcex, forcey;
    double forcex2,forcey2;
} Element;

struct BeltTB {
    char rho[10], E[15], zeta[10], width[10], thick[10];
};

class AHBelt {
private:
    Element *element;
    int number;
    double length, k, kt, c, mass;
    double E, thick, width, rho, zeta;
    double Hor(int i);
    double Ver(int i);
    double ElementLength(int i);
    void Straight( AHPulley *Pulley, int start, int number);
    void Arc( AHPulley *Pulley, int start, int number);
    void InContact( AHPulley *Pulley, double dt);
    void InContact2( AHIdler *idler);
    void InContact3( AHPulley *Pulley, double dt);
};

```

```

public:
    AHBelt();
    void Draw( HDC hDC);
    void Initialize( AHPulley *Tail);
    void ZeroForces();
    void InternalForces();
    void PulleyContact( AHPulley *Tail, double dt);
    void NoFriction( AHPulley *Tail, double dt);
    void IdlerContact( AHIdlers *idlers);
    void Move(double dt);
    void SetBeltProp( BeltTB *tb);
    void GetBeltProp( BeltTB *tb);
    double KineticEnergy();
    double PotentialEnergy();
    double dt() { return 2.0/sqrt(k/mass);}
    double Mass() {return mass;}
    void Tension( FILE *fp);
    void Speed( FILE *fp);
    void SaveElements( HFILE hf);
    void DeleteElements();
    void ReadElements( HFILE hf);
    void ZeroKineticEnergy();
};

```

```

// *****
// *** Conveyor Class Definition
// *****

```

```

class AHConveyor {
private:
    AHPulley *Tail, *Cursor;
    AHBelt *belt;
    AHIdlers *idlers;
    double dt;
public:
    AHConveyor();
    int IsEmpty();
    void AddItem( AHPulley *item);
    void Initialize();
    void Move(int z);

```

```

//      void Stabalize( HDC hDC);
void Stabalize(HDC hDC, FILE *fp, FILE *fp2, FILE *fp3, FILE *fp4);
void Draw( HDC hDC);
void SetBeltProp( BeltTB *tb) {belt->SetBeltProp(tb);}
void GetBeltProp( BeltTB *tb) {belt->GetBeltProp(tb);}
void Print(FILE *fp, FILE *fp2, FILE *fp3, FILE *fp4);
void Save(HFILE hf);
void Read(HFILE hf);
double KineticEnergy();
void ZeroKineticEnergy();
void AddMass() {Tail->AddMass();}
};

```

convpart.cpp

```

#include "convpart.h"
#include <stdio.h>
#include <string.h>
#define KC 10.0E8

// *****
// *** Extra Functions
// *****

double sqr( double a) {
    return a*a;
}

double cube( double a) {
    return a*a*a;
}

double sgn( double a) {
    return a<0.0?-1.0:1.0;
}

double Sum( double *a) {
    int i;
    double sum = 0.0;

```

```

        for (i=0;i<12;i++)
            sum += a[i];
        return sum;
    }

// *****
// *** Pulley Functions
// *****
AHPulley::AHPulley() {
    vx = 0.0;
    vy = 0.0;
    ax = 0.0;
    ay = 0.0;
    theta = 0.0;
    omega = 0.0;
    inertia = mass*radius*radius;
    torque = 0.0;
    theta_out = 0.0;
    theta_in = 2*PI;
}

AHPulley::AHPulley( PulleyTB *tb) {
    x = atof(tb->x);
    y = atof(tb->y);
    radius = atof(tb->radius);
    mass = atof(tb->mass);
    if (tb->ClockWise)
        rotation = 1;
    else
        rotation = -1;
    mustatic = atof(tb->mustatic);
    mudynamic = atof(tb->mudynamic);
    if (tb->Carry)
        type = 0;
    else
        type = 1;
    spacing = atof(tb->spacing);
    vx = 0.0;
    vy = 0.0;
    ax = 0.0;

```

```

    ay = 0.0;
    theta = 0.0;
    omega = 0.0;
    inertia = mass*radius*radius;
    torque = 0.0;
    theta_out = 0.0;
    theta_in = 2*PI;
//    AHPulley();
}

void AHPulley::Initialize() {
    double error;
    do {
        distance = sqrt(sqrt((next->X0()-next->rotation*next->radius
            *sin(theta_out))-(X0()-rotation*radius*sin(theta_out)))
            +sqrt((next->Y0()+next->rotation*next->radius*cos(theta_out))-
            (Y0()+rotation*radius*cos(theta_out))));
        if (distance == 0.0) {
            theta_out = 0.0;
            next->theta_in = 2*PI;
            return;
        }
        theta_out = atan2((next->Y0()+next->rotation*next->radius
            *cos(theta_out))-(Y0()+rotation*radius*cos(theta_out)),
            (next->X0()-next->rotation*next->radius*sin(theta_out))
            -(X0()-rotation*radius*sin(theta_out)));
        error = next->theta_in - theta_out;
        next->theta_in = theta_out;
    } while (sqrt(error) > 0.0001);
}

void AHPulley::Draw( HDC hDC) {
    // Circle
    Ellipse( hDC, x+radius,y+radius,x-radius,y-radius);
    MoveTo(hDC, x, y);
    LineTo(hDC, x+sin(theta)*radius,y+cos(theta)*radius);
    if (torque>0.0) {
        MoveTo(hDC, x - radius/5.0, y);
        LineTo(hDC, x + radius/5.0, y);
    }
}

```

```

}

void AHPulley::Move( double dt) {
    alpha = torque/inertia;
    omega += alpha*dt;
    theta += omega*dt;
}

double AHPulley::Length(void) {
    double arc;

    arc = theta_out - theta_in;
    if (arc>0.0)
        arc = -1.0*(2.0*PI - arc);
    if (Rotation()=-1)
        arc = (2.0*PI + arc);

    return distance + fabs(arc)*radius;
}

// *****
// *** Takeup Functions
// *****

AHTakeup::AHTakeup() : AHPulley() {
}

AHTakeup::AHTakeup( TakeupTB *tb) : AHPulley() {
    x = x0 = atof(tb->x);
    y = y0 = atof(tb->y);
    radius = atof(tb->radius);
    mass = atof(tb->mass);
    if (tb->ClockWise)
        rotation = 1;
    else
        rotation = -1;
    mustatic = atof(tb->mustatic);
    mudynamic = atof(tb->mudynamic);
    if (tb->Carry)
        type = 0;
}

```

```

        else
            type = 1;
            spacing = atof(tb->spacing);
            tumass = atof(tb->tumass);
            vx = 0.0;
            vy = 0.0;
            theta = 0.0;
            omega = 0.0;
            inertia = mass*radius*radius;
            torque = 0.0;
            theta_out = 0.0;
            theta_in = 2*PI;
    }

void AHTakeup::Move(double dt) {
    ax = 0.0;
    ay = forcey/(tumass+mass);
//    vx += dt*forcex/(tumass+mass);
    vy += dt*ay;
//    x += vx*dt;
    y += vy*dt;
    alpha = torque/inertia;
    omega += dt*alpha;
    theta += omega*dt;
}

// *****
// *** Drive1 Functions
// *****

AHDrive1::AHDrive1( Drive1TB *tb) : AHPulley() {
    x = atof(tb->x);
    y = atof(tb->y);
    radius = atof(tb->radius);
    mass = atof(tb->mass);
    if (tb->ClockWise)
        rotation = 1;
    else
        rotation = -1;
    mustatic = atof(tb->mustatic);
}

```

```

    modynamic = atof(tb->mdynamic);
    if (tb->Carry)
        type = 0;
    else
        type = 1;
    spacing = atof(tb->spacing);
    maxtorque = atof(tb->maxtorque);
    maxrpm = atof(tb->maxrpm);
    vx = 0.0;
    vy = 0.0;
    theta = 0.0;
    omega = 0.0;
    inertia = mass*radius*radius;
    torque = 0.0;
    theta_out = 0.0;
    theta_in = 2*PI;
}

// *****
// ** Belt Functions
// *****

AHBelt::AHBelt() {
    element = NULL;
    E = 300000000.0;
    zeta = 0.5;
    width = 1.0;
    thick = 0.0254;
    rho = 50.0;
    length = 1.0;
}

void AHBelt::Initialize( AHPulley *Tail) {

    // check to see if belts have been previously defined
    if (element)
        delete element;

    // determine minimum element length from minimum pulley radius
    length = MinRadius(Tail)*PI/180.0*30.0;
}

```

```

// determine # of elements
number = (int)(Length(Tail)/length) + 1;

// re-calculate element length
length = Length(Tail)/number;

// calculate stiffness, mass, damping factors
mass = rho * length;
k = E*thick*width/length;
kt = 3.0*E*width*thick*thick*thick/length;
c = 2.0*mass*zeta*sqrt(k/mass);

// allocate memory for elements
element = new Element[number];

int i;
// zero velocities and accelerations
for (i=0;i<number; i++) {
    element[i].vx = 0.0;
    element[i].vy = 0.0;
    element[i].forcex2 = 0.0;
    element[i].forcey2 = 0.0;
}

// locate elements
AHPulley *temp = Tail;
i = 0;
do {
    Arc( temp, i, (temp->Length() - temp->Distance())/length);
    temp->Start(i);
    i += (temp->Length() - temp->Distance())/length;
    if (temp->next == Tail)
        Straight( temp, i, number-i);
    else
        Straight( temp, i, temp->Distance()/length);
    i += temp->Distance()/length;
    temp = temp->next;
} while (temp != Tail);
length *= 1.0;

```

```

}
```

```

void AHBelt::Straight( AHPulley *Pulley, int start, int number) {
    int i;

    for (i=0;i<number;i++) {
        element[i+start].x = Pulley->X0() - Pulley->Rotation()
            *Pulley->Radius()*sin(Pulley->Theta_out())
            + i*Pulley->Distance()/number*cos(Pulley->Theta_out());
        element[i+start].y = Pulley->Y0() + Pulley->Rotation()
            *Pulley->Radius()*cos(Pulley->Theta_out())
            + i*Pulley->Distance()/number*sin(Pulley->Theta_out());
    }
}

```

```

void AHBelt::Arc( AHPulley *Pulley, int start, int number) {
    double theta, arc;
    int i;

    arc = Pulley->Theta_out() - Pulley->Theta_in();
    if (arc>0.0)
        arc = -1.0*(2.0*PI - arc);
    if (Pulley->Rotation()==-1)
        arc = (2.0*PI + arc);

    for (i=0;i<number;i++) {
        theta = Pulley->Theta_in() + i*arc/number;
        element[i+start].x = Pulley->X0() - Pulley->Rotation()
            *Pulley->Radius()*sin(theta);
        element[i+start].y = Pulley->Y0() + Pulley->Rotation()
            *Pulley->Radius()*cos(theta);
    }
}

```

```

void AHBelt::ZeroForces() {
    int i;
    for (i=0;i<number;i++) {
        element[i].forcex = 0.0;
        element[i].forcey = -mass*g;
        element[i].forcex2 = 0.0;
    }
}

```

```

        element[i].forcey2 = 0.0;
    }
}

void AHBelt::InternalForces() {
    int h, i, j;
    double delt, deltv, force, beta;

    for (i=0;i<number;i++) {
        j = i+1;
        if (j==number)
            j = 0;
        h = i-1;
        if (h==-1)
            h = number - 1;
        // spring forces
        delt = ElementLength( i ) - length;
        deltv = ((element[j].vx-element[i].vx)*Hor(i)
            + (element[j].vy-element[i].vy)*Ver(i))/ElementLength(i);
        force = (delt*k+deltv*c)/ElementLength(i);
        // axial forces x-direction - spring & damping
        element[i].forcex += force*Hor(i);
        element[j].forcex -= force*Hor(i);
        // axial forces y-direction - spring & damping
        element[i].forcey += force*Ver(i);
        element[j].forcey -= force*Ver(i);

        beta = kt*atan2(Hor(i)*Ver(h)-Hor(h)*Ver(i),
            Hor(i)*Hor(h)+Ver(i)*Ver(h));
        // rotational forces x-direction - spring only
        element[h].forcex -= beta*Ver(h)/sqr(ElementLength(h));
        element[i].forcex += beta*(Ver(i)/sqr(ElementLength(i)) +
            Ver(h)/sqr(ElementLength(h)));
        element[j].forcex -= beta*Ver(i)/sqr(ElementLength(i));
        // rotational forces y-direction - spring only
        element[h].forcey += beta*Hor(h)/sqr(ElementLength(h));
        element[i].forcey -= beta*(Hor(i)/sqr(ElementLength(i)) +
            Hor(h)/sqr(ElementLength(h)));
        element[j].forcey += beta*Hor(i)/sqr(ElementLength(i));
    }
}

```

}

```
double AHBelt::Hor( int i) {
    int j;

    j = i+1;
    if (j==number)
        j = 0;
    if (i==-1)
        i = number-1;

    return element[j].x-element[i].x;
}
```

```
double AHBelt::Ver( int i) {
    int j;

    j = i+1;
    if (j==number)
        j = 0;
    if (i==-1)
        i = number-1;

    return element[j].y-element[i].y;
}
```

```
double AHBelt::ElementLength( int i) {
    return sqrt(sqr(Hor(i))+sqr(Ver(i)));
}
```

```
void AHBelt::PulleyContact(AHPulley *Tail, double dt) {
    int i;
    AHPulley *temp=Tail;

    do {
        InContact( temp, dt);
        temp = temp->next;
    } while ( temp != Tail);
}
```

```

void AHBelt::InContact(AHPulley *Pulley, double dt) {
    double a[12],b[12],c,delt,trv, shift, fric[12];
    double frictionforce[12], norforce[12], tanforce, masse[12];
    int i,j,k,node[12];
    BOOL SLIP;

    i = Pulley->Start()-2;
    if (i<0)
        i += number;
    k = i;

    do {
        // loop over belt elements not in contact
        i++;
        if (i==number) i = 0;
        a[0] = element[i].x - Pulley->X();           // find start
        b[0] = element[i].y - Pulley->Y();
        c = sqrt(sqrt(a[0]) + sqrt(b[0]));
        delt = c - Pulley->Radius();
    } while (delt>0.0 && i!=k);

    Pulley->Start(i);

    for (j=0;j<12;j++) {

        a[j] = element[i].x - Pulley->X();
        b[j] = element[i].y - Pulley->Y();
        c = sqrt(sqrt(a[j]) + sqrt(b[j]));
        delt = Pulley->Radius() - c;

        a[j] /= c;
        b[j] /= c;

        if (delt>0.0) {
            norforce[j] = delt/Pulley->Radius()*KC;
            element[i].forcex += norforce[j]*a[j];
            element[i].forcey += norforce[j]*b[j];
            Pulley->ForceX( -norforce[j]*a[j]);
            Pulley->ForceY( -norforce[j]*b[j]);

            trv = -(element[i].vx-Pulley->Vx())*b[j]

```

```

        + (element[i].vy-Pulley->Vy())*a[j]
        - Pulley->Radius()*Pulley->Omega());
tanforce = -element[i].forcex*b[j] + element[i].forcey*a[j];
frictionforce[j] = trv/dt*mass + tanforce;
masse[j] = mass;
    }
    else {
        norforce[j] = 0.0;
        frictionforce[j] = 0.0;
        masse[j] = 0.0;
    }
    node[j] = i;
    i++;
    if (i==number) i = 0;
}
do {
    SLIP = FALSE;
    shift = (sqr(Pulley->Radius()*Sum(frictionforce)
        +Pulley->Radius()*Pulley->Torque()*mass/
        (Pulley->Inertia() + sqr(Pulley->Radius()*Sum(masse)));
    for (j=0;j<12;j++) {
        fric[j] = (frictionforce[j]-shift)*masse[j]/mass;
        if (fabs(fric[j]) > Pulley->Mustatic()*norforce[j]) {
            fric[j] = Pulley->Mudynamic()*norforce[j]*sgn(fric[j]);
            masse[j] = 0.0;
            frictionforce[j] = 0.0;
            Pulley->Torque( Pulley->Radius()*fric[j]);
            element[node[j]].forcex2 += fric[j]*b[j];
            element[node[j]].forcey2 -= fric[j]*a[j];
            Pulley->ForceX( -fric[j]*b[j]);
            Pulley->ForceY( +fric[j]*a[j]);
            SLIP = TRUE;
        }
    }
} while (SLIP == TRUE);

for (j=0;j<12;j++) {
    Pulley->Torque( Pulley->Radius()*fric[j]);
    element[node[j]].forcex2 += fric[j]*b[j];
    element[node[j]].forcey2 -= fric[j]*a[j];
}

```

```

        Pulley->ForceX( -fric[j]*b[j]);
        Pulley->ForceY( +fric[j]*a[j]);
    }
}

void AHBelt::NoFriction(AHPulley *Tail, double dt) {
    int i;
    AHPulley *temp=Tail;

    do {
        InContact3( temp, dt);
        temp = temp->next;
    } while ( temp != Tail);
}

void AHBelt::InContact3(AHPulley *Pulley, double dt) {
    double a[12],b[12],c,delt, norforce;
    int i,j,k;

    i = Pulley->Start()-2;
    if (i<0)
        i += number;
    k = i;

    do {
        // loop over belt elements not in contact
        i++;
        if (i==number) i = 0;
        a[0] = element[i].x - Pulley->X();           // find start
        b[0] = element[i].y - Pulley->Y();
        c = sqrt(sqrt(a[0]) + sqrt(b[0]));
        delt = c - Pulley->Radius();
    } while (delt>0.0 && i!=k);

    Pulley->Start(i);

    for (j=0;j<12;j++) {

        a[j] = element[i].x - Pulley->X();
        b[j] = element[i].y - Pulley->Y();
        c = sqrt(sqrt(a[j]) + sqrt(b[j]));
    }
}

```

```

    delt = Pulley->Radius() - c;

    a[j] /= c;
    b[j] /= c;

    if (delt>0.0) {
        norforce = delt/Pulley->Radius()*KC;
        element[i].forcex += norforce*a[j];
        element[i].forcey += norforce*b[j];
        Pulley->ForceX( -norforce*a[j]);
        Pulley->ForceY( -norforce*b[j]);
    }
    i++;
    if (i==number) i = 0;
}
}

void AHBelt::IdlerContact(AHIdlers *idlers) {
    int i;
    for (i=0;i<idlers->Number();i++) {           // loop over idlers
        InContact2(&idlers->idler[i]);
    }
}

void AHBelt::InContact2(AHIdler *Idler) {
    double a,b,c,d,dist,arm_i,arm_j,force;
    int i, j, k;
    i = Idler->Contact()-1;
    j = i + 1;
    if (i==-1) i = number-1;
    if (j==number) j = 0;
    k = i;
    do {
        a = Idler->X() - element[i].x;
        b = Idler->Y() - element[i].y;
        c = Idler->X() - element[j].x;
        d = Idler->Y() - element[j].y;

        dist = (c*Ver(i) - d*Hor(i))/ElementLength(i);

```

```

        if (abs(dist) < Idler->Radius()) { // may be in contact
            arm_i = sqrt(sqr(c) + sqr(d) - sqr(dist));
            arm_j = sqrt(sqr(a) + sqr(b) - sqr(dist));

            if ((arm_i > arm_j ? arm_i : arm_j) < ElementLength(i)) { // in contact
                force = 10000.0 * (Idler->Radius() - abs(dist)) * sgn(dist)
                    /sqr(ElementLength(i));
                element[i].forcex -= force*Ver(i)*arm_i;
                element[i].forcey += force*Hor(i)*arm_i;
                element[j].forcex -= force*Ver(i)*arm_j;
                element[j].forcey += force*Hor(i)*arm_j;

                Idler->Contact(i);
                return;
            }
        }
        i++;
        if (i==number) i = 0;
        j++;
        if (j==number) j = 0;
    } while (i!=k);
}

void AHBelt::Move(double dt) {
    int i;

    for (i=0;i<number;i++) {
        element[i].vx += (element[i].forcex+element[i].forcex2)/mass*dt;
        element[i].vy += (element[i].forcey+element[i].forcey2)/mass*dt;
        element[i].x += element[i].vx*dt;
        element[i].y += element[i].vy*dt;
    }
}

void AHBelt::Draw( HDC hDC) {
    int i;
    HPEN hPen, hOldPen, hTempPen;
    double maxforce = 1.0, force;

    MoveTo(hDC, element[number-1].x, element[number-1].y);

```

```

hPen = CreatePen( PS_SOLID, 1, RGB(0,0,255));
hOldPen = SelectObject( hDC, hPen);
for (i=0;i<number;i++) {
    LineTo(hDC, element[i].x, element[i].y);
    force = sqrt(sqrt(element[i].forcex2)+sqrt(element[i].forcey2));
    if (force > maxforce) maxforce = force;
}

hPen = CreatePen( PS_SOLID, 1, RGB(255,0,0));
hTempPen = SelectObject( hDC, hPen);
DeleteObject(hTempPen);
for (i=0;i<number;i++) {
    MoveTo(hDC, element[i].x,element[i].y);
    LineTo(hDC, element[i].x+element[i].forcex2/maxforce*25.0,
           element[i].y+element[i].forcey2/maxforce*25.0);
}

SelectObject( hDC, hOldPen);
DeleteObject(hPen);
}

void AHBelt::SetBeltProp( BeltTB *tb) {
    rho = atof(tb->rho);
    E = atof(tb->E);
    zeta = atof(tb->zeta);
    thick = atof(tb->thick);
    width = atof(tb->width);

    mass = rho * length;
    k = E*thick*width/length;
    kt = 3.0*E*width*thick*thick*thick/length;
    c = 2.0*mass*zeta*sqrt(k/mass);
}

void AHBelt::GetBeltProp( BeltTB *tb) {
    sprintf( tb->rho, "%4.2f", rho);
    sprintf( tb->E, "%9.1f", E);
    sprintf( tb->zeta, "%4.2f",zeta);
    sprintf( tb->thick, "%5.3f",thick);
}

```

```

        sprintf( tb->width, "%4.2f",width);
    }

double AHBelt::KineticEnergy() {
    int i;
    double KE = 0.0;
    for (i=0;i<number;i++)
        KE += 0.5*(sqr(element[i].vx)+sqr(element[i].vy))*mass;
    return KE;
}

double AHBelt::PotentialEnergy() {
    int i;
    double PE = 0.0;
    for (i=0;i<number;i++)
        PE += 0.5*k*sqr(ElementLength(i)-length) + mass*10.0*element[i].y;
    return PE;
}

void AHBelt::Tension(FILE *fp) {
    int i;
    for (i=0;i<number;i++) {
        fprintf(fp, "%4.3f ",(ElementLength(i)-length)*k);
    }
    fprintf(fp, "\n");
}

void AHBelt::Speed(FILE *fp) {
    int i;
    for (i=0;i<number;i++) {
        fprintf(fp, "%4.3f ",sqrt(sqr(element[i].vx)+sqr(element[i].vy)));
    }
    fprintf(fp, "\n");
}

void AHBelt::SaveElements(HFILE hf) {
    _lwrite( hf, element, sizeof(Element)*number);
}

void AHBelt::DeleteElements() {

```

```
        delete element;
        element = NULL;
    }

void AHBelt::ReadElements(HFILE hf) {
    element = NULL;
    element = new Element[number];
    _lread( hf, element, sizeof(Element)*number);
}

void AHBelt::ZeroKineticEnergy() {
    int i;
    for (i=0;i<number;i++) {
        element[i].vx = 0.0;
        element[i].vy = 0.0;
    }
}

// *****
// *** Idler Functions
// *****

AHIdlers::AHIdlers() {
    idler = NULL;
    radius = 2.0;
}

void AHIdlers::Initialize( AHPulley *Tail) {

    if (idler)
        delete idler;

    number = NumberOfIdlers( Tail);

    if (!number)
        return;

    idler = new AHIdler[number];

    AHPulley *temp = Tail;
```

```

int i=0,j;
double startx, starty;
do {
    startx = temp->X0()-sin(temp->Theta_out())*temp->Rotation()
        *temp->Radius()+ sin(temp->Theta_out())*radius
        *(cos(temp->Theta_out())<0.0?-1.0:1.0) + ((temp->Distance()
        -(temp->NumberOfIdlers()-1)*temp->Spacing())/2.0)
        *cos(temp->Theta_out());
    starty = temp->Y0()+cos(temp->Theta_out())*temp->Rotation()
        *temp->Radius() - cos(temp->Theta_out())*radius
        *(cos(temp->Theta_out())<0.0?-1.0:1.0) + ((temp->Distance()
        -(temp->NumberOfIdlers()-1)*temp->Spacing())/2.0)
        *sin(temp->Theta_out());
    for (j=0;j<temp->NumberOfIdlers();j++) {
        idler[i+j].X(startx + j*temp->Spacing()*cos(temp->Theta_out()));
        idler[i+j].Y(starty + j*temp->Spacing()*sin(temp->Theta_out()));
    }
    i += temp->NumberOfIdlers();
    temp = temp->next;
} while (temp != Tail);
}

void AHIdlers::Draw( HDC hDC) {
    int i;
    for (i=0;i<number;i++) {
        Ellipse( hDC, idler[i].X()-radius,idler[i].Y()-radius,
            idler[i].X()+radius,idler[i].Y()+radius);
    }
}

// *****
// *** Conveyor Functions
// *****

AHConveyor::AHConveyor() {
    Tail = NULL;
    Cursor = NULL;
    belt = new AHBelt();
    idlers = new AHIdlers();
}

```

```

void AHConveyor::Draw(HDC hDC) {
    AHPulley *temp;
    temp = Tail;
    do {
        temp->Draw(hDC);
        temp=temp->next;
    } while (temp!=Tail);

    belt->Draw(hDC);
    idlers->Draw(hDC);
}

void AHConveyor::Move(int z) {
    AHPulley *temp;
    temp = Tail;
    do {
        temp->ZeroForces();
        temp->DriveForces(z);
        temp=temp->next;
    } while (temp!=Tail);

    belt->ZeroForces();
    belt->InternalForces();
    belt->PulleyContact( Tail, dt);
    belt->IdlerContact( idlers);
    belt->Move(dt);

    temp = Tail;
    do {
        temp->Move(dt);
        temp=temp->next;
    } while (temp!=Tail);
}

void AHConveyor::Stabalize(HDC hDC, FILE *fp, FILE *fp2, FILE *fp3, FILE *fp4) {
    AHPulley *temp;
    char buffer[80];
    double maxKE = 0.0, maxmaxKE = 0.0, ke=0.0, keold=0.0;

```

```

int i;
for (i=0;i<10000;i++) {
    temp = Tail;
    do {
        temp->ZeroForces();
        temp=temp->next;
    } while (temp!=Tail);

    belt->ZeroForces();
    belt->InternalForces();
    belt->NoFriction( Tail, dt);
    belt->IdlerContact( idlers);
    belt->Move(dt);

    temp = Tail;
    do {
        temp->Move(dt);
        temp=temp->next;
    } while (temp!=Tail);

    ke = KineticEnergy();
    if (ke < keold) {
        ZeroKineticEnergy();
        ke = 0.0;
        maxKE = keold;
        if (maxKE > maxmaxKE)
            maxmaxKE = maxKE;
        sprintf(buffer, "%5.1f%% of max KE ",
            maxKE/maxmaxKE*100.0);
        TextOut( hDC, 100, 200, buffer, strlen(buffer));
    }
    if (!(i%100)) {
        Print( fp, fp2, fp3, fp4);
    }
    keold = ke;
}
}

int AHConveyor::IsEmpty() {
    return !Tail;
}

```

```

}
```

```

void AHConveyor::AddItem( AHPulley *item) {
    if (IsEmpty()) {
        Tail = item;
        Tail->next = item;
        Cursor = Tail;
    }
    else {
        item->next = Cursor->next;
        Cursor->next = item;
        Cursor = Cursor->next;
    }
    Initialize();
}

```

```

void AHConveyor::Initialize() {
    AHPulley *temp;
    temp = Tail;
    do {
        temp->Initialize();
        temp=temp->next;
    } while (temp!=Tail);

    belt->Initialize( Tail);
    idlers->Initialize( Tail);
    dt = belt->dt()/2.0;
    double dt2 = 2.0/sqrt(KC/belt->Mass())/2.0;
    if (dt2 < dt)
        dt = dt2;
}

```

```

void AHConveyor::Print( FILE *fp, FILE *fp2, FILE *fp3, FILE *fp4) {
    char buffer[80];
    fprintf( fp, "%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\t%f\n", Tail->Y(),
        belt->PotentialEnergy(),belt->KineticEnergy(),
        Potential(Tail),Kinetic(Tail),Tail->Omega(),Tail->Torque(),
        Tail->next->Omega(),Tail->next->Torque());
    belt->Tension(fp2);
    belt->Speed(fp3);
}

```

```
}

```

```
void AHConveyor::Save( HFILE hf) {
    AHPulley *temp = Tail;

    _lwrite(hf, "Andrew's Belt Program Ver 1.0", 30);
    do {
        temp->Name(hf);
        _lwrite(hf, temp, temp->Size());
        temp=temp->next;
    } while (temp!=Tail);
    _lwrite(hf, "__END_",7);

    _lwrite(hf, belt, sizeof(AHBelt));
    belt->SaveElements( hf);
}

void AHConveyor::Read( HFILE hf) {
    AHPulley *temp;
    char buffer[80];
    _lread(hf, buffer, 30);
    if(strcmp(buffer,"Andrew's Belt Program Ver 1.0")) {
        MessageBox( NULL, "Wrong File Type!!", "Error",
            MB_ICONEXCLAMATION | MB_OK);
        return;
    }
    do {
        _lread(hf, buffer, 7);
        if (!strcmp(buffer,"PULLEY")) {
            temp = new AHPulley();
            _lread(hf, temp, sizeof(AHPulley));
            AddItem(temp);
        }
        else if (!strcmp(buffer,"TAKEUP")) {
            temp = new AHTakeup();
            _lread(hf, temp, sizeof(AHTakeup));
            AddItem(temp);
        }
    } while (strcmp(buffer,"__END_"));
    belt->DeleteElements();
}

```

```

        _lread( hf, belt, sizeof(AHBelt));
        belt->ReadElements(hf);
    }

void AHConveyor::ZeroKineticEnergy() {
    AHPulley *temp;
    temp = Tail;
    do {
        temp->ZeroKineticEnergy();
        temp=temp->next;
    } while (temp!=Tail);

    belt->ZeroKineticEnergy();
}

double AHConveyor::KineticEnergy() {
    return Kinetic(Tail) + belt->KineticEnergy();
}

double Length( AHPulley *Tail) {
    double length = 0.0;
    AHPulley *temp;

    if (Tail==NULL)
        return 0.0;
    temp = Tail;
    do {
        length += temp->Length();
        temp = temp->next;
    } while (temp != Tail);
    return length;
}

double MinRadius( AHPulley *Tail) {
    double min = Tail->Radius();
    AHPulley *temp;

    temp = Tail;
    do {
        if (temp->Radius()<min)

```

```
        min = temp->Radius();
        temp = temp->next;
    } while (temp != Tail);
    return min;
}

int NumberOfIdlers( AHPulley *Tail) {
    int number = 0;
    AHPulley *temp;

    temp = Tail;
    do {
        number += temp->NumberOfIdlers();
        temp = temp->next;
    } while (temp != Tail);
    return number;
}

double Potential ( AHPulley *Tail) {
    double PE = 0.0;
    AHPulley *temp;

    temp = Tail;
    do {
        PE += temp->PotentialEnergy();
        temp = temp->next;
    } while (temp != Tail);
    return PE;
}

double Kinetic ( AHPulley *Tail) {
    double KE = 0.0;
    AHPulley *temp;

    temp = Tail;
    do {
        KE += temp->KineticEnergy();
        temp = temp->next;
    } while (temp != Tail);
    return KE;
}
```