

CHARACTERIZATION OF A NIP IMPINGED
WOUND ROLL

By

PAUL HOFFECKER

Bachelor of Science

University of Colorado

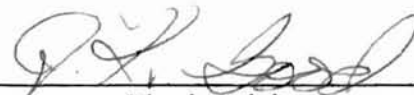
Boulder, Colorado

1994


Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 1997

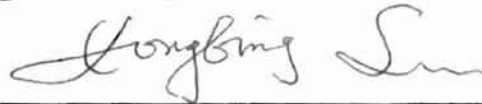
CHARACTERIZATION OF A NIP IMPINGED
WOUND ROLL

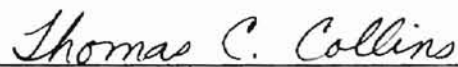
Thesis Approved:



Thesis Advisor







Dean of the Graduate College

ACKNOWLEDGMENTS

Rarely has the verse "I can do all things through Him who strengthens me" (Phillipians 4:13) applied more than in this project's undertaking. Every step was a new experience in faith and praise of God.

My sincere appreciation to Dr. J. K. Good for his wisdom and insight. His intuition motivated and inspired me to explore. In addition I thank him for allowing me great leeway in that exploration.

I extend my gratitude to my family for their support and encouragement which made the rough times smoother. But to Heather I owe a great debt for her constant care, understanding and interest. I am truly blessed by her love.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION.....	1
Webs and Wound Rolls.....	1
Web Handling and Winders.....	2
Web Geometry.....	3
Modeling.....	4
Objective.....	4
II. LITERATURE SURVEY.....	5
Center Winding Models.....	5
Adding a Nip Roller.....	6
III. DEVELOPMENT OF A THEORETICAL MODEL.....	8
Pfeiffer's Stress versus Strain Relationships.....	8
Hertzian Contact.....	11
Finite Elements.....	16
Material Behavior Parameters.....	25
Nip Roll Impingement.....	29
Contact Profile.....	39
IV. RESULTS.....	41
Material Behavior Parameters.....	41
The Standard.....	43
Geometry.....	49
Material.....	55
Loading.....	58
Modeling.....	60
Incremental Deformations and Gapping.....	66
V. CONCLUSIONS.....	71
Assumptions.....	73
Further Investigation.....	74

VI.	BIBLIOGRAPHY.....	75
VII.	APPENDIX A, IMPINGE PROGRAM CODE.....	77
VIII.	APPENDIX B, EXAMPLE OUTPUT FILE (PINGELPW.OUT).....	132
IX.	APPENDIX C, STANDARD INPUT FILE (DATASAMP.IN).....	134
X.	APPENDIX D, EXAMPLE OUTPUT FILES (PINGESUM.OUT, PINGEPRO.OUT, PINGEINC.OUT).....	136
XI.	APPENDIX E, EXAMPLE OUTPUT FILE (PINGEDEF.OUT).....	142

LIST OF TABLES

Table		Page
1.	FEM Example Dimensions.....	32
2.	Standard Configuration.....	44
3.	Stiff Polyester Configuration.....	67

LIST OF FIGURES

Figure	Page
1. Wound Rolls.....	1
2. Center Winder.....	2
3. Center Winder with Undriven Nip.....	3
4. Hertzian Cylinder in Contact.....	12
5. Cylinders in Cross Section.....	13
6. Beam Element.....	17
7. Beam Loads.....	18
8. Winkler Element.....	19
9. Nip Roller as a Beam.....	20
10. DOF Links.....	21
11. IMPINGE Program Flowchart.....	23-24
12. HERTZIAN Program Flowchart.....	27
13. Wound Roll Behavior versus Geometry.....	29
14. CMD Locations.....	30
15. Winding Geometry End View.....	31
16. FEM of Winding System.....	32
17. Example FEM Mesh.....	33

18.	CONTACT Program Flowchart.....	35-36
19.	Stiffness Determination Loop.....	38
20.	Deformation/ Stiffness Cycle.....	38
21.	Deformation versus Effective Radius.....	41
22.	Stiffness and Load per Width versus Deformation for 83 μm Newsprint.....	42
23.	Stiffness and Load per Width versus Deformation for 25 μm Polyester.....	43
24.	Deformation across the Width for the Standard Configuration.....	46
25.	Load per Width across the Width for the Standard Configuration.....	47
26.	Maximum Pressure across the Width for the Standard Configuration....	48
27.	Deformation across varying Wound Roll Widths.....	49
28.	Deformation across equal Wound Roll/ Nip Roll Widths.....	51
29.	Maximum Pressure and Load per Width across equal Wound Roll/ Nip Roll Widths.....	52
30.	Deformation across the Wound Roll Width for Nips with Reduced MOI ends.....	53
31.	Load per Width across the Wound Roll Width for Nips with Reduced MOI ends.....	54
32.	Maximum Pressure across the Wound Roll Width for Nips with Reduced MOI ends.....	54

33.	Deformation across the Wound Roll Width for varying Nip and Wound Roll Materials.....	55
34.	Load per Width across the Wound Roll Width for varying Nip and Wound Roll Materials.....	56
35.	Maximum Pressure across the Wound Roll Width for varying Nip and Wound Roll Materials.....	57
36.	Deformation across the 83 μm material Wound Roll Width for Different Applied Loads.....	58
37.	Deformation across the 25 μm material Wound Roll Width for Different Applied Loads.....	59
38.	Deformation across the Wound Roll Width for Varying number of Load Steps.....	60
39.	Load per Width across the Wound Roll Width for Varying number of Load Steps.....	61
40.	Maximum Pressure across the Wound Roll Width for Varying number of Load Steps.....	62
41.	Deformation across the Wound Roll Width for Two Different Starting Forces	63
42.	Deformation across the Width due to Wound Roll Material and Iteration.....	64
43.	Deformation across the Wound Roll Width for Nips with Rotationally Constrained Ends.....	64

44.	Load per Width across the Wound Roll Width for Nips with Rotationally Constrained Ends.....	65
45.	Maximum Pressure Distribution across the Wound Roll Width for Nips with Rotationally Constrained Ends.....	65
46.	Deformation across the Wound Roll Width in Load Increments up to Applied Load.....	66
47.	Incremental Deformations across Gapped Wound Roll with Instability..	67
48.	Incremental Deformations at 6 inch CMD Locations for Gapped Wound Roll with Instability.....	68
49.	Incremental Deformations across Gapped Wound Roll with Convergence Induced Stability.....	69
50.	Deformations across Gapped Wound Roll with Convergence Induced Stability at Final Load Steps.....	69
51.	Load per Width across the Wound Roll Width for Increasing Load.....	70

NOMENCLATURE

A	(m ² , in ²), cross sectional area
a	(m, in.), Hertzian half width of contact
CMD	Cross Machine Direction
DOF	Degree of Freedom
E	(Pa, psi), radial modulus of elasticity
E ₁ , E ₂	(Pa, psi), radial modulus of elasticity of generic bodies 1 and 2 respectively
E _{roll}	(Pa, psi), radial modulus of elasticity for the wound roll
ESM	Element Stiffness Matrix
e	FEM element number
F	(N, lb.), applied force
FEM	Finite Element Model
FEs	Finite Elements
GSM	Global Stiffness Matrix
I ^(e)	(m ⁴ , in ⁴), moment of inertia for FEM element e
i, j	generic FEM element left and right node designators
K	(N/ m, lb./ in.), GSM, stiffness matrix of entire FEM
K ^(e)	(N/ m, lb./ in.), ESM e, stiffness matrix for FEM element e
K ⁻¹	(m/ N, in./ lb.), inverse GSM

K_1	(Pa, psi), Pfeiffer's wound roll material multiplier parameter
K_2	(m/m, in./in.), Pfeiffer's wound roll material "springiness factor"
K_3	(Pa or psi), Pfeiffer's wound roll material strain multiplier parameter
k	(N/ m, lb./ in.), Hooke's spring stiffness or spring constant
L	(m, in.), length
$L^{(e)}, L_e$	(m, in.), length for FEM element e
M	(Nm, lb.in.), applied moment
N	number of DOF
P	(N, lb., Nm, lb.in.), $N \times 1$ global load vector
$P^{(e)}$	(N, lb., Nm, lb.in.), load vector for FEM element e
P_{\max}	(Pa or psi), maximum Hertzian contact pressure
R_0	(m, in.), effective radius
r	(m, in.), radius
r_1, r_2	(m, in.), radius of generic bodies 1 and 2 respectively
u	(m, in.), radial deformation
$u^{(e)}$	(m, in, radian), displacement vector for FEM element e
W	(N/ m, lb./ in.), Hertzian load per width
x	(m, in.), displacement DOF
ϵ	(m/m, in./in.), radial strain
κ	(m^{-1} , in^{-1}), curvature
κ_1, κ_2	(m^{-1} , in^{-1}), curvature of generic bodies 1 and 2 respectively
κ_0	(m^{-1} , in^{-1}), effective curvature

λ	(N, lb., Nm, lb.in.), Lagrange multiplier reaction load
ν_1, ν_2	(unitless), Poisson's ratios of generic bodies 1 and 2 respectively
θ	(radian), rotational DOF

CHAPTER I

INTRODUCTION

WEBS AND WOUND ROLLS

Many materials like film, newspaper, and plastic packaging have long, thin, sheet like forms which require unique handling not necessary for thicker materials. For example, to store them efficiently and conveniently, the materials (referred to as webs) are often wound into rolls around a core. Such a "wound roll" is depicted in figure # 1.

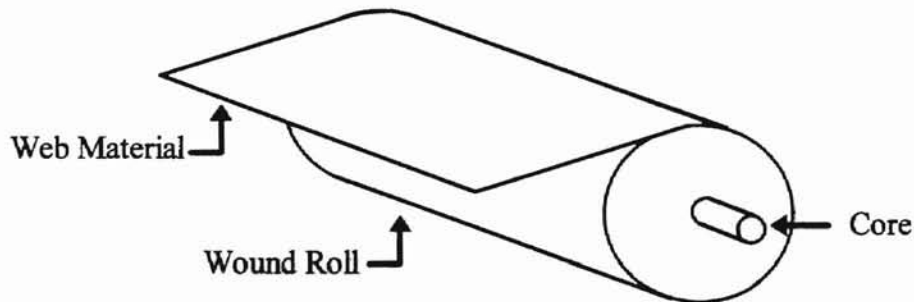


Figure # 1: Thin sheets wrapped around cores are known as wound rolls.

Web quality in the wound roll depends on the winding process. If it is wound quickly or is loose, air remains entrained between the layers. The air reduces inter-layer friction and the roll layers slide across each other in a telescoping manner. This often results in scratches on the web's surface. Conversely, tight rolls induce large inter-layer stresses which in extreme cases stretch the web locally or even cause solid adhered bands known as hard streaks. Changes in relative humidity, or storage for extended periods, may relax the wound roll or the core. This results in circumferential slip which can cause scratches or adhesion. A web which is misaligned as it winds onto the roll may even

wrinkle. Any of these conditions may render the web useless, and in fact, industry loses millions of dollars each year to useless webs. Consequently, ensuring the web's quality while wound into a roll is very important.

It is the wound roll's radial compressive stress, or pressure, versus wound roll radius which historically characterizes its quality. Low pressure corresponds to a loose web which as mentioned is susceptible to slipping type failures. Similarly, high pressure corresponds to tight webs. The actual amount of pressure depends on many factors including winding, and wound roll geometry.

WEB HANDLING AND WINDERS

Improving the quality of wound rolls is one focus of "web handling". Research goals include optimizing winding parameters, characterizing wound roll quality, developing winding models for various winders, and recommending methods to reduce web losses. Each goal contributes to the overall understanding of wound rolls.

At the heart of web handling research are two winders used heavily in industry. In center winders (figure # 2) the core rotates and thereby draws the web onto the roll. The tension in the web before it is on the roll dictates the roll's tightness. Air is easily entrained, and the overall directional control is limited. Often center winders add an

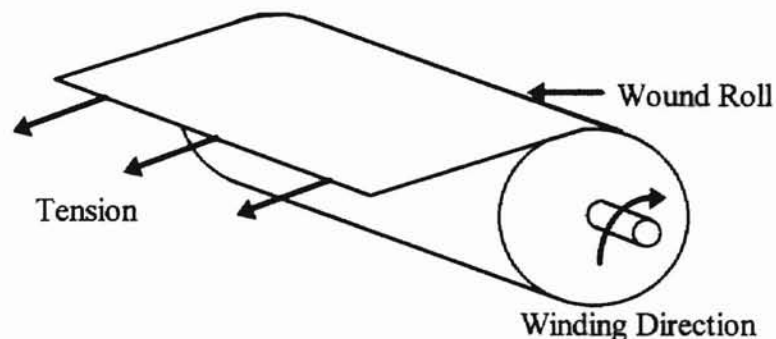


Figure # 2: Center winding involves a rotating core drawing the web onto the roll. undriven roller to the wound roll's outside edge as seen in figure # 3. This "nip roller" as it is called compresses the wound roll, offers more directional control, and also reduces entrained air. It contrasts with the driven nip rollers commonly used in the paper industry in surface winders. Instead of producing torque by rotating the core, surface winders supply all of the winding torque through the nip roller.

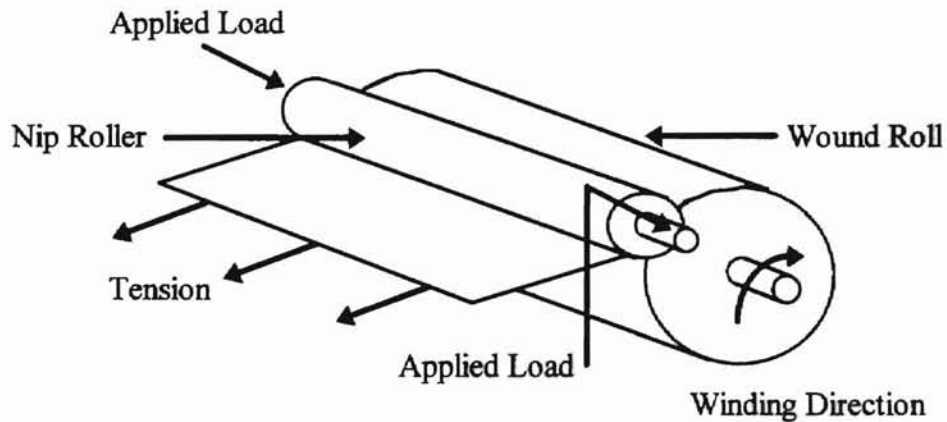


Figure # 3: Center winders apply extra loading to the wound roll through the addition of an undriven nip roller.

WEB GEOMETRY

The web's geometry affects the pressure and thereby the wound roll quality. Wide webs often have persistent thickness variations across their width. Winding the web into a roll causes an additive effect which becomes quite pronounced with increasing radius. A nip roller reduces the effect, but at the cost of increasing the local pressure. Consequently the pressure across the wound roll width will vary. Even the pressure in a uniform web

will vary across the width because most nip rollers are loaded at their edges as shown in figure # 3. The concentrated load at the edges equates to increased pressure there.

MODELING

Often winding parameters, such as pressure, exhibit trends which suggest idealizations known as mathematical models. The models are less complex than the physical wound rolls. This promotes comparisons between wound rolls and theories regarding their behavior. Also, the models often incorporate other research efforts by using their models as building blocks. This advances web handling research as a whole.

Of course, web handling research's great industrial value stems largely from using its models in making web handling recommendations. While numerous, accurate, center winding models exist from which to make recommendations, applying and developing models for wide webs is just beginning.

OBJECTIVE

The main objective in characterizing a nip impinged wound roll is establishing the loading between the nip roller and the wound roll across the entire wound roll width. The loading not only includes the contact loads, but also the deformation, and the contact pressures. In addition, the characterization focuses on variables such as geometry and material. Ultimately, the results will aid in further research and development of both center and surface winding models concentrating on wound roll quality across the entire width.

CHAPTER II

LITERATURE SURVEY

CENTER WINDING MODELS

As mentioned, many models exist which address center winding. For the most part, however, they do not take into account the width of the web and its influences on pressure and deformation.

Hakiel [1] proposes a nonlinear model which "winds" a web to find its stresses. The model utilizes strain compatibility in conjunction with hoop stress to determine the incremental pressure caused by winding one lap onto a roll. Applying central difference approximations gives the effect that each lap has on every lap before it. Winding a lap, calculating the resulting stresses, and then repeating this for each additional lap ultimately produces the wound roll. The model does not take into account the width of a wound roll, or compressibility in the radial direction.

In Hakiel [2], the model expands to include the wound roll width. The actual focus of the model is on thickness variations across the width. The model attempts to link widthwise segments together by defining a relaxation radius at which each lap undergoes no strain. Then, each segment's strain behavior is determined based on its radius with respect to the relaxation radius. Using the model presented in [1], the stresses in each segment are determined. The result is stresses in a wound roll as a function of width. The model however still assumes no radial compression.

In Hakiel [3] radial compression and wound roll width are brought together. Each lap addition is analyzed for radial deformation, and the strain is adjusted to compensate. The result is a fairly comprehensive nonlinear center winding model which accounts for thickness variations across the width.

ADDING A NIP ROLLER

Pfeiffer addresses the addition of a nip in [4]. In it, the nip roller provides a straining mechanism by inserting additional strain into the web. The strain increases the wound roll pressure locally under the nip roller. The result is a pressure wave which travels around the wound roll as it winds. An experimental data curve fit produces a relationship between the wound roll pressure and its radial strain. The expression is a refinement of a previous two parameter expression also relating the wound roll pressure to strain. The two and three parameter expressions are listed later as equations # 1 and # 4 respectively. Pfeiffer's relations are designed to generally describe material behavior. They do not address the width of the wound roll, or how the loads distribute across it.

Good [5] addresses the stresses induced by a rolling nip. The nip to wound roll contact was modeled as a Hertzian contact with a moving elliptical pressure distribution. A foundation of two dimensional plain strain finite elements represented the wound roll. The results pointed towards a theoretical nip-induced tension mechanism. The mechanism results compared well with experimental data. The analysis did not address load variations across the width, but it validated the use of finite elements and Hertzian contact formulas.

Rodal [6] addresses the use of Hertzian contact formulas for a nip contacting a wound roll. Like [5], the web is represented by plain strain finite elements. The web is

then subjected to calendaring over a range of loads. The resulting stress distribution in the web is shown to be nearly Hertzian, and differs only at the edge of contact. Once again, the stress distribution across the web width is not addressed.

Vaidyanathan [7] addresses the validity of Hertzian contact formulas applied to nip impinged wound rolls. Half widths of contact were generated theoretically using Hertzian equations. Then he measured the half widths of contact on narrow wound rolls impinged by a nip roller. The experimental values agreed well with the theory. As mentioned, the comparison was only for narrow wound rolls.

In general, models examining the loading across the width of a wound roll exist, but only for center wound rolls. Other models address the addition of a nip roller, but do not account for wound roll width. Consequently, the need exists for a model which addresses both the width of a wound roll and the effect of a nip roller. This model will perpetuate investigations into the wound on tension mechanisms associated with the nip roller for wound rolls with widthwise varying loads.

CHAPTER III

DEVELOPMENT OF A THEORETICAL MODEL

The theoretical model establishes a web material's behavior during general cylindrical contact. While Pfeiffer's two and three parameter models characterize each material's inherent stress versus strain behavior, they do not accommodate the different geometries of the contacting surfaces. But, combining them with the Hertzian cylindrical contact theories produces a geometry-specific behavior model. The model's parameters (hereby known as material behavior parameters) summarize the wound roll's behavior dependent only on the material and the contact area.

The theoretical model also correlates a material's behavior with a specific applied nip load. The model steps up through nip load increments to the applied load. A finite element model of the wound roll/ nip roller combination computes the resulting deformations at each nip step. Inserting these deformations into the material's behavior equations produces a contact profile of the material's behavior under the specified nip loading.

PFEIFFER'S STRESS VERSUS STRAIN RELATIONS

Determining the compressive stress versus strain relationship is essential in modeling wound roll behavior. One model already mentioned is Pfeiffer's [4] stress versus strain data curve fit. Shown in equation # 1, the model gives the stress (in the form of pressure, P) as a function of two parameters and the strain, ϵ (m/m, in./in.).

$$P = -K_1 + K_1 \exp(K_2 * \epsilon) \quad 1)$$

K_1 (Pa, psi) is referred to as the multiplier parameter, while K_2 (unitless) is the "springiness factor". Setting the strain ϵ equivalent to the deformation u (m, in.), per unit length l (where the unit length must have the same units as the deformation), as shown in equation # 2,

$$\epsilon = \frac{u}{l} \quad 2)$$

and substituting this into equation # 1 yields equation # 3.

$$P = -K_1 + K_1 \exp(K_2 * u) \quad 3)$$

Pfeiffer notes, however, this model has up to 9% error. To reduce the error he proposes the model listed in [4] with an additional strain multiplier parameter K_3 (Pa, PSI) as shown in equation # 4.

$$P = -K_1 + K_1 \exp(K_2 * u) + K_3 u \quad 4)$$

Here again equation # 2 is applied.

Also of great importance, is the pressure change versus deformation. This relation comes from taking the derivative of equations # 1, and # 4 with respect to the deformation. The result is the compressive modulus of elasticity, E (Pa, PSI) as shown for the two and three parameters models respectively in equations # 5, and # 6.

$$\frac{dP}{du} = E(K_1, K_2) = K_1 K_2 \exp(K_2 * u) \quad 5)$$

$$\frac{dP}{du} = E(K_1, K_2, K_3) = K_1 K_2 \exp(K_2 * u) + K_3 \quad 6)$$

Using Pfeiffer's models provides an enormous advantage. The parameters K_1 , K_2 , and K_3 , are material unique parameters already available for a number of different materials. Thus, the model provides both a practical method of comparing wound rolls, and an easy way to incorporate material tendencies into other models. In addition, if K_3 is not known, it can be set to zero and models based on equation # 4 default to equation # 1.

HERTZIAN CONTACT

While Pfeiffer's models relate wound roll pressure and strain, Hertzian contact theories address their production. In his theories, Hertz (adopted from [8]) identifies that the geometry of two elastic bodies in contact plays a critical role in their resulting deformations. One very important geometrical parameter is curvature. Curvature, κ (m^{-1} , in^{-1}), is best defined as the inverse of radius r (m , in), as shown in equation # 7.

$$\kappa = \frac{1}{r} \quad 7)$$

A flat object would have an infinite radius and zero curvature, and a small sphere would have a small radius with a large curvature.

For two contacting bodies, the combination of their curvatures directly affects the contact area. The combination is referred to as the effective radius R_0 (m , in .) and is defined in equation # 8,

$$R_0 = \kappa_0^{-1} = (\kappa_1 + \kappa_2)^{-1} = \left(\frac{1}{r_1} + \frac{1}{r_2} \right)^{-1} \quad 8)$$

where κ_1 (m^{-1} , in^{-1}) and κ_2 (m^{-1} , in^{-1}) are the curvatures of the two bodies, r_1 (m , in .) and r_2 (m , in .) are their respective radii, and κ_0 (m^{-1} , in^{-1}) is the effective curvature. If R_0 is infinite, as for two totally flat surfaces, the area of contact will not change with the loading. As a result, the area in the pressure formula ($P = F/ A$) becomes a constant and pressure only depends on applied load. For small R_0 , the area of contact will vary rapidly with the applied load. Therefore, R_0 essentially measures the rate the contact area changes

under loading. Note that if one radius is much greater than the other, the problem simplifies to the smaller curved surface contacting a flat surface. In this case only the curved surface radius matters. This is what essentially happens at the outside of a large wound roll.

Another factor affecting the reaction of bodies in elastic contact is the actual deformation the bodies undergo. If the loads forcing the bodies together are small, the bodies contact lightly, the deformation is minimum, and the area of contact is small. Conversely large loads cause more deformation and a larger contact area.

Hertz quantified contact area as a function of effective radius and deformation for various geometries including two cylinders. A nip contacting a wound roll has the geometry of two cylinders with their axis parallel as shown in figure # 4.

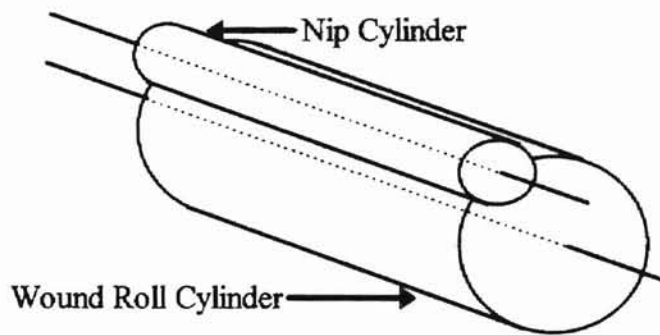


Figure # 4: Hertz explored the contact between two cylinders with parallel axis.

A cross section or end on close-up view of figure # 4 appears in Figure # 5.

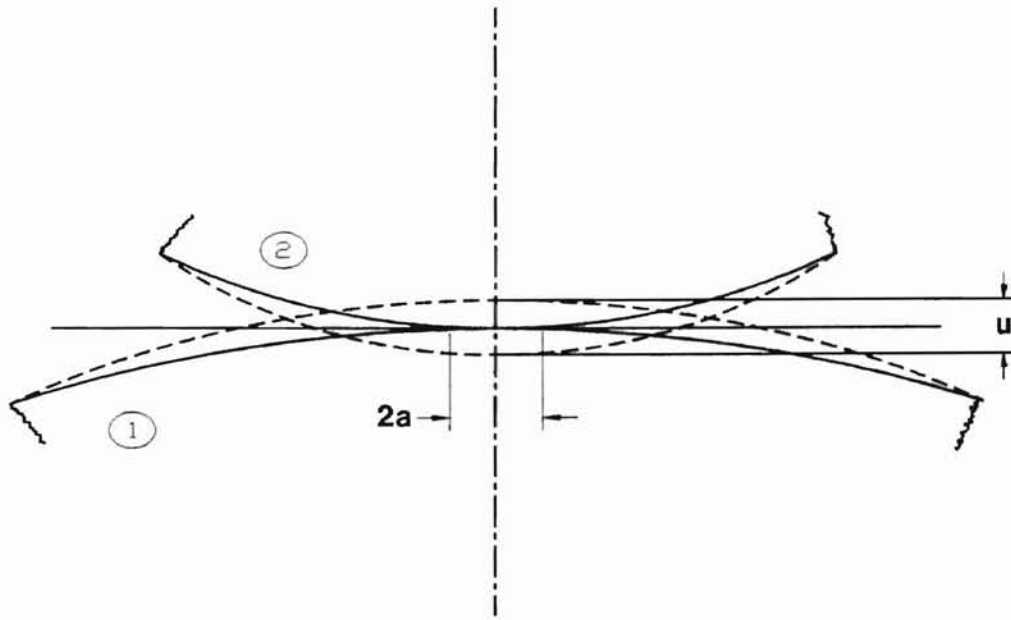


Figure # 5: The cross sectional view of two cylinders in contact shows the width of contact and the overall deformation.

As seen in figure # 5, Hertz defined the parameter a (m, in.) as half the cross section's width of contact. Also, Hertz defined the half width of contact's dependence on both R_0 and deformation u through Equation # 9 (adopted from [8]).

$$a = \sqrt{4R_0u} \quad 9)$$

Again, a is the half width of contact, R_0 the effective radius, and u the deformation. The half width of contact measures the cross sectional contact area between two bodies and is only defined when it is much less than either of the two radii.

The cylinder cross sections, with their local deformation and half width of contact, define one two-dimensional plane. In fact, Hertz defined them noting they applied to a condition of plane strain. However, the nip impingement of a wound roll is a three-dimensional, plane stress, contact. Hertz's equations remain valid for any cross section of

the wound roll because most wound rolls have a Poisson's ratio of nearly zero. This reduces the plane stress condition to plane strain.

To apply Hertz's equations to the wound roll width, the half width and deformation must be made functions of position across the width. The functions must maintain some sort of continuity across the width. Likewise, the load which causes the deformations should be continuous and distribute out across the width. The continuity will be discussed later.

To define half width and deformation across the wound roll width requires knowledge of the load across the width. Hertz addresses the load distribution across a cross section's width in a load per width equation, W (N/ m, lb./ in.) shown in equation # 10 (adopted from [8]).

$$\frac{\text{load}}{\text{width}} = W = \frac{\pi * a^2}{4R_0 \left(\frac{1 - \nu_1^2}{E_1} + \frac{1 - \nu_2^2}{E_2} \right)} \quad 10)$$

In this equation, ν_1 and E_1 are the Poisson's ratio and Modulus of elasticity of the wound roll, and ν_2 and E_2 are the nip's Poisson's ratio and Modulus of elasticity. If ν and E for both cylinders are invariant with roll width, W across the width will depend only on the deformation at each location. If the contact is uniform across the width the half width of contact and equivalently, the load per width, are also invariant.

Another parameter, the contact pressure P , varies across both the width and the cross section. The pressure variance with width results from non-uniform conditions, while the variance with cross section is due to the geometry. Hertz showed that the cross

sectional pressure has a semi-elliptical distribution, with the maximum pressure at the center of the contact area [9]. The maximum pressure is P_{\max} (Pa, psi) as given in equation # 11.

$$P_{\max} = \frac{2W}{\pi * a} \quad 11)$$

The Hertzian contact analysis is intended for two solid cylinders. This is because the elliptical pressure distribution penetrates radially into the cylinders. The depth of the pressure's penetration is small however. From [8], the pressure drops below 20% at a depth of $2*a$ below the surface. As noted from [7], the predicted Hertzian half widths of contact are experimentally verified to be very small even for larger wound roll radii. Consequently, the hollow wound roll core and hollow nip will not impact the results. This is under the assumption that the stiffness of the wound roll core and the nip are both greater than that of the wound roll material itself.

FINITE ELEMENTS

In order to use the Hertzian equations, either a , u , or the load must be known. As mentioned, under uniform loading they do not vary. However for a nip roller impinging a wound roll, the loading is non-uniform. The result is the half width, deformation, and load vary across the roll width.

Good, in [5], successfully uses two dimensional Finite Elements (FEs) to model a narrow wound roll impinged by a nip roller. The results closely match experimental values, which suggests that FEs adequately model wound roll/ nip roller contact.

Finite Elements also are an excellent tool for determining contact variables as they vary in the third dimension (roll width). In linear problems with one Degree of Freedom (DOF), Finite Element Models (FEMs) follow Hooke's spring law (equation # 12).

$$F = k x \quad 12)$$

Here, F (N, lb.) is the applied load in the direction of x (m, in.), the displacement DOF. The term k (N/ m, lb./ in.) is the spring stiffness, or spring constant. If the object undergoing F is an axially loaded rod, k is effectively equation # 13.

$$k = \frac{EA}{L} \quad 13)$$

E (Pa, psi) is the elastic modulus, A (m^2 , in^2) is the cross sectional area and L (m, in.) is the rod's length. In FEM with N DOF, equation # 12 becomes a global expression with the form of equation # 14.

$$\bar{P} = \bar{K}\bar{u} \quad 14)$$

In this form the size of the applied load vector, \mathbf{P} (N, lb.) is $N \times 1$, \mathbf{K} (N/ m, lb./ in.) is an $N \times N$ Global Stiffness Matrix (GSM), and \mathbf{u} (m, in.) is the $N \times 1$ displacement vector.

Equation # 14 states for a known \mathbf{K} , any given displacement \mathbf{u} requires the applied loading \mathbf{P} . The result of pre-multiplying both sides of the equation by \mathbf{K}^{-1} yields equation # 15, the displacement in terms of the applied force.

$$\bar{\mathbf{u}} = \bar{\mathbf{K}}^{-1} \bar{\mathbf{P}} \quad 15)$$

Here, \mathbf{K}^{-1} is defined as the inverse of \mathbf{K} , where $\mathbf{K}^{-1}\mathbf{K}=\mathbf{I}$.

Individual elements in a FEM have an inherent Element Stiffness Matrix (ESM) $\mathbf{K}^{(e)}$, load vector $\mathbf{P}^{(e)}$, and displacement vector $\mathbf{u}^{(e)}$. $\mathbf{K}^{(e)}$, $\mathbf{P}^{(e)}$, and $\mathbf{u}^{(e)}$ also relate through equation # 15, which becomes equation # 16.

$$\bar{\mathbf{u}}^{(e)} = \bar{\mathbf{K}}^{(e)-1} \bar{\mathbf{P}}^{(e)} \quad 16)$$

The components of the load and displacement vectors, and the stiffness matrix depend on the type of element used. Some common element types are the bar, beam, plate, and Winkler foundation elements.

A beam element, like the one in Figure # 6, appropriately models a nip roller. It is naturally constrained in the X, and Z directions, accommodates the geometrical bending

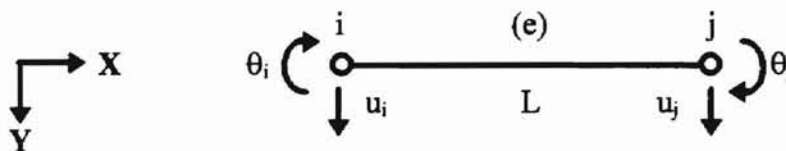


Figure # 6: A typical beam element contains two nodes with two degrees of freedom each.

(MOI) and material (E) stiffnesses, and maintains deformation and rotation continuity from element to element. Thus it reacts just like a nip roller. The letters i and j refer to the left and right node of a general element respectively. The element number is e and is denoted in parenthesis. The element's length is given by L . The coordinate frame defines the direction of positive X and Y . Normally Y is defined positive upward, but wound roll geometry lends itself to this convention. Each node has two DOF, a displacement u , and a rotation θ . The displacement DOF is the node's Y displacement. The rotation DOF is defined as the element's slope at the node. A positive θ corresponds to an increasing displacement in X . Equation # 17 shows this explicitly.

$$\theta = \frac{du}{dx} \quad (17)$$

Corresponding to each DOF is a load component. For the beam element, loads include both forces (F) and moments (M) as depicted in figure # 7. All of the DOF for an element make up the displacement vector $\mathbf{u}^{(e)}$. Likewise, all the loads for an element make up its load vector $\mathbf{P}^{(e)}$. Equation # 18 shows $\mathbf{u}^{(e)}$ and $\mathbf{P}^{(e)}$ in component form for a beam element.

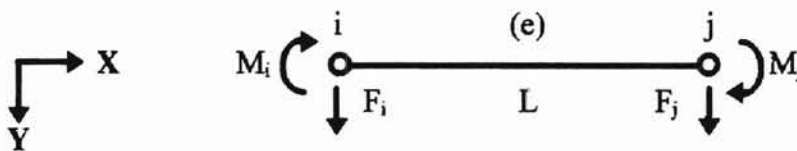


Figure # 7: Every degree of freedom in a beam element has a corresponding load.

$$\bar{\mathbf{u}}^{(e)} = \begin{bmatrix} u_i \\ \theta_i \\ u_j \\ \theta_j \end{bmatrix} \quad \bar{\mathbf{P}}^{(e)} = \begin{bmatrix} F_i \\ M_i \\ F_j \\ M_j \end{bmatrix} \quad (18)$$

Finally, the ESM $\mathbf{K}^{(e)}$ is the mapping of $\mathbf{u}^{(e)}$ to $\mathbf{P}^{(e)}$ as shown in equation # 19.

$$\bar{\mathbf{P}}^{(e)} = \begin{bmatrix} F_i \\ M_i \\ F_j \\ M_j \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K_{21} & K_{22} & K_{23} & K_{24} \\ K_{31} & K_{32} & K_{33} & K_{34} \\ K_{41} & K_{42} & K_{43} & K_{44} \end{bmatrix} \begin{bmatrix} u_i \\ \theta_i \\ u_j \\ \theta_j \end{bmatrix} = \bar{\mathbf{K}}^{(e)} \bar{\mathbf{u}}^{(e)} \quad (19)$$

Equation # 20 gives $\mathbf{K}^{(e)}$'s components for a beam element in terms of the element's length $L^{(e)}$, elastic modulus $E^{(e)}$, and its moment of inertia, $I^{(e)}$. [10] It is a symmetric matrix.

$$\bar{\mathbf{K}}^{(e)} = \frac{E^{(e)} I^{(e)}}{L^{(e)}} \begin{bmatrix} 12 & 6L^{(e)} & -12 & 6L^{(e)} \\ \cdot & 4(L^{(e)})^2 & -6L^{(e)} & 2(L^{(e)})^2 \\ \cdot & \cdot & 12 & -6L^{(e)} \\ \cdot & \cdot & \cdot & 4(L^{(e)})^2 \end{bmatrix} \quad (20)$$

A Winkler Foundation element, like the one in figure # 8, appropriately models a wound roll. Its derivation sums the potential energy associated with the foundation stiffness per length into the total potential energy formulation. The result is an element with beam-like DOF, but also stiffness dependent on the wound roll material stiffness.

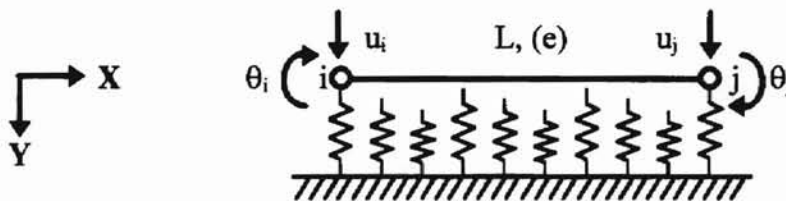


Figure # 8: A typical Winkler element contains two nodes with two degrees of freedom each and an inherent "spring" foundation.

The element's load vector \mathbf{P} and displacement vector \mathbf{u} match the beam element, but the

stiffness matrix components are different. The springs in figure # 8 aid in visualizing this difference. They give the feeling of a continuous foundation beneath the element. In addition the different lengths indicate the foundation stiffness can be non-linear in nature. Equation # 21 is the Winkler Foundation's ESM from [11].

$$\bar{K}^{(e)} = \frac{\text{Stiffness} * L^{(e)}}{420} \begin{bmatrix} 156 & 22L^{(e)} & 54 & -13L^{(e)} \\ \cdot & 4(L^{(e)})^2 & 13L^{(e)} & -3(L^{(e)})^2 \\ \cdot & \cdot & 156 & -22L^{(e)} \\ \cdot & \cdot & \cdot & 4(L^{(e)})^2 \end{bmatrix} \quad 21)$$

A global FEM essentially connects multiple elements together into a pattern that represents the geometry of a physical system. The elements connect together at their nodes to form the pattern. At the node connections, the DOF from one element dissolve into those from the other element. This makes them indistinguishable. This way, the global FEM provides the continuity needed for the Hertzian values on the cross sectional planes. Mathematically the individual $\mathbf{P}^{(e)}$ and $\mathbf{u}^{(e)}$ vectors and the $\mathbf{K}^{(e)}$ matrices combine. For instance, the ESM's given by equation # 20 and equation # 21 combine to form the GSM in equation # 14. The global notation for the element number changes from superscripts to subscripts (e.g. $L^{(e)}$ becomes L_e) to indicate the model is global.

A global FEM of the nip is a line of beam elements connected together. If (for illustration) two elements were used to model the nip, it would look like figure # 9.

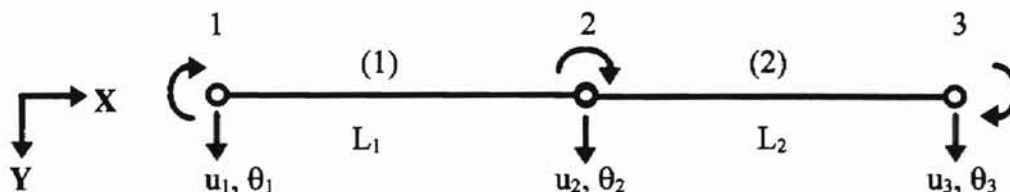


Figure # 9: The nip as a two element beam FEM.

Equation # 14 becomes the global FEM equation of the two element beam. In component form it is equation # 22. Likewise, a string of individual Winkler elements combine to produce a global FEM of the wound roll.

$$\bar{P} = \begin{bmatrix} 12 \frac{E_1 I_1}{L_1^3} & 6 \frac{E_1 I_1}{L_1^2} & -12 \frac{E_1 I_1}{L_1^3} & 6 \frac{E_1 I_1}{L_1^2} & 0 & 0 \\ \cdot & 4 \frac{E_1 I_1}{L_1} & -6 \frac{E_1 I_1}{L_1^2} & 2 \frac{E_1 I_1}{L_1} & 0 & 0 \\ \cdot & \cdot & 12 \left[\frac{E_1 I_1}{L_1^3} + \frac{E_2 I_2}{L_2^3} \right] & -6 \left[\frac{E_1 I_1}{L_1^2} + \frac{E_2 I_2}{L_2^2} \right] & -12 \frac{E_2 I_2}{L_2^3} & 6 \frac{E_2 I_2}{L_2^2} \\ \cdot & \cdot & \cdot & 4 \left[\frac{E_1 I_1}{L_1} + \frac{E_2 I_2}{L_2} \right] & -6 \frac{E_2 I_2}{L_2^2} & 2 \frac{E_2 I_2}{L_2} \\ \cdot & \cdot & \cdot & \cdot & -12 \frac{E_2 I_2}{L_2^3} & -6 \frac{E_2 I_2}{L_2^2} \\ \cdot & \cdot & \cdot & \cdot & \cdot & 4 \frac{E_2 I_2}{L_2} \end{bmatrix} \bar{u}$$

equation 22)

Connecting the individual FEMs of the beam and wound roll requires "links". A link fixes two nodes together by requiring that the corresponding DOF be the same. The method, known as Lagrange Multiplier Adjunction (Fellipa [10]), then replaces the link with a reaction load pair, $\pm\lambda$. Figure # 10 presents two elements linked at their left nodes.

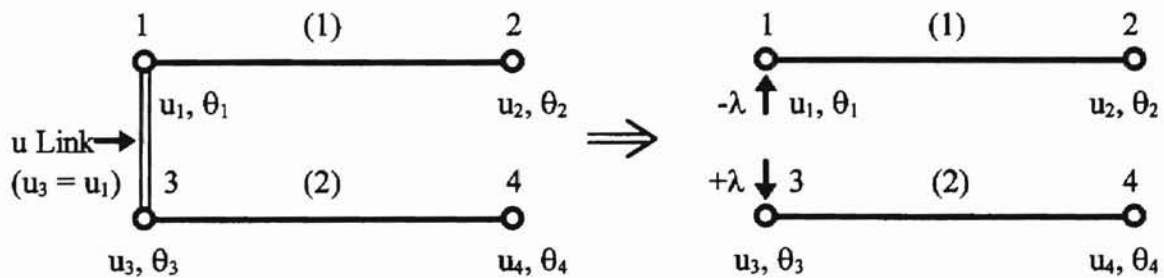


Figure # 10: Links tie together individual DOF by inserting reaction load pairs.

This is different than the previous element connection, because links affect only one DOF. That is, the link is only for the displacement (u) DOF, but another link between nodes 1 and 3 would also link the rotational (θ) DOF. The reaction load adds to the displacement vector as an unknown and the pair is set to zero in the load vector. If element 1 is a beam, element 2 a Winkler element, and the link doesn't exist, equation # 14 is equation # 23.

$$\bar{P} = \begin{bmatrix} F_1 \\ M_1 \\ F_2 \\ M_2 \\ F_3 \\ M_3 \\ F_4 \\ M_4 \end{bmatrix} = \begin{bmatrix} \bar{K}_{beam} & \bar{0} \\ \bar{0} & \bar{K}_{wink} \end{bmatrix} \begin{bmatrix} u_1 \\ \theta_1 \\ u_2 \\ \theta_2 \\ u_3 \\ \theta_3 \\ u_4 \\ \theta_4 \end{bmatrix} \quad 23)$$

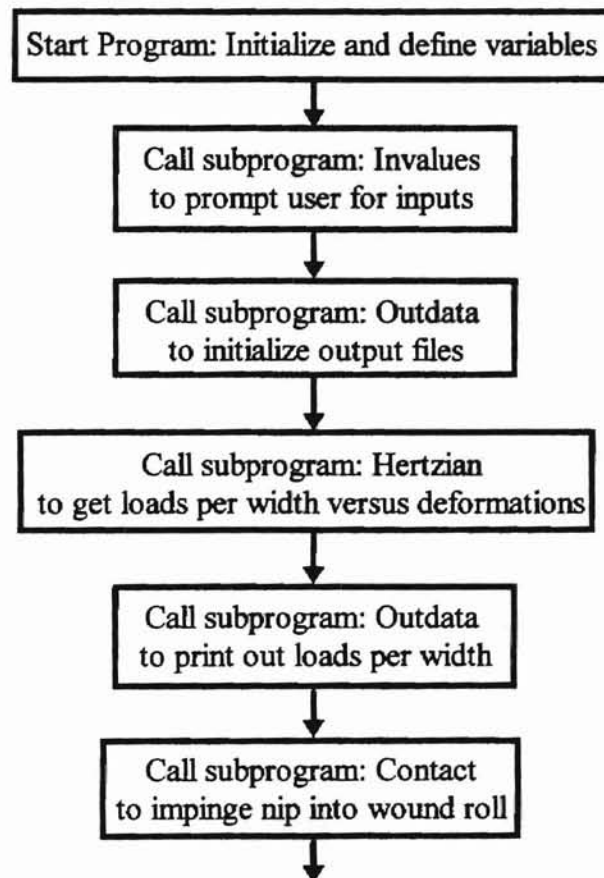
However equation # 24 shows the system with the link where $A = [1 \ 0 \ 0 \ 0]^T$.

$$\bar{P} = \begin{bmatrix} F_1 \\ M_1 \\ F_2 \\ M_2 \\ F_3 \\ M_3 \\ F_4 \\ M_4 \\ 0 \end{bmatrix} = \begin{bmatrix} \bar{K}_{beam} & \bar{0} & \bar{A} \\ \bar{0} & \bar{K}_{wink} & -\bar{A} \\ \bar{A}^T & -\bar{A}^T & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ \theta_1 \\ u_2 \\ \theta_2 \\ u_3 \\ \theta_3 \\ u_4 \\ \theta_4 \\ \lambda \end{bmatrix} \quad 24)$$

Putting links between all contacting nodes produces a global FEM which accounts for both the nip roller and the wound roll.

With the global FEM, loads input into the nip roller transmit to the wound roll. These loads produce displacements associated with specific finite element nodes. Since the nodes have fixed positions across the width, the final result is wound roll displacements as functions of position across the width. The deformations insert into the Hertzian equations to establish a profile of the contact across the roll width.

The computer program IMPINGE applies the theoretical model to the user's specified case. During program execution the user first inputs the case variables via the subprogram INVALUES as shown in figure # 11. Next the program proceeds to determine the material behavior parameters in HERTZIAN. The CONTACT and PROFILE subprograms impinge the nip into the wound roll and determine the contact profile respectively. Values are written to files throughout by subprogram OUTDATA.



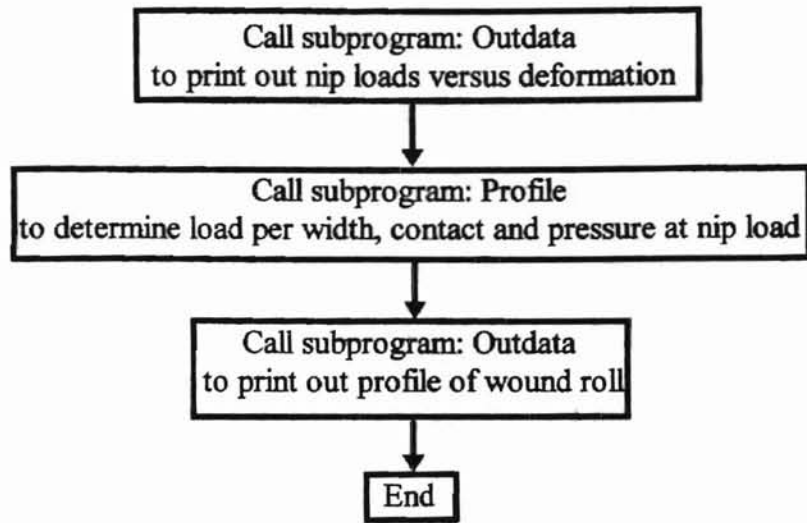


Figure # 11: IMPINGE determines general loads per width versus displacement, and the loads per width for a wound roll impinged by a nip roller.

MATERIAL BEHAVIOR PARAMETERS

The material behavior parameters establish how the wound roll material reacts to general loading. The parameters include the half width of contact, deformation, load per width and wound roll foundation stiffness. Since the other parameters all depend on the half width of contact, it is a suitable control parameter. In fact, the expression of the other parameters in terms of the half width provides an excellent means for comparing the effect web material plays on a wound roll profile.

Determining the deformation as a function of half width is straightforward. Simply solving the half width equation (equation # 9) for deformation in terms of half width yields equation # 25.

$$u = \frac{a^2}{4R_0} \quad 25)$$

The load versus width function is a bit more involved. Starting with equation # 10 and altering the subscripts for center winding with a nip roll gives equation # 26.

$$\frac{\text{load}}{\text{width}} = W = \frac{\pi a^2}{4R_0 \left[\frac{1 - \nu_{roll}^2}{E_{roll}} + \frac{1 - \nu_{nip}^2}{E_{nip}} \right]} \quad 26)$$

Using equation # 25 to eliminate a , and leaving it in terms of u yields equation # 27.

$$\frac{\text{load}}{\text{width}} = W = \frac{\pi u}{\left[\frac{1 - \nu_{roll}^2}{E_{roll}} + \frac{1 - \nu_{nip}^2}{E_{nip}} \right]} \quad 27)$$

As can be seen, W is dependent on the displacement u , the Poisson's ratio and elastic modulus of the roll, ν_{roll} and E_{roll} , and the Poisson's ratio and elastic modulus of the nip, ν_{nip} and E_{nip} . Since the nip roll is usually made from a well characterized material (such as aluminum), E and ν for the nip are known. ν_{roll} is widely accepted to be 0.01 as seen in Good [12]. E_{roll} is dependent on u however as given by Pfeiffer [4].

E_{roll} is obtained from equations # 5 or # 6 which are repeated here as equation # 28 for reference. (as always $K_3 = 0$ maintains the two parameter version of the model)

$$E_{roll} = K_1 K_2 \exp(K_2 u) + K_3 \quad 28)$$

With E_{roll} determined, the load per width (equation # 27) versus deformation, and a (equation # 9), can be calculated when u is known.

The only remaining parameter desired as a function of half width is the effective foundation stiffness. The effective stiffness should take into account the effects of the nip roller, the wound roll, and the deformation stemming from the half width of contact. Equation # 27 for the load per width provides a starting point since it already considers these parameters. Conceptually the stiffness is how the foundation handles the distributed load (load per width) at different deformations or half widths. Mathematically, it is the derivative of W with respect to u . Equation # 29 gives the result.

$$\frac{dW}{du} = \frac{d}{du} \left[\frac{\pi u}{\left[\frac{1 - \nu_{roll}^2}{E_{roll}} + \frac{1 - \nu_{nip}^2}{E_{nip}} \right]} \right]$$

$$Stiffness = \pi \left\{ \frac{\left[\frac{1 - \nu_{roll}^2}{E_{roll}} + \frac{1 - \nu_{nip}^2}{E_{nip}} \right] + \left[\frac{u(1 - \nu_{roll}^2) K_1 K_2^2 \exp(K_2 u)}{E_{roll}^2} \right]}{\left[\frac{1 - \nu_{roll}^2}{E_{roll}} + \frac{1 - \nu_{nip}^2}{E_{nip}} \right]^2} \right\} \quad 29)$$

The HERTZIAN subprogram produces an overview sheet of the material behavior parameters. Figure # 12 shows the subprogram's flow.

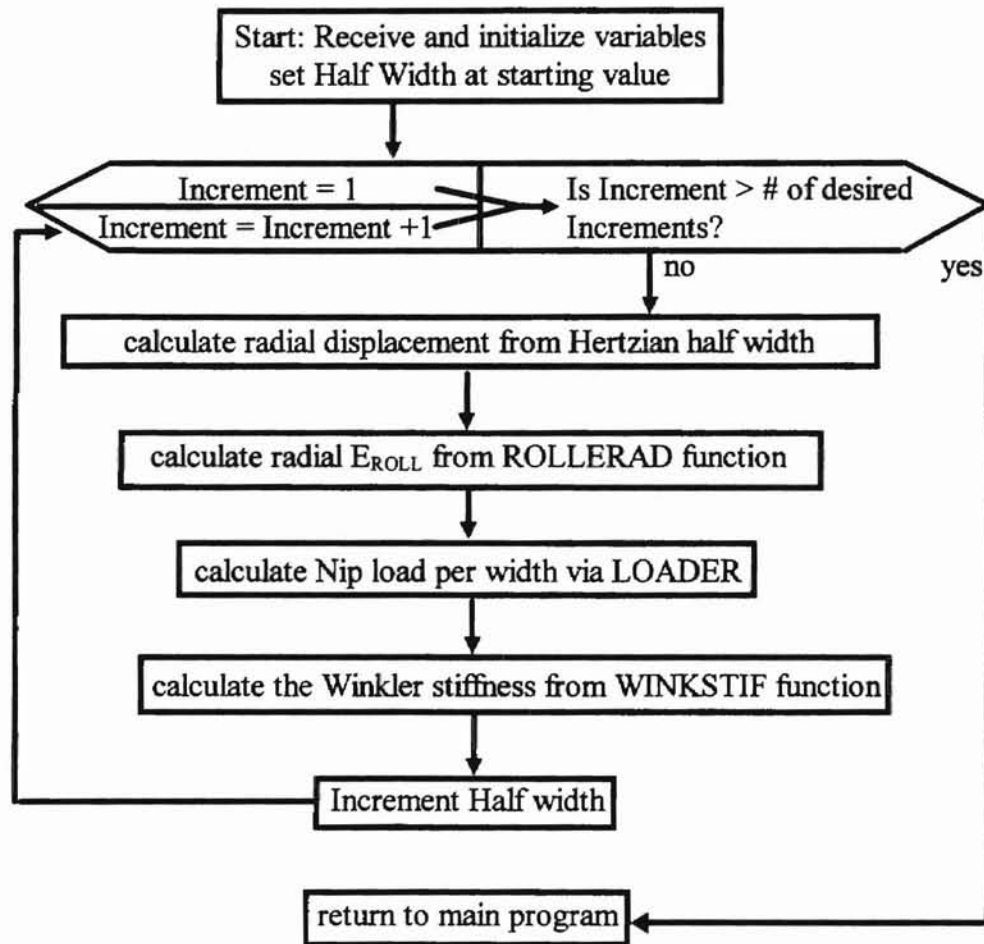


Figure # 12: The HERTZIAN subprogram increments the half width of contact and calculates the material's resulting behavior.

An incrementing loop steps through the desired range of half widths. Equation # 25 provides the corresponding deformation at each half width increment. Next the subprogram ROLLERAD calculates the wound roll's radial modulus. This value along with the deformation is sent to the LOADER and WINKSTIF subprograms. These programs determine via equations # 27 and # 29 the load per width and the roll stiffness respectively. The results are then returned to the main program to be written to the file PINGELPW.OUT.

NIP ROLL IMPINGEMENT

The theoretical model also correlates a material's behavior with a specific applied nip load by tracking the state of important locations across the web during contact. Since these locations vary with each winding geometry, they directly affect the winding behavior. For example, if a long nip roller contacts a short wound roll the results across the wound roll will differ considerably from a short nip roller contacting a long wound roll. Figure # 13 exaggerates the differences for sake of visualization.

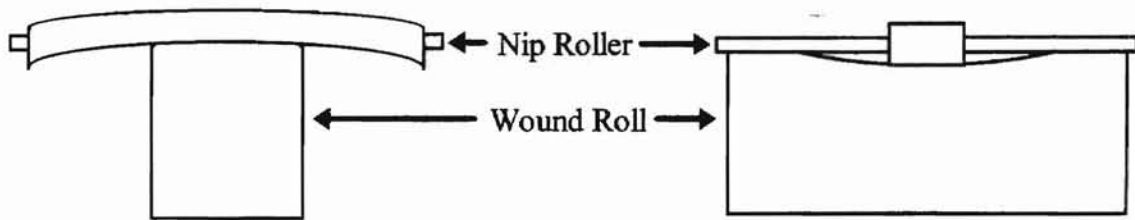


Figure # 13: These two greatly exaggerated views depict wound roll behavior as a function of geometry.

For this reason, the geometry of each physical winding setup must be input into the theoretical model.

The important locations are expressed in terms of their distance from a reference point. As shown in figure # 14, the reference point used for this model is the far left point (point A) where the winding machine applies load to the nip roller. The load (often applied by a pneumatic cylinder) is at A and also at F. Usually the load is exerted on the nip roller's stub shaft, so this is also depicted in the figure. The direction from left to right across the web's width is referred to as the Cross Machine Direction, or the CMD. Consequently the point A is referred to as CMD zero. The machine's right boundary is

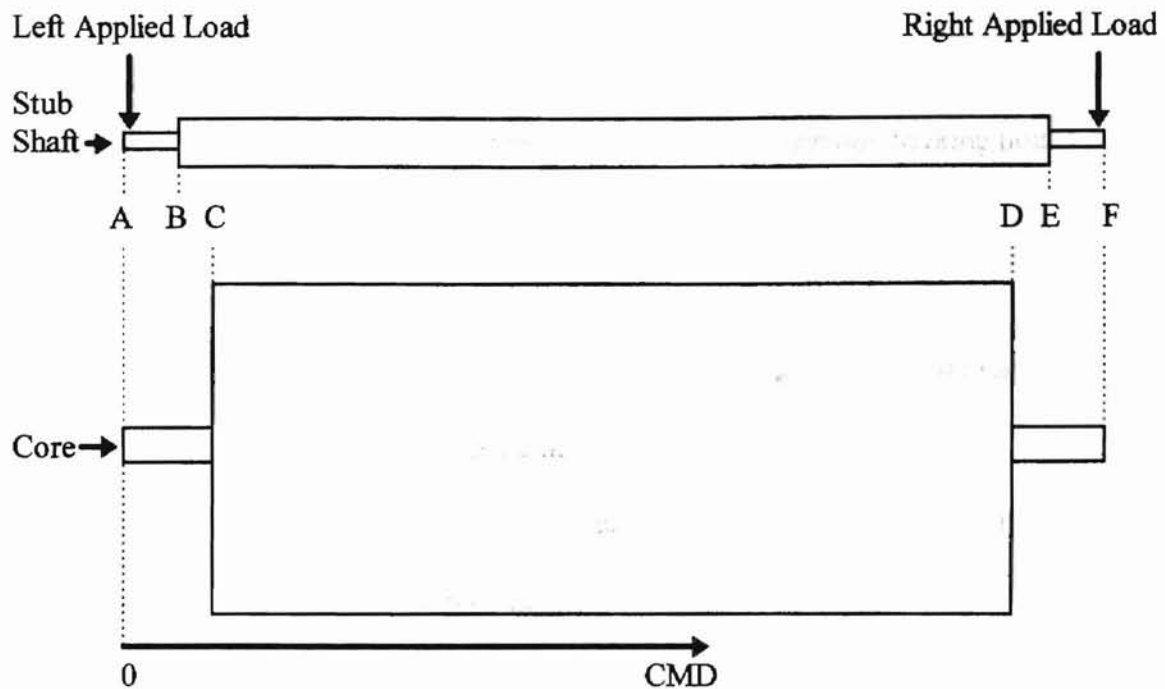


Figure # 14: The physical winding machine contains locations across the web's width which must be preserved.

defined by the far right point F. Thus, the machine's overall CMD width is referred to as the CMD distance between the applied loads (distance AF). This model allows AF to be up to 48 inches.

The change in radii of the nip and wound roll are important points to define. The CMD distance to the transition from stub shaft radius to nip roll radius (also known as the stub shaft length) is the CMD distance to the nip roller's left edge (distance AB). Likewise, the CMD distance to the transition from wound roll core radius to wound roll radius is the CMD distance to the wound roll (distance AC). The CMD distance from the nip roller's left edge to its right edge is the nip roller width (distance BE). The wound roll width is defined in the same fashion (distance CD).

The model is not restricted to the specific geometry given in figure # 14. As alluded to back in figure # 13, either of the distances AC or DF may be less than AB or EF respectively. Points B and C, and/ or points D and E can coincide. Making points A and B, and points E and F coincide (that is, $AB=0$, and $AE=AF$) eliminates the stub shafts from consideration. In general the model tolerates a wide range of geometries.

The radii are shown in the end view of the winding geometry, figure # 15. As expected they are measured from their axis of rotation. The stub shaft and Nip roll radii share the same rotation axis, as do the core and wound roll. The two axes of rotation are assumed to be parallel and remain that way throughout winding.

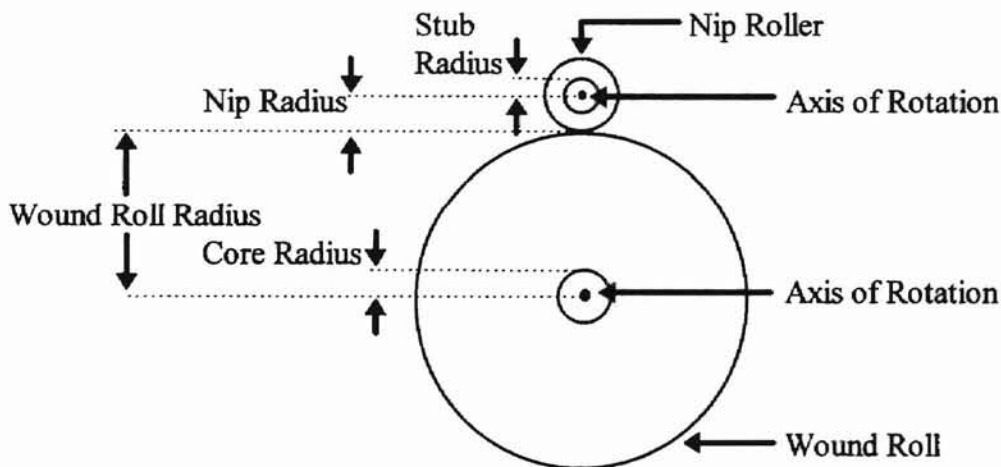


Figure # 15: Each radius is measured from the axis of rotation to the outer edge.

With the geometry established, the theoretical model converts the physical winding system into the FEM. As mentioned, the nip roller FEM is a beam with stiffness, K as given by equation # 20. The nip roll stub shafts are extensions of the nip roll beam. They have however unique stiffness values to accommodate their reduced radius and/ or different elastic moduli. Again, the Winkler foundation stiffness (equation # 21)

represents the wound roll in terms of stiffness (equation # 29). The core however is assumed to be of infinite stiffness, and therefore is represented as rigid. The FEM model of figure # 14 is given as figure # 16.

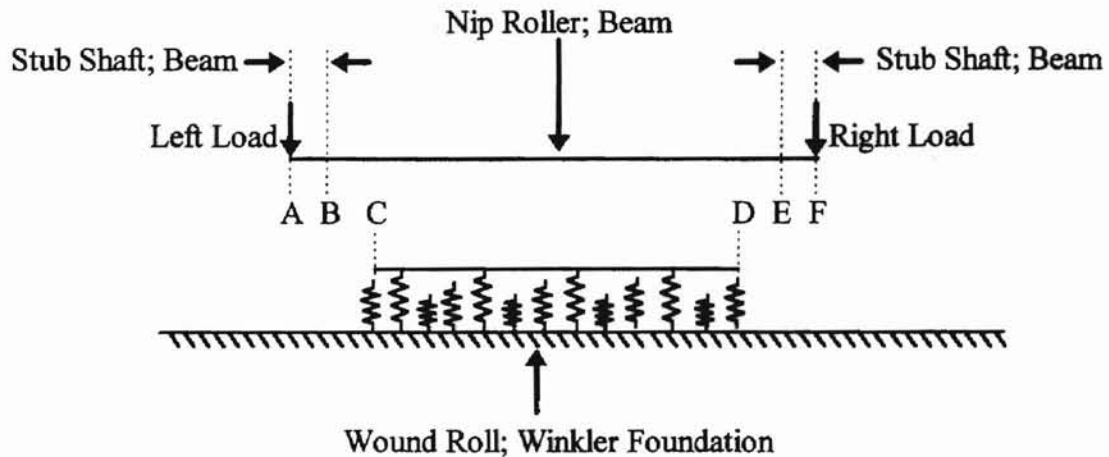


Figure # 16: The FEM impinges a beam into a Winkler Foundation.

The FEM mesh derives from the dimensions of the Winkler Foundation. Points C and D are maintained as rigid locations. Then, the desired number of evenly spaced elements are placed in between them. Next, the beam points (A, B, E, and F) are set as rigid locations with as many 1 inch long elements as possible placed in between them. If the distances AC or DF are not evenly divisible by 1 in., shorter elements fill the remaining spaces. The elements and nodes are numbered from left to right beginning with the beam and ending with the Winkler foundation. Table # 1 lists the dimensions for a simplified

<u>location</u>	<u>distance (in.)</u>
AF	5.6
AC	1.6
CD	3
AB	1.2
BE	3.4

Table # 1: The dimensions for the example FEM illustrate possible relative placements.

example geometry. Assuming 6 elements are desired across the wound roll, the resulting CMD element placement is represented pictorially in figure # 17.

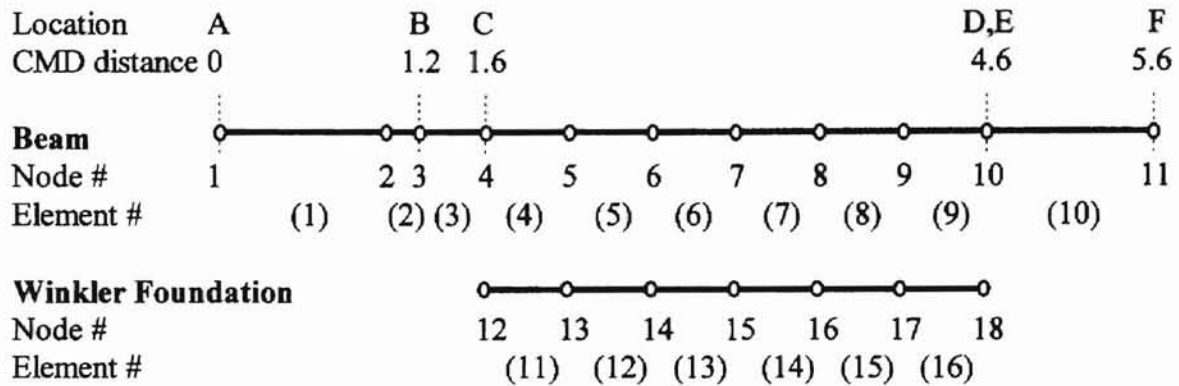


Figure # 17: The FEM mesh preserves important locations.

The 1 inch element size is arbitrarily chosen to create a suitably fine mesh for the stub shaft regions while reducing the number of total elements. Since the total number of elements across the machine's width is not allowed to exceed 96, using less elements for the stub shafts leaves more available for the wound roll.

The FE mesh establishes how each element contributes to the global stiffness matrix. Each ESM ($K^{(e)}$) inserts into the GSM (K) based on its position, type and physical attributes as demonstrated in equation # 23. Like equation # 24 shows, inserting links establishes their contact dependence. In the example, this translates into nodes 4 through 10 displacing and rotating the same as nodes 12 through 18 respectively. The resulting global stiffness matrix for the example is then a 50x50 matrix (22+14+14).

A simpler global stiffness matrix results from a method without links. Adding the individual element stiffness matrices for the elements in contact results in combination

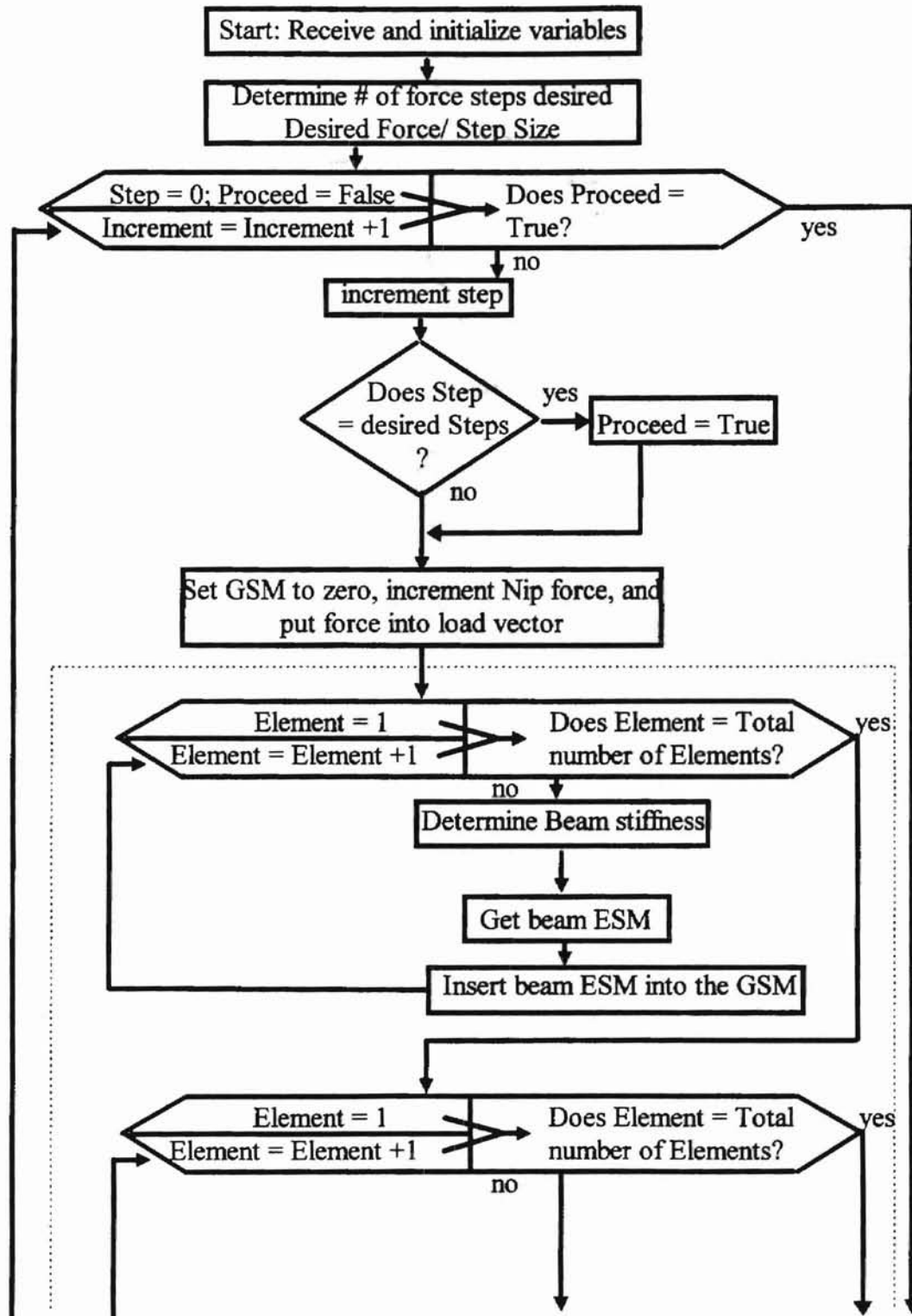
beam/ Winkler elements. This dissolves the DOF associated with nodes 12 through 18 into the ones for nodes 4 through 10. Then, the total DOF across the width is 22 and the global stiffness matrix is 22x22. Although this is an 80 % reduction in elements which would represent a considerable reduction in the calculations required for large models, the method has a drawback. It destroys the information needed to analyze gapping.

The links are essential in indicating gaps. Gaps occur when the nip roller pulls away from the wound roll (which makes the rolls independent). Since the links act to hold the nip roller and wound roll together, and they represent the reaction forces, they go into compression. To maintain equilibrium, the force they exert equals the force the nip roller is pulling away with, except that it has a negative value. Therefore, locations with negative valued links are gap locations. Eliminating those links eliminates the local nip roller-wound roll dependence and reduces the FEM contact area just like the physical gap.

Substituting the inverse of the GSM into equation # 15 yields the FEM solution. The force vector applies the specific loading which is designed to be the force at points A and F. The solution is the displacement vector of deformations and rotations for the applied nip load.

The subprogram CONTACT performs the entire FEM analysis. As shown in figure # 18, CONTACT receives the FEM mesh, the applied nip load, and other values from the main program. The applied nip force divided by the size of each force step produces the number of steps needed to fully impinge the nip. Next, the program zeroes out the stiffness matrix and inserts the first nip load into the force vector. Another loop

cycles through the elements determining the beam's stiffness, ESM, and plugs it into the GSM.



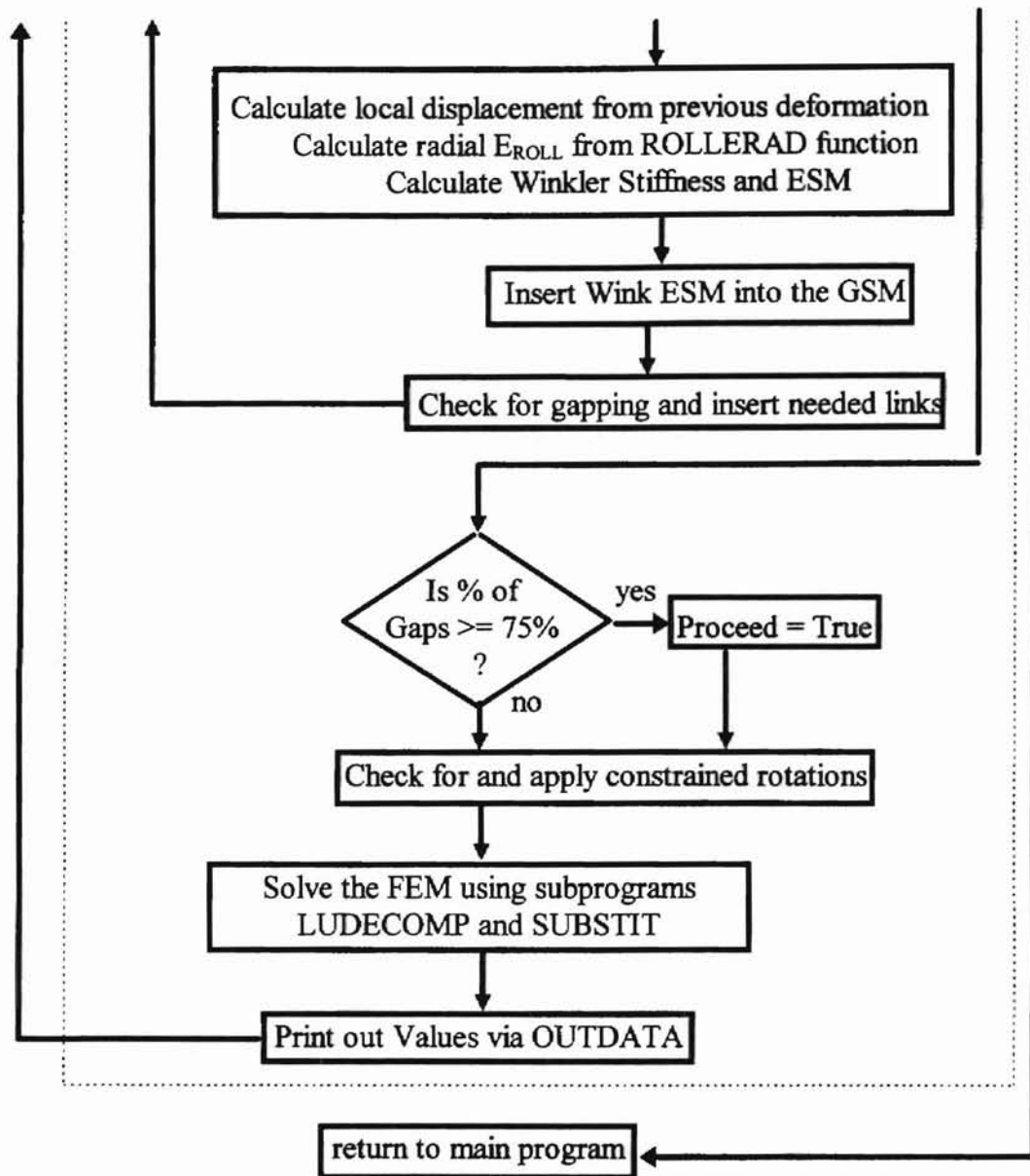


Figure # 18: The CONTACT subprogram determines the displacements in a wound roll impinged by a nip roller.

The program then loops through the elements again to address the Winkler elements. In order to use equation # 29 for the Winkler stiffness, it needs the displacement of each element. The program averages the nodal displacements for each element from the previous run's displacements. (The first time through the displacements

are zero.) The displacements feed into equation # 28's E_{roll} (subprogram ROLLERAD), and the subprogram WINKSTIF calculates the element's Winkler stiffness via equation # 29. The stiffness then inserts into subprogram WINKELEM which uses equation #21 for the Winkler ESM. This fills into the GSM as dictated by the mesh. The Winkler elements are then checked for gapping and the links are adjusted depending on the contact condition.

From here CONTACT solves the FEM. The subprogram LUDECOMP [13] breaks down the GSM into upper and lower triangular matrices which are sent to SUBSTIT [13]. This subprogram uses forward and back substitution methods to find the solution. The values write to file, and the program then returns to increment the nip load again. The process repeats incrementing the load until reaching the applied nip load, or when 75 % or more of the elements gap. After the final run CONTACT returns to the main program.

For some winding configurations the deformations become unstable. This is due to the way they are determined. As mentioned in figure # 18, CONTACT uses the previous loop's deformations to determine the new stiffness of the foundation. At the same time the load increments one step. Therefore the deformations going into the foundation stiffness calculation do not correspond to the applied load. Figure # 19 summarizes this action. The program starts at 1, proceeds to 2. Then the load is applied at 3. The deformations at 4 are thus the result of 2 and 3. These deformations feed back into 2 and the cycle repeats.

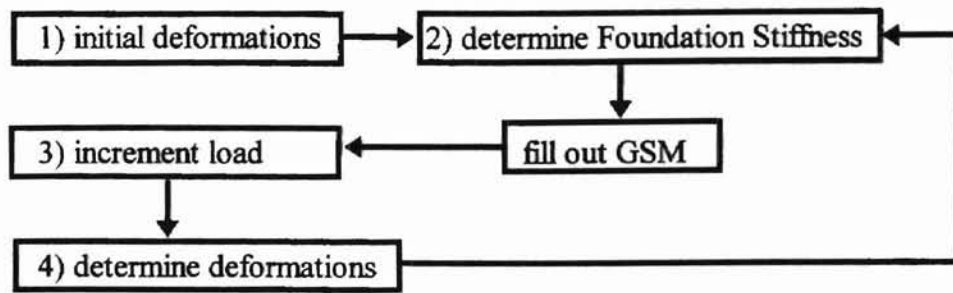


Figure # 19: The CONTACT program uses previous deformations to determine the new foundation stiffness.

As the deformations increase in magnitude the stiffness increases exponentially causing the wound roll to be extremely stiff very suddenly. The result is the FEM output deformations will drop in response to the large stiffness. As the cycle repeats the deformation values will go unstable. This is best understood from the following figure # 20. By starting at deformation (1) a low stiffness (2) results. The applied load (3)

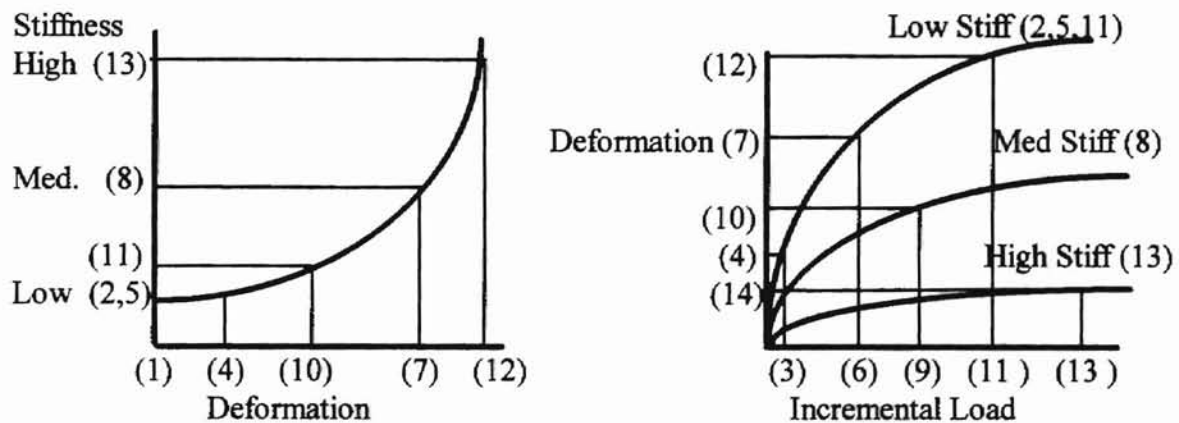


Figure # 20: The program becomes unstable at higher loads.

at stiffness level (2) produces the deformation (4). Which when plugged back into the stiffness curve produces another low stiffness (5). Incremental load (6) at stiffness level

(5) produces the much larger deformation (7). Deformation (7) produces a medium stiffness (8), which yields (at load level (9)) a smaller deformation (10)! Since the load keeps incrementing upwards, the program never has a chance to settle as seen by steps (10) through (14).

For this reason an extra convergence iteration loop was added to CONTACT. The loop iterates over the dotted area in figure # 18 until the deformations converge. This means that the deformations input into the stiffness calculations are equal (within a tolerance) to the FEM deformations at that load level. The method used is the bracketing bisection method from [14]. The convergence loop eliminates the instability, but increases computation time.

CONTACT PROFILE

The displacements, half widths of contact, and the loads and pressures per width all define the final contact profile. While the displacements are simply the FEM deformations, the other values require further calculations. The half widths of contact come immediately from equation # 9. Likewise equation # 27 with equation # 28 seemingly produces the loads per width. The load per width and E_{roll} equations however assume a constant geometry and a consistent loading across the width. Essentially, they make no distinction between local and global values. Therefore, they break down when used to determine local deformations resulting from the nip's globally varying loading.

The FEM provides a viable relationship between the deformations and the loads per width that doesn't break down locally. It inherently accounts for the local variations because its results are designed to be globally continuous. Therefore, any load or pressure

calculations should be made through the FEM. This is done by evaluating each element separately. Inserting the displacements for the element's two nodes into the element form of equation # 14 produces the element's loads. Repeating this for the next element produces its loads. Since two consecutive elements share a node, the node has loads from both elements. Summing these loads and dividing by the sum of half each element's length produces the load per width at that node. This result then properly plugs into equation # 11 for the maximum pressure. Subprogram PROFILE performs these calculations.

CHAPTER IV

RESULTS

MATERIAL BEHAVIOR PARAMETERS

Figure # 21 displays how the deformation depends on the effective radius as given by equation # 9 . As the effective radius varies in the plot, the half width of contact is held

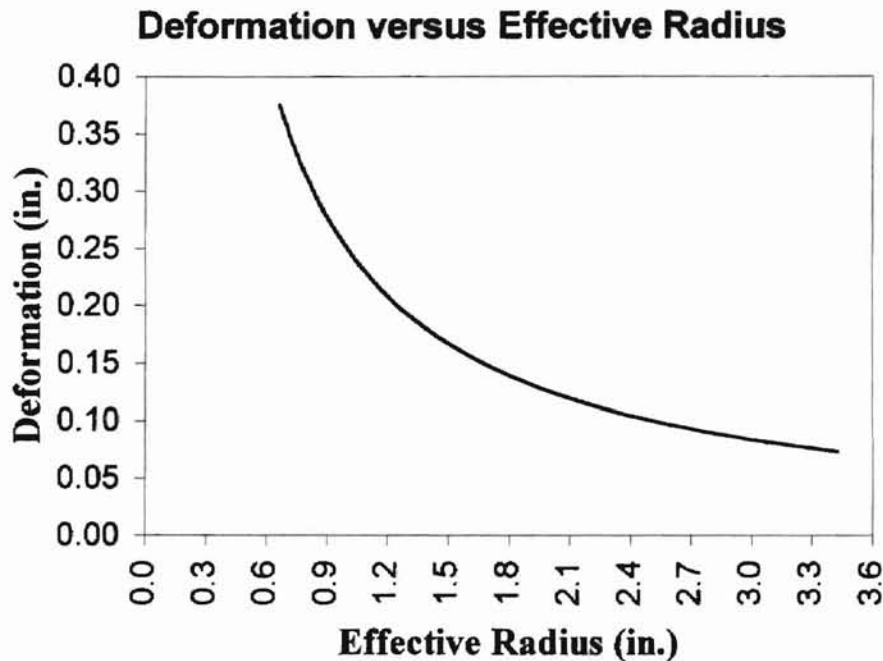


Figure # 21: The deformation of two cylinders in contact varies inversely with the effective radius.

constant. Therefore, the plot indicates in order to maintain the half width of contact, the deformation must decrease as the effective radius increases. Since the effective radius indicates a cumulative effect of the individual cylinder radii, a larger R_0 translates into larger cylinders. So, as the cylinders increase in size, less penetration produces the same half width of contact.

The IMPINGE program's output file PINGELPW.OUT (listed in Appendix A) gives the foundation's stiffness and the load per width as functions of deformation. Figure # 22 plots the values for 83 μm newsprint. The newsprint is relatively soft with a

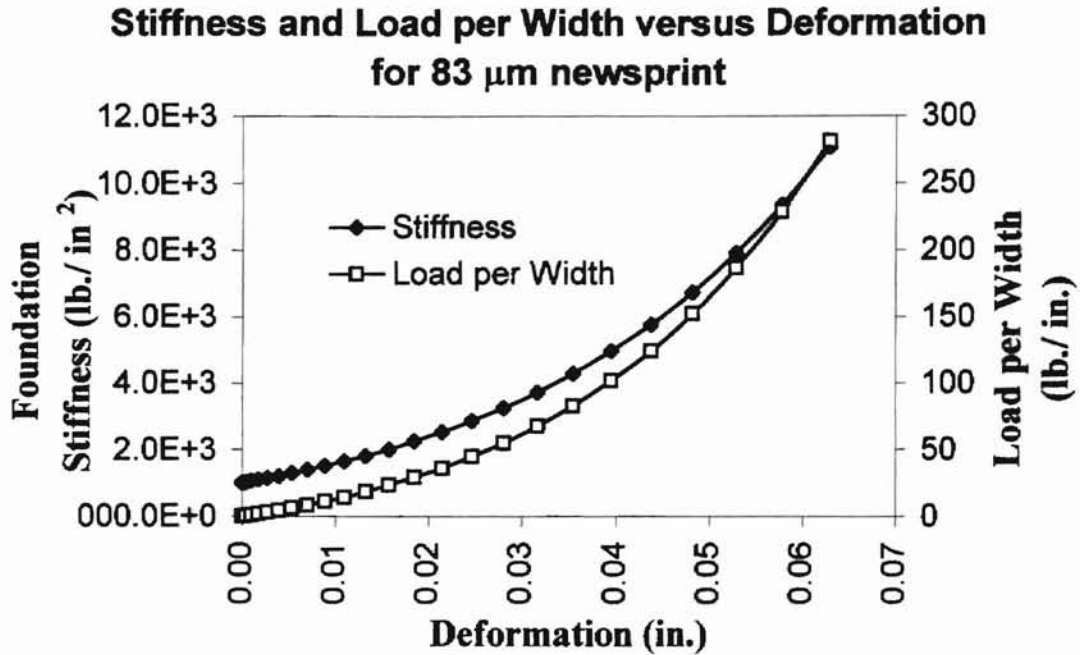


Figure # 22: Both the foundation's stiffness and load per width vary non-linearly with deformation.

Pfeiffer 2 parameter pressure representation of $P = -13.537 + 13.537 \cdot \exp(23.55u)$.

Nonetheless, the foundation stiffness reaches 10,000 lb./in² at 0.06 in. of deformation which is three times the 3400 lb./in² at 0.03 in. The load per width acts similarly going from 60 lb./in. to 250 lb./in. over the same interval. Thus, as the foundation compresses it rapidly increases in stiffness and requires increasingly more load to deform it. The same trend occurs for 25 μm polyester sheets as shown in figure # 23. Deforming from

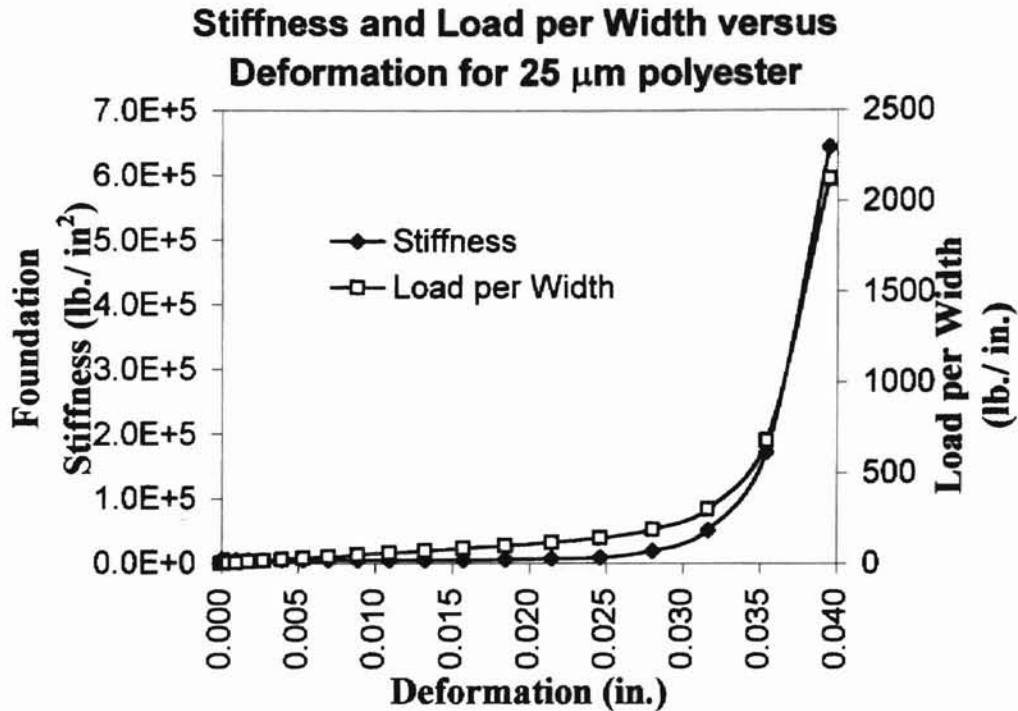


Figure # 23: Even small deformation produces large stiffness and load per width for the 25 μm polyester material.

0.02 in. to 0.04 in. produces nearly a 100 fold increase in foundation stiffness and a 20 fold increase in load per width. In comparison, the polyester is considerably stiffer than the newsprint as seen from the sharper exponential increase and the smaller corresponding deformation value. This is because Pfeiffer's 3 parameter pressure representation for the polyester ($P = -2.66\text{E-}04 + 2.66\text{E-}04*\exp(307.5*u) + 1631*u$) has a much larger K_2 value.

THE STANDARD

Proper investigation of the results from the IMPINGE program dictates the use of a standard. As there is no absolute standard for the field of web handling, any reasonable arbitrary configuration will work. This "standard configuration" provides a reference to

relate all other IMPINGE program results. Varying the configuration values separately facilitates determining their respective influences on the profile. The sum of the influences make up the overall characterization of the nip impingement.

The standard configuration appears in table # 2. The geometry, material, and

Group	Description	Standard Value
GEOMETRY		
	Nip Roller Radius	2 inches
	CMD Distance between applied load locations	36 inches
	CMD Distance from zero to Nip Roller	0 inches
	Nip roller CMD width	36 inches
	Radius of Wound Roll	5 inches
	CMD Distance from zero to Wound Roll	0 inches
	CMD width of wound roll	36 inches
MATERIAL		
	Nip Roller Modulus of Elasticity	1.03E+07 lb./ in ²
	Nip Roller Moment of Inertia	5.2 in ⁴
	Nip Roller Poisson's Ratio	0.3
	Nip Roller Stub Shaft Modulus of Elasticity	1.03E+07 lb./ in ²
	Nip Roller Stub Shaft Moment of Inertia	0.785 in ⁴
	Wound Roll Poisson's Ratio	0.01
	K1	13.537 lb./ in ²
	K2	23.55
	K3	0 lb./ in ²
LOADING		
	Applied Nip Force Load	200 lb.
MODELING		
	Nip Force Step Size	25 lb.
	Starting Nip Force	0 lb.
	Should the FEM Iterate for Deformations?	N
	Is Nip's left end constrained from Rotating?	N
	Is Nip's right end constrained from Rotating?	N
	Number of Finite Elements across the Wound Roll	36

Table # 2: The standard configuration serves as a reference for comparison.

loading groups categorize the values by type. The modeling group comprises the remaining values which control how the program models the impingement. Appendix C contains the standard configuration as a data input file (DATASAMP.IN) for IMPINGE. In general, the data input file groups are the same as those in the table with two exceptions. The data file's Hertzian contact analysis group is not listed because it provided the results for the material behavior parameters. The data file's Finite Element Analysis group contains the table's loading and modeling groups, and a decision if the program should output rotations as well as deformations. Note that this decision does not impact the results, and therefore will not be discussed further.

Some of the values in the standard configuration require more explanation. The modulus of elasticity values for the nip and the stub shaft are for aluminum. The 5.2 in⁴ MOI of the nip roller is for the 2 inch radius with a 0.25 inch wall thickness. The stub shaft's MOI is for a 1 inch radius solid rod. The remaining values are arbitrary, but are common to web handling.

The profile results from the standard configuration make up the standard profile. Appendix D contains three of the standard configuration output files from IMPINGE. The files, PINGESUM.OUT, PINGEPRO.OUT, and PINGEINC.OUT give the data used for the deformation, load per width and pressure profiles. Figure # 24 shows the deformation across the wound roll width for the standard configuration.

The loads applied to the nip are 36 inches apart corresponding to CMD zero and 36 inches respectively. Consequently, the nip bends across the width. Since the nip roller and the wound roll have the same width for the standard configuration, the deformation of

the nip matches that of the wound roll. As seen in the figure, the wound roll deformation increases towards the outer edges. So, the nip deforms more near the locations of the applied loads. This matches expectations established from classical beam bending at concentrated loads.

Deformation across the Width for the Standard Configuration

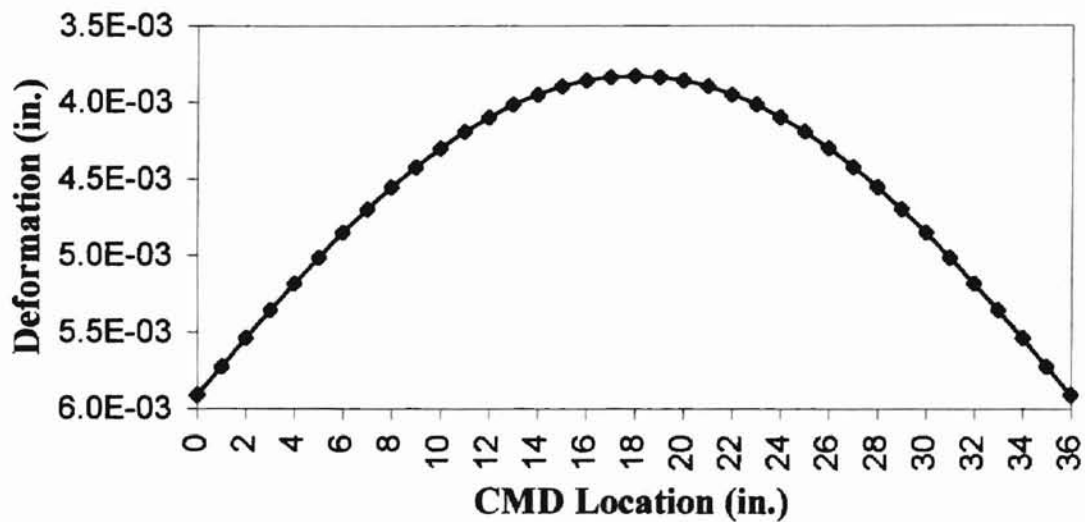


Figure # 24: The deformation across the wound roll is greater at the ends where the load is applied to the nip roller.

Figure # 25 plots the load per width across the wound roll width for the standard configuration. Again, the values at the ends are greater than in the center. The average load per width of 5.56 lb./ in. is also on figure # 25. It is the total load (200 lb.) divided by the wound roll's CMD width (36 in.). The plot shows that the load does not distribute evenly across the width. But, the sum of the loads across the roll equals 200 lb., which indicates that the solution is valid. Figure # 26 displays the maximum pressure variation

for the standard configuration. Each maximum pressure value is the Hertzian maximum pressure on the cross section corresponding to the CMD location. The maximum pressure distribution across the wound roll's width is also greater at the ends of the roll. The largest maximum pressure for the standard configuration is -25.76 lb./in^2 (compressive).

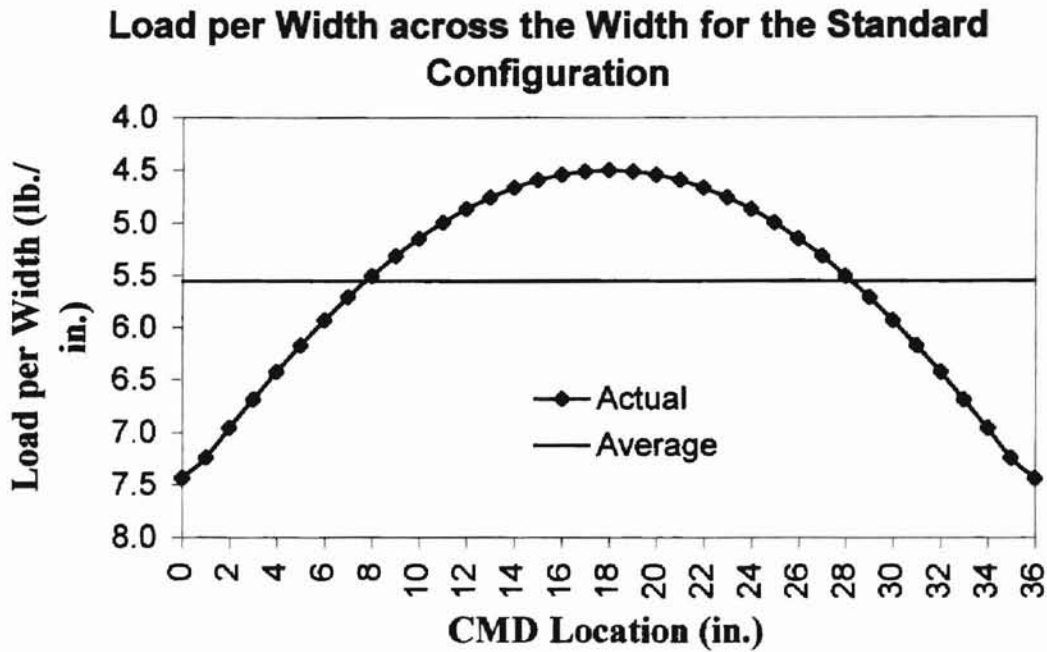


Figure # 25: The actual load per width across the web varies in comparison to the constant average value.

Maximum Pressure across the Width for the Standard Configuration

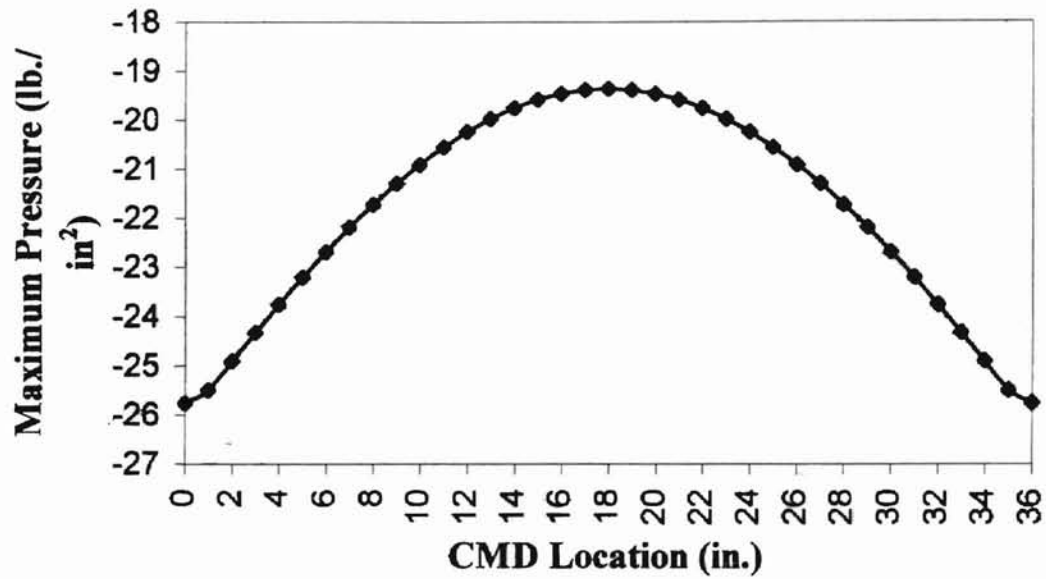


Figure # 26: The maximum pressure distributes in the same manner as the deformation and the load per width.

GEOMETRY

The wound roll impingement profile depends first on the geometry of the wound roll. The effect of the wound roll's radius is through the already mentioned effective radius. The wound roll width also affects the impingement profile. A shorter wound roll has less surface area in contact with the nip roller and should have larger profile values than the standard. The deformations across the wound roll from IMPINGE for a 24 in., 12 in., and 6 in. wound roll appear in figure # 27. The wound roll in each case is

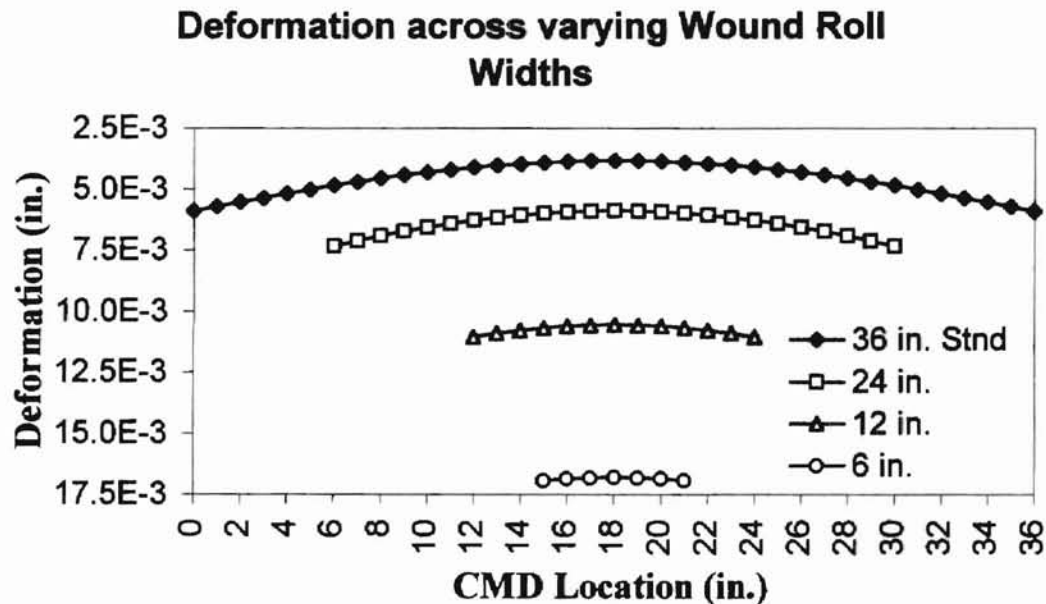


Figure # 27: The wound roll deformations increase as the width of the roll decreases.

centered under the nip roll. As expected, the shorter wound rolls deform more than the longer ones. The amount of deformation increase from the 24 inch wound roll to the 12 inch wound roll is approximately 0.0045 inches. However, the amount of increase from the 12 inch roll to the 6 inch roll is approximately 0.0062 inches.

The wound roll impingement profile also depends on the nip roller's width. Logically, the profile should react to varying nip widths in the same manner as it did with the wound roll. That is, a shorter nip roller has less surface area to distribute loads, and therefore the resulting profile values should be larger than those of the standard. To verify the profile's dependence on nip width requires the program to accept different nip widths. IMPINGE does accept different nip widths, but the CMD locations of the applied load must accommodate the wound roll's width. Since the wound roll standard width is 36 inches, the loads must remain at zero and 36 inches CMD. Therefore, as the nip width decreases, the program assumes the stub shafts take up the remaining width as illustrated previously in figure # 13. In addition, the model does not account for the reduced radius of the stub shafts which means the wound roll is assumed to contact the stub shafts across the entire width. Therefore, IMPINGE can not examine the effects of reduced nip widths.

The combination of the wound roll and nip roller geometries decides the actual impingement profiles. Although IMPINGE can not accommodate an analysis of reducing nip widths with respect to wound rolls, it does accommodate reducing both the nip width and the wound roll width. Figure # 28 compares the deformations of the standard 36 inch wound roll impinged by a 36 inch nip roller and a 12 inch wound roll impinged by a 12 inch nip roller. The distributions of the deformations are the same while the magnitudes increase uniformly by a factor of 2.7. In addition, the values for the 12 inch wound roll/nip combination are identical to those for the 36 inch nip impinging the 12 wound roll.

Deformation across equal Wound Roll/ Nip Roll Widths

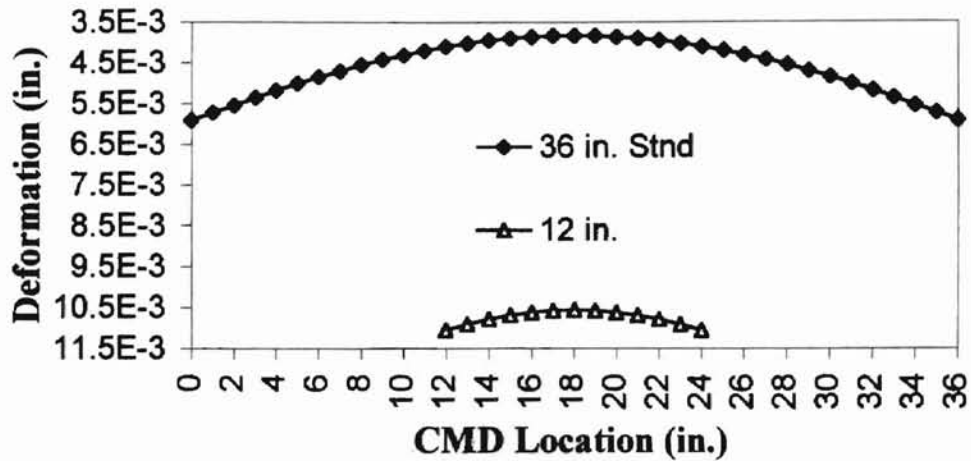


Figure # 28: The deformations for the 12 inch nip roller impinging a 12 inch wound roll have the same shape as the standard.

Figure # 29 shows the maximum pressures and the loads per width across the wound roll width. The pressures across the 12 inch wound roll increase 2.2 times to around -42 lb./in². The load per width increases more dramatically by a factor of 3.6.

IMPINGE's shortcoming in accommodating reduced width nip rollers is a side effect of another feature. As mentioned, stub shafts are assumed to take up any remaining width between the applied loads that is not the actual nip roller. The purpose of this is to accurately represent the typical 3 or 4 inch separation between an applied load and the nip roller's edge (AB in figure # 14). By permitting a separate stub shaft MOI, IMPINGE models the effect of the stub shafts and their reduced stiffness. The program just as readily models nip rollers with two different MOIs across the width.

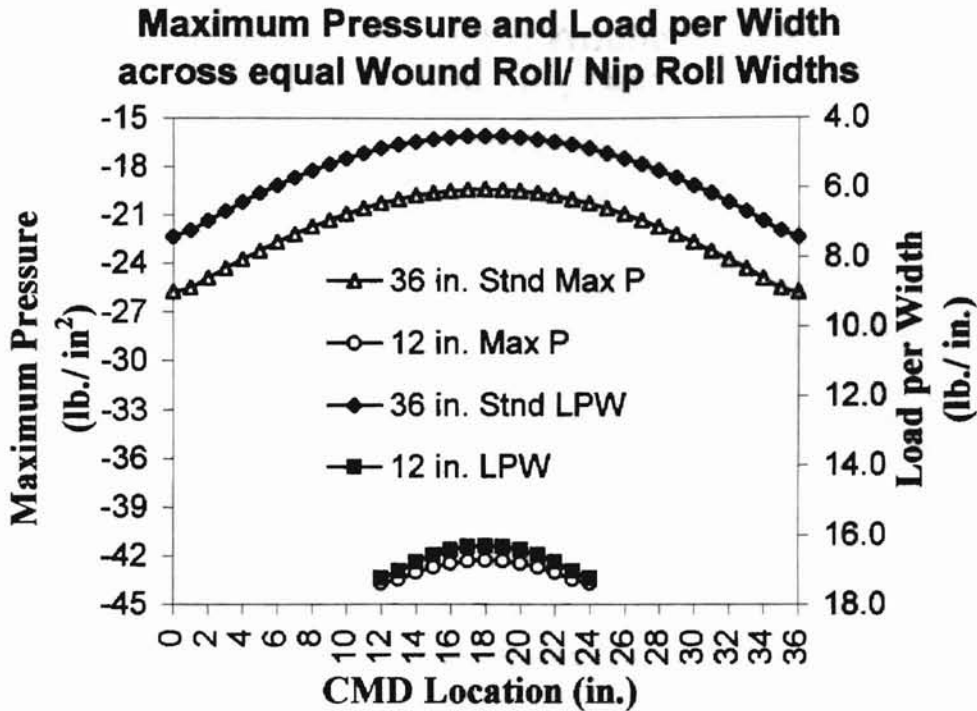


Figure # 29: The maximum pressures and loads per width across the wound rolls have the same distribution.

The deformations for three such nip rollers are plotted against the standard in figure # 30. The first nip roller with a reduced MOI has two 6 inch sections on each end. The MOI value is 0.785 in^4 corresponding to a solid 2 in. diameter rod. The center section retains the standard 5.2 in^4 MOI. The second and third nips have two 12 in. and 15 in. end sections respectively. All three reduced MOI nips have a clear bending behavior transition. In comparison to those corresponding to the standard (uniform) nip, the wound roll deformations are greater at the ends and smaller in the center.

Deformation across the Wound Roll Width for Nips with reduced MOI ends

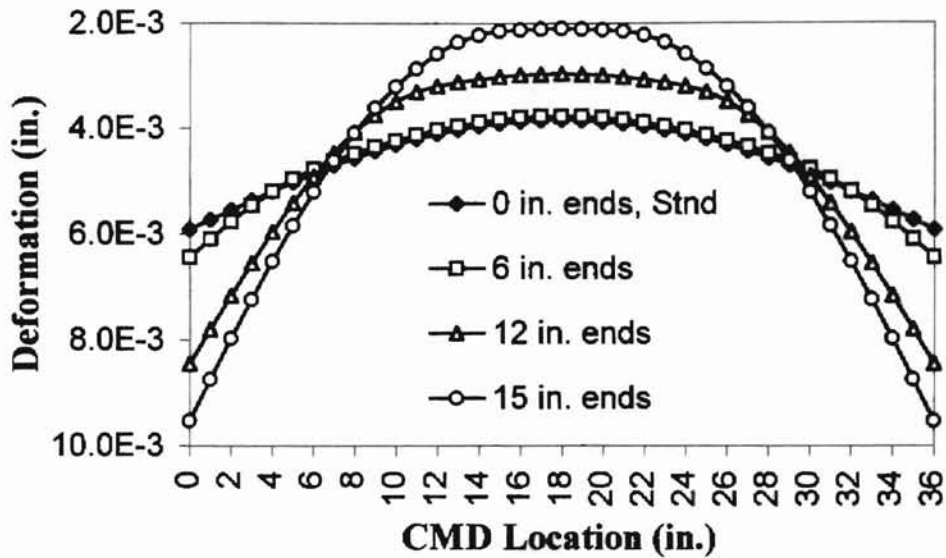


Figure # 30: The slope changes in each plot draw attention to the MOI transitions.

Likewise, the load per width across the width increases at the ends and decreases in the center with respect to the standard as seen in figure # 31. In fact, the load per width for the nip with 15 in. reduced MOI ends is nearly double at CMD zero and 36 in., while in the center it is approximately half. As seen in figure # 32, the maximum pressure follows the same trend with values for the nip with 15 in. reduced MOI ends almost 1.5 times larger at the ends and half as large in the middle.

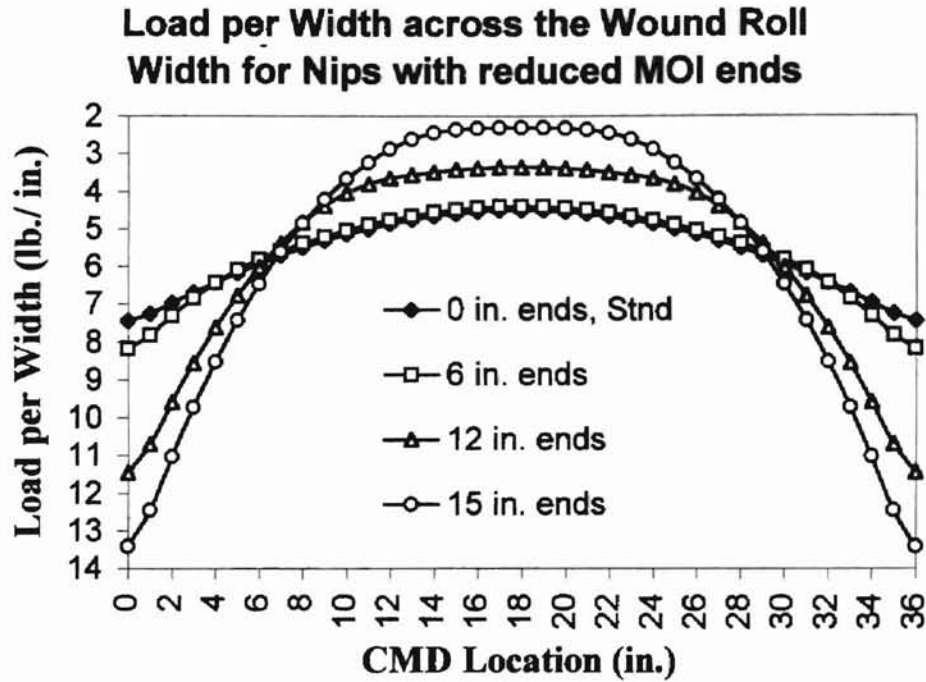


Figure # 31: The load per width varies more drastically across the wound roll when impinged by reduced MOI nips.

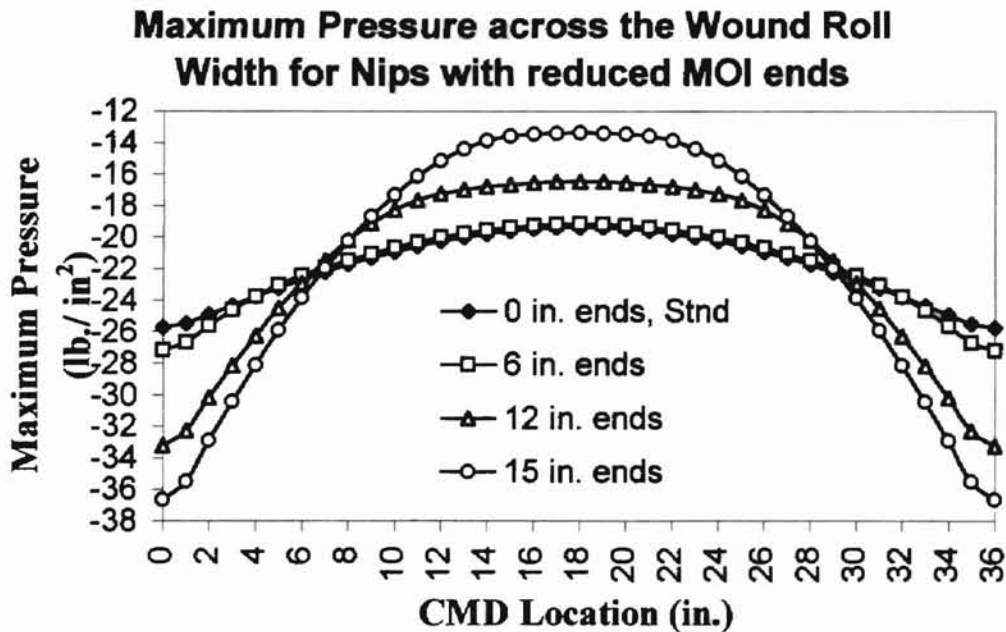


Figure # 32: The maximum pressure also varies more drastically across the wound roll when impinged by reduced MOI nips.

MATERIAL

Changing the nip roller and wound roll materials heavily impacts the impingement profile. Figure # 33 displays the deformation results for an elastic modulus corresponding to a steel nip versus the standard aluminum nip. As expected, the nip is stiffer and reacts oppositely from the reduced MOI nips. Also in the figure are curves for both a 36 in. and a 24 in., 25 μm polyester wound roll material impinged by 36 in. nip.

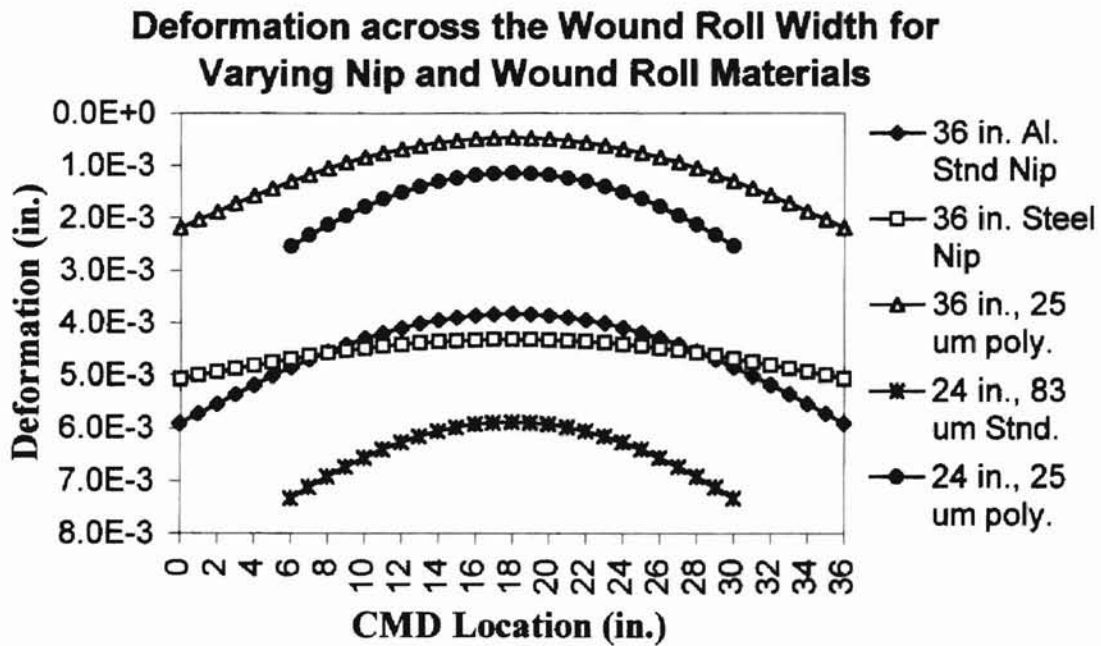


Figure # 33: The nip and wound roll materials significantly affect the deformation across the width.

The 36 in. and 24 in., 25 μm curves have 20 % and 12 % of the deformation corresponding to their respective 83 μm curves. The loads per width for the same materials behave similarly as seen in figure # 34.

Load per Width across the Wound Roll Width for Varying Nip and Wound Roll Materials

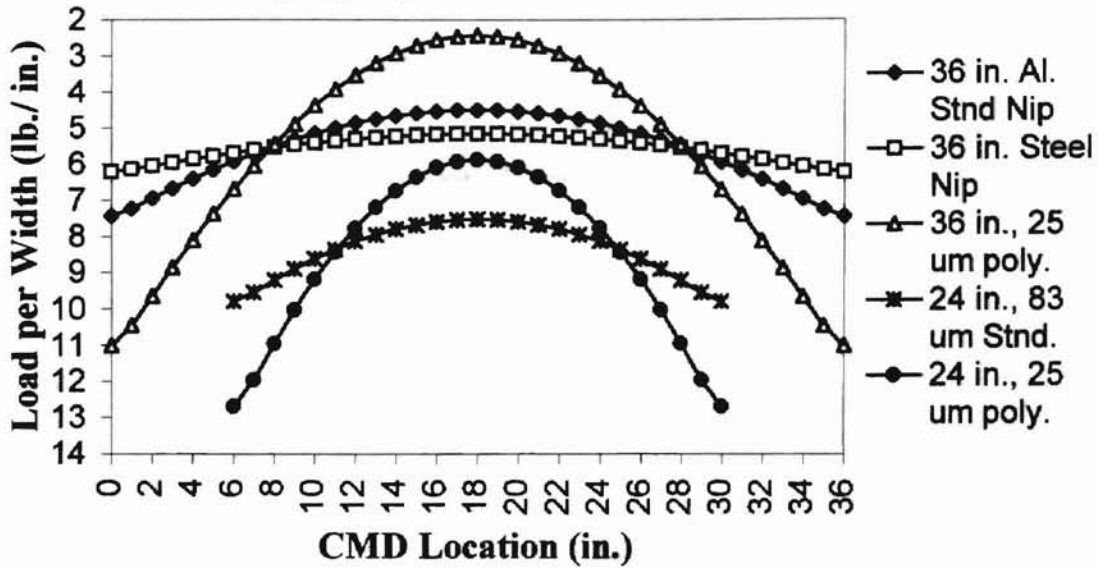


Figure # 34: The load per width increases drastically at the wound roll's ends for the 25 μm material.

Most notably from the figure are the large differences in maximum and minimum load per width for the 25 μm material. The maximum load per width values for the 25 μm , 24 in. material are over twice the minimum values while they are only 1.3 times for the 83 μm , 24 in. material. The maximum pressure also has much larger values at the ends for the 25 μm polyester (figure # 35). However the maximum pressure behavior differs from the load per width in the center of the wound roll. Whereas the load per width values are less in the center of the 25 μm , 24 in. material than for the 83 μm , 24 in. material, the maximum pressure values for the 25 μm , 24 in. material are all greater than those for the 83 μm , 24 in. material.

Maximum Pressure across the Wound Roll Width for Varying Nip and Wound Roll Materials

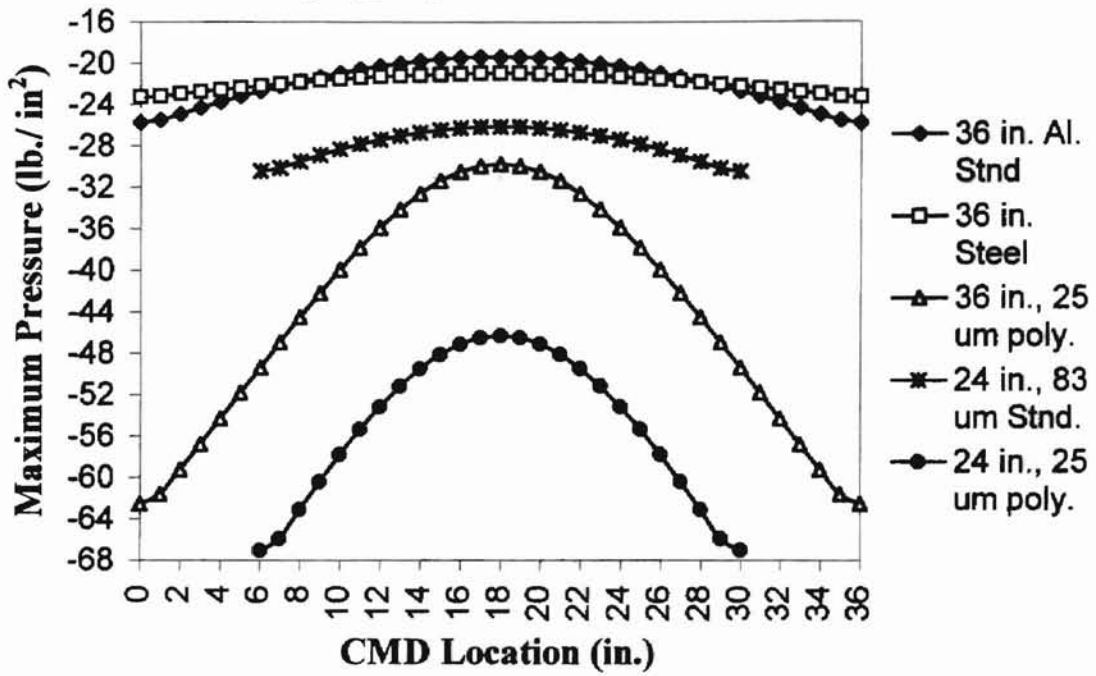


Figure # 35: The maximum pressure is greater across the entire width for the stiffer 25 μm material.

LOADING

Figures # 36, and # 37 summarize the applied load's impact on the deformation.

For figure # 36 the wound roll material is the standard 83 μm newsprint.

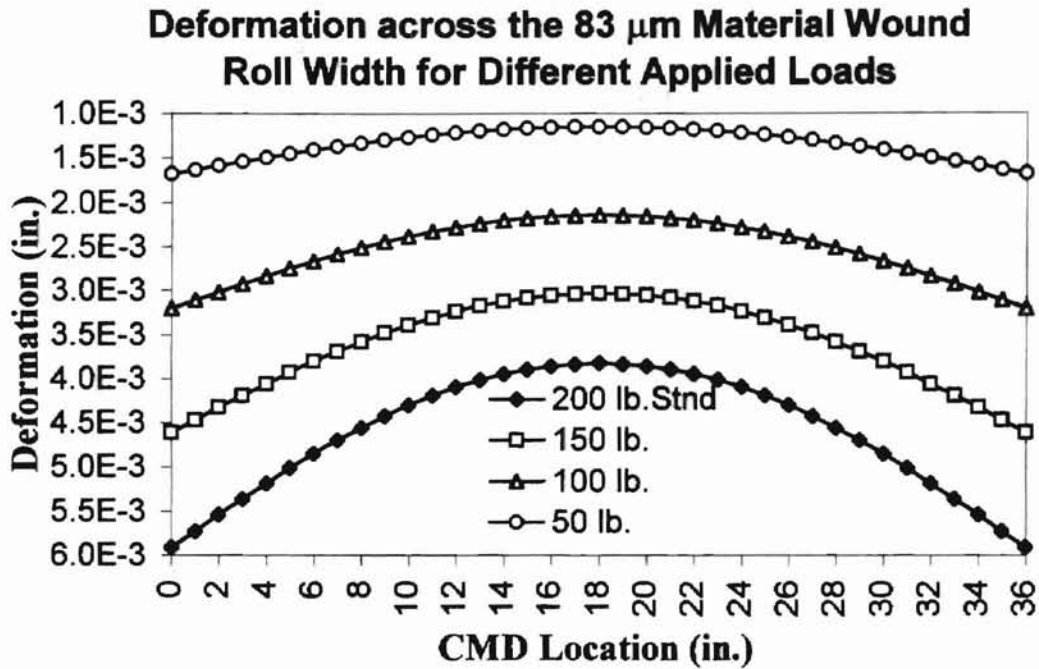


Figure # 36: The 83 μm newsprint overall deformation increases with applied load.

As expected, the deformation increases with applied load. Also, the difference between the maximum and minimum values across the width becomes more pronounced with increasing load. Likewise, in figure # 37, the deformations increase with applied load for the 25 μm polyester, although as expected the magnitudes are less than for the 83 μm newsprint. The load per width and maximum pressure across the width exhibit similar behavior for both materials.

**Deformation across 25 μm polyester Material
Wound Roll Width for Different Applied Loads**

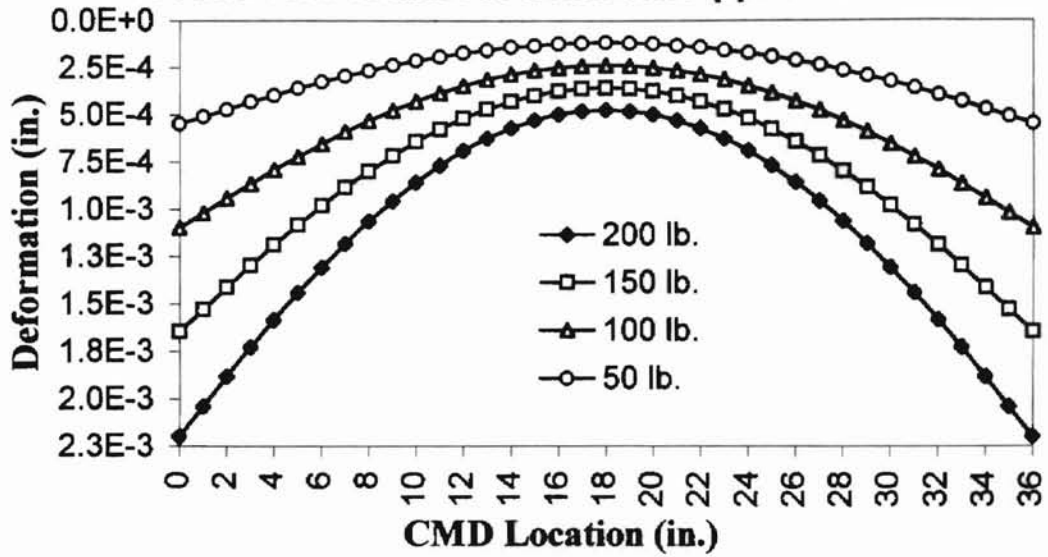


Figure # 37: The 25 μm polyester overall deformation increases with applied load.

MODELING

Characterizing the effect modeling has on the output from IMPINGE requires an examination of the iterative load step size, the program's starting load, the convergence iteration, and the rotational constraints. The iterative load step size should alter the computed profile's accuracy. As discussed in chapter III, the foundation stiffness at each iteration depend on the deformations of the previous iteration. A smaller step size reduces the computed stiffness error by restricting the possible range of deformations. Figure # 38 compares the deformation outputs for the 25 lb. standard, 50 lb., 100 lb. and the 200 lb. step sizes. Since the nip load remains at 200 lb., the number of steps are 8, 4, 2, and 1 respectively.

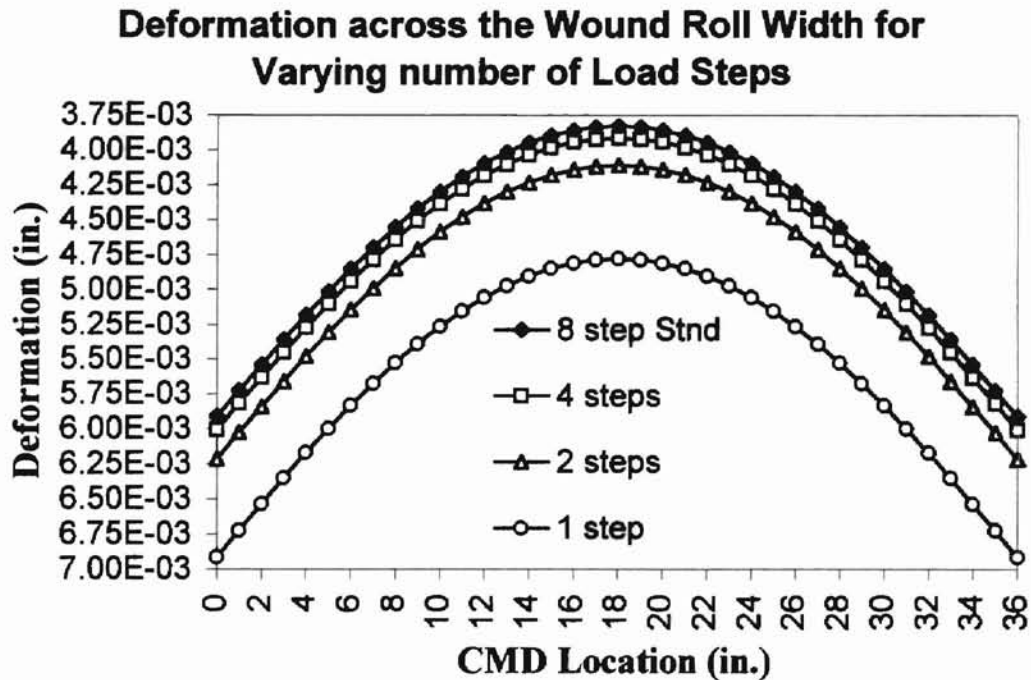


Figure # 38: IMPINGE over-estimates the deformations when the number of force steps are reduced.

As the step size increases, or the number of steps decrease, the program's output deformations increase. The difference between the deformations reduces rapidly with increased steps. The results for 1 step versus 8 steps have a relative error of approximately 25 % at the center and 17 % at the ends. The 4 step results however have a relative error of only 2%. The loads per width across the width for the different number of steps are plotted in figure # 39.

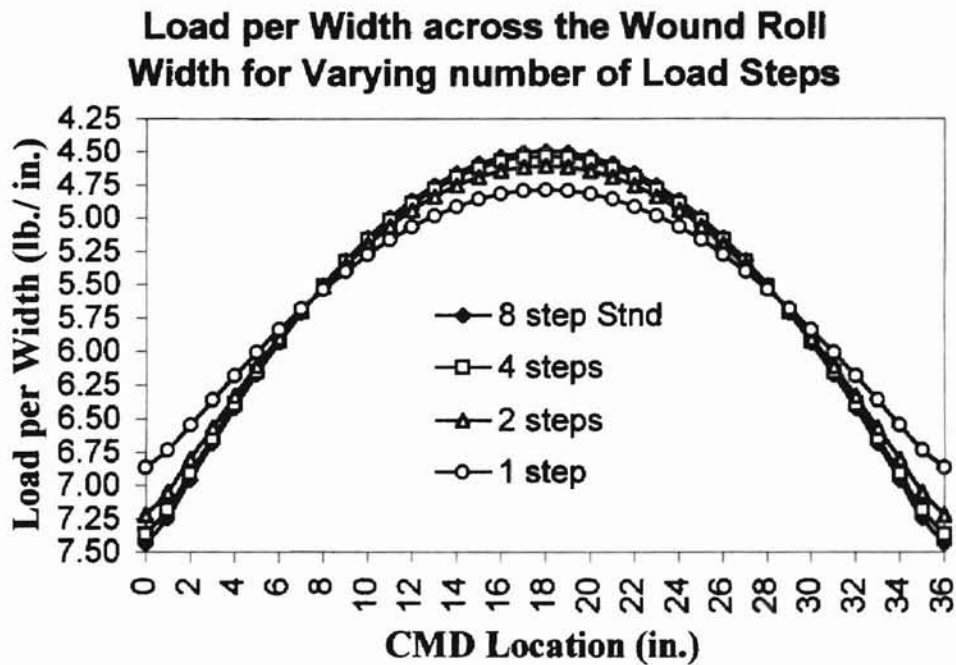


Figure # 39: The loads per width converge as the number of force steps increase.

Their behavior varies from the deformation in that the program under-estimates the values towards the ends and over-estimates the values at the center. The relative error for the 1 step results is about 7 %, whereas it is less than 1 % for the 4 step results. Figure # 40 shows the maximum pressure's dependence on the number of steps.

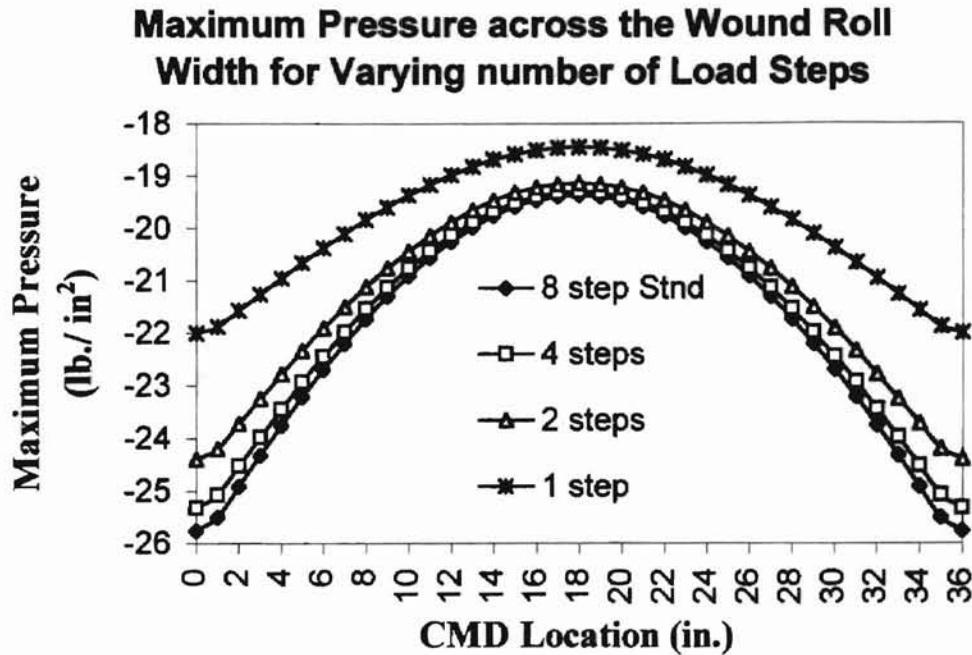


Figure # 40: IMPINGE under-estimates the maximum pressures across the wound roll when the numbers of steps is reduced.

While the program overestimates all of the deformations when the steps were reduced, it underestimates all the pressures. The largest relative error of 15 % occurs at the ends for the 1 step results. At 4 steps the error is only 2 %.

Since IMPINGE allows different starting load sizes, two different starting loads are compared. The results for a starting load of 0 lb. and for 150 lb. appear in figure # 41. The force step size for both curves is 25 lb. Since the program calculates the load per width and the maximum pressure from the deformations, they are identical for both starting values also. Evidently, the starting load size has little effect on the outcome. However, the error may increase under higher loading or for stiffer materials.

Deformation across the Wound Roll Width for Two Different Starting Forces

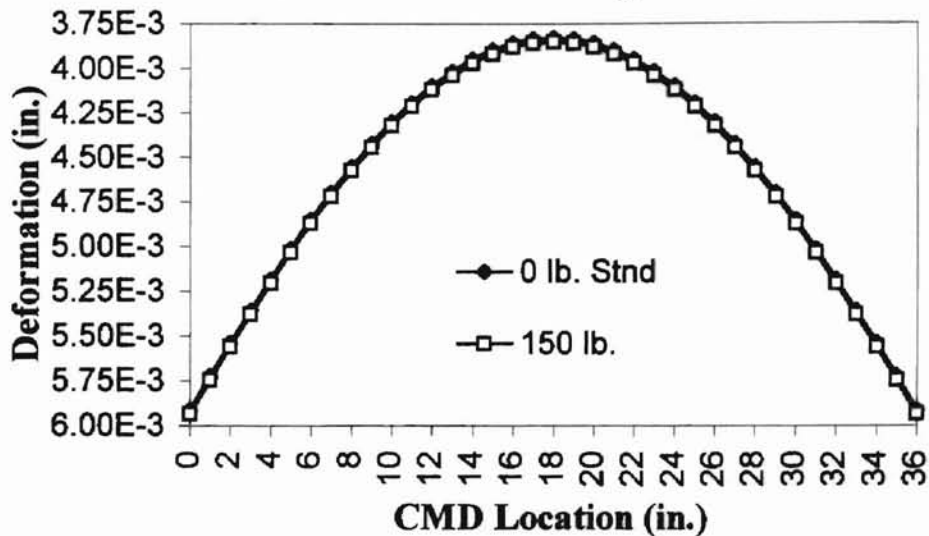


Figure # 41: The deformations for two different starting forces are nearly identical.

The investigation of the convergence iteration impact examines the effect on both the 83 μm newsprint and the 25 μm polyester wound rolls. As depicted in figure # 42, the deformation across the width at the 200 lb. load is insensitive to the convergence iteration regardless of the material. In addition, the load per width and the maximum pressure do not differ.

Constraining the nip ends from rotation alters the profile significantly. Figure # 43 plots the deformations of the wound roll when the nip core is restrained from rotation on the left, the right, and for both. For the nip with one end constrained from rotating the deformations at the constrained end are about 67 % of the values for the totally free nip. Deformations for the non constrained end are 20 % greater however. When the nip is constrained on both ends, the deformation across the width is much more uniform,

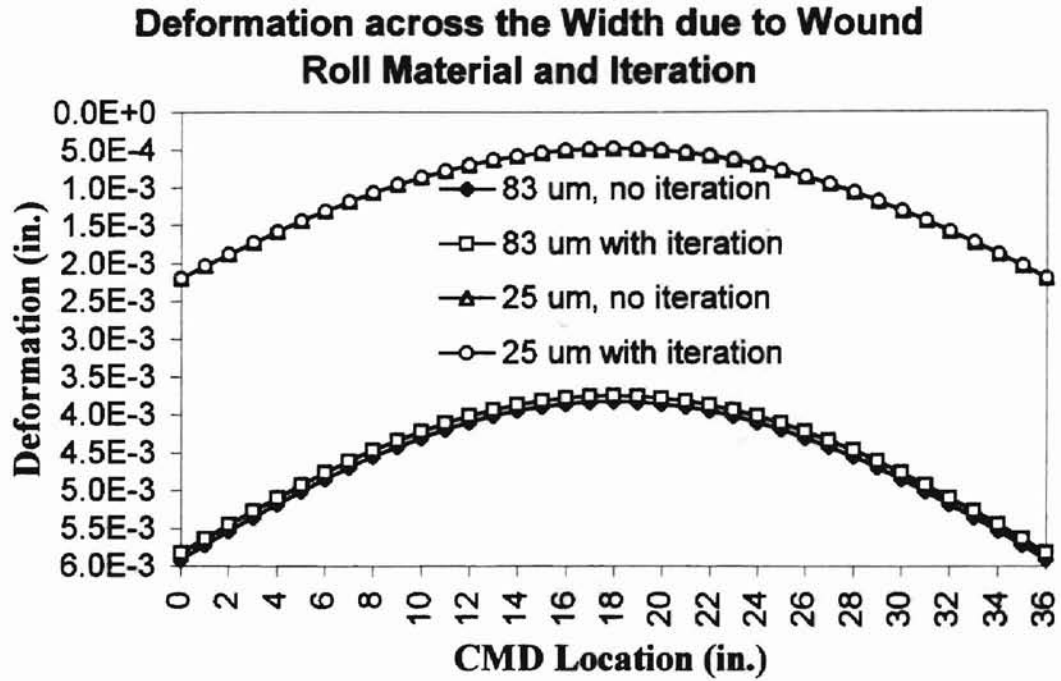


Figure # 42: The convergence iteration has no impact on the deformations for the 200 lb. applied load.

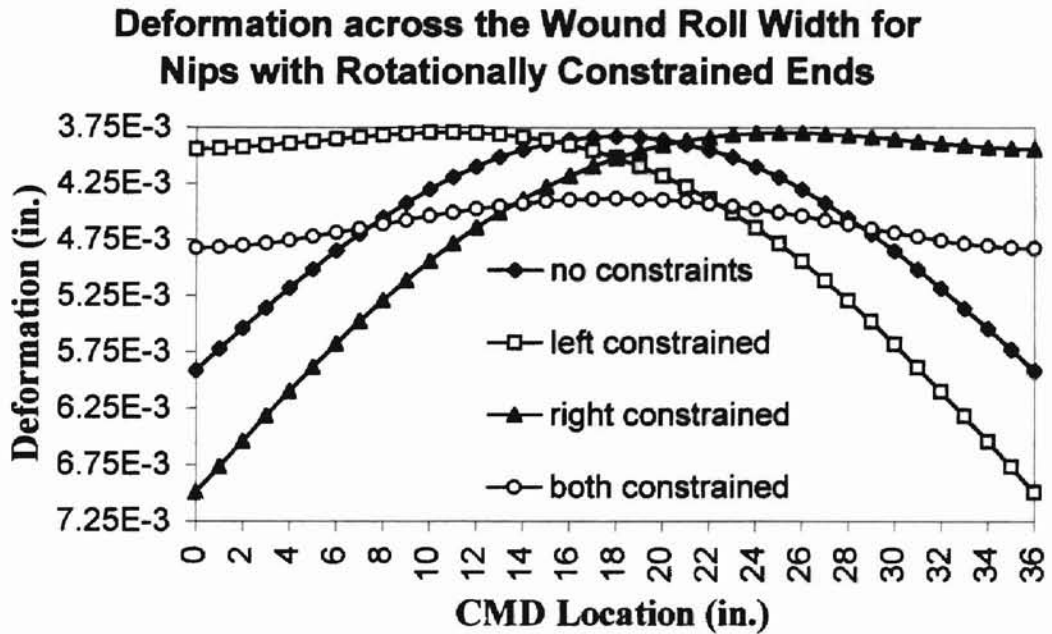


Figure # 43: The deformations of the nip ends decrease when they are constrained from rotating.

but is larger in the middle of the wound roll than for the other configurations. As seen in figures # 44 and # 45 the load per width and maximum pressure behave in the same way. Their center values reach 5.3 lb./ in. and -21.2 lb./ in² respectively.

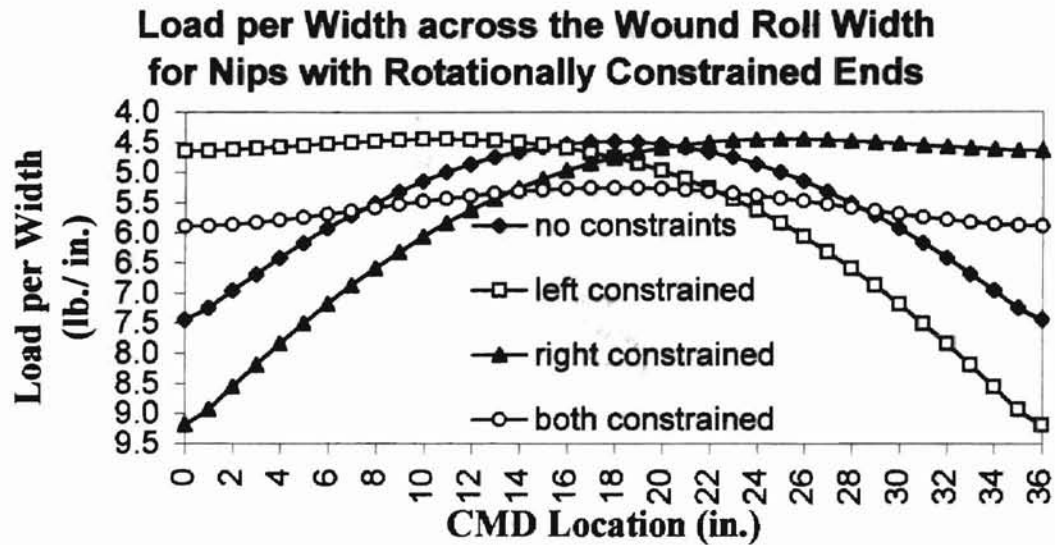


Figure # 44: The load per width values are fairly uniform across the width for the totally constrained nip.

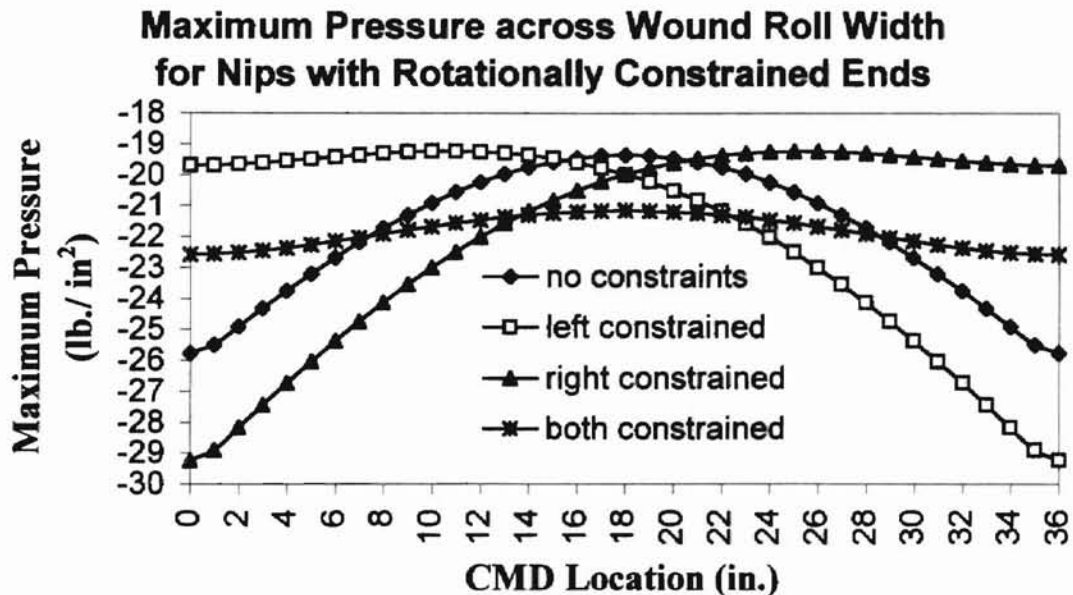


Figure # 45: The maximum pressure values vary from -22.6 lb./ in² to -21.2 lb./ in² for the totally constrained nip.

INCREMENTAL DEFORMATIONS AND GAPPING

IMPINGE outputs the deformation results at each load step to a file. An example of the output file (PINGEDEF.OUT) is in Appendix E. The plotted data is in figure # 46.

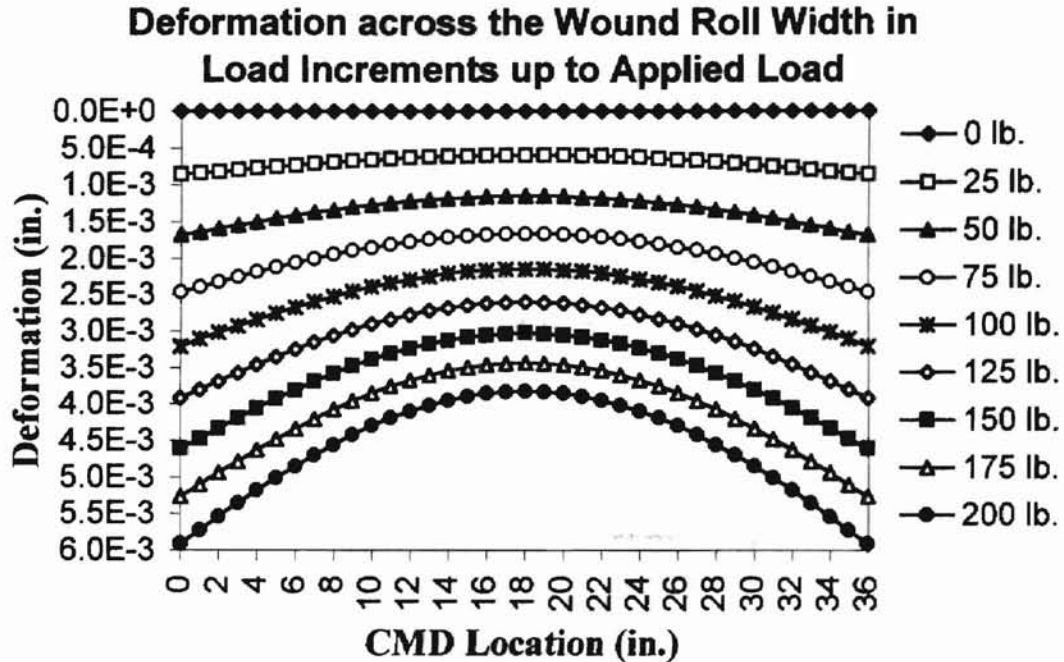


Figure # 46: The rate of deformation increase, decreases at each load step.

While the standard configuration did not gap, stiffer wound rolls will gap under similar applied loads. Table # 3 lists the parameters for another configuration. The values not listed in the table are the same as for the standard. This configuration gaps across more than 75 % of its width before the program reaches 300 lb. The incremental deformations at loads prior to the applied load are plotted in figure # 47.

Group	Description	Value
GEOMETRY		
	CMD Distance from zero to Wound Roll	6 inches
	CMD width of wound roll	24 inches
MATERIAL		
	K1	1.32E-08 lb./in ²
	K2	524.6
	K3	62 lb./in ²
LOADING		
	Applied Nip Force Load	300 lb.
MODELING		
	Nip Force Step Size	15 lb.
	Should the FEM Iterate for Deformations?	N
	Number of Finite Elements across the Wound Roll	24
	Rigid Left, and Rigid Right	

Table # 3: The configuration is for a stiff 25 μm polyester material 24 inches in width.

Incremental Deformations Across Gapped Wound Roll with Instability

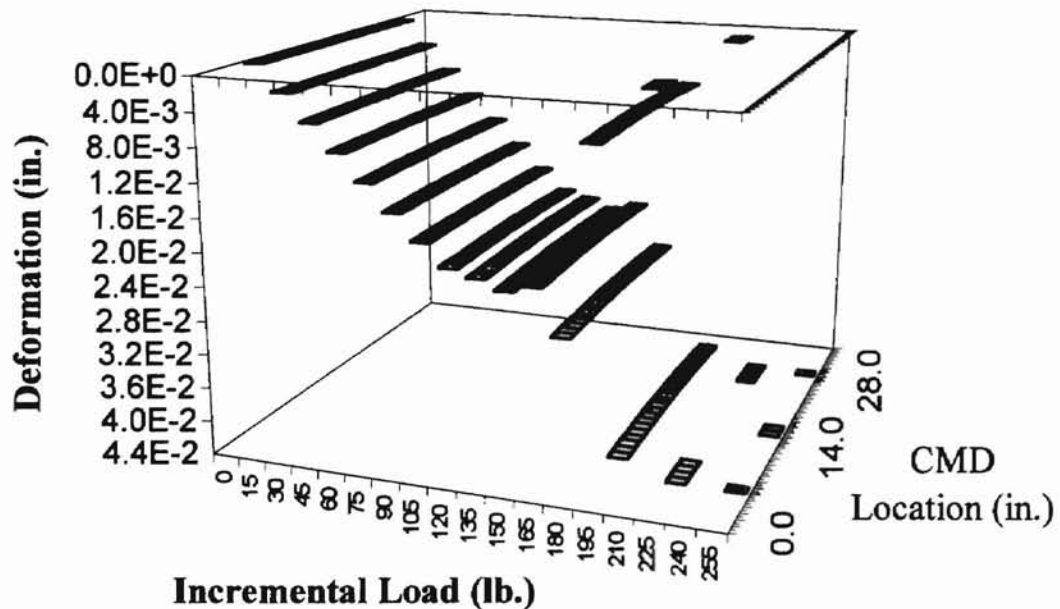


Figure # 47: The incremental deformations across the width become unstable at loads closer to the applied load.

Figure # 48 plots the deformation of the six inch CMD location versus incremental loads.

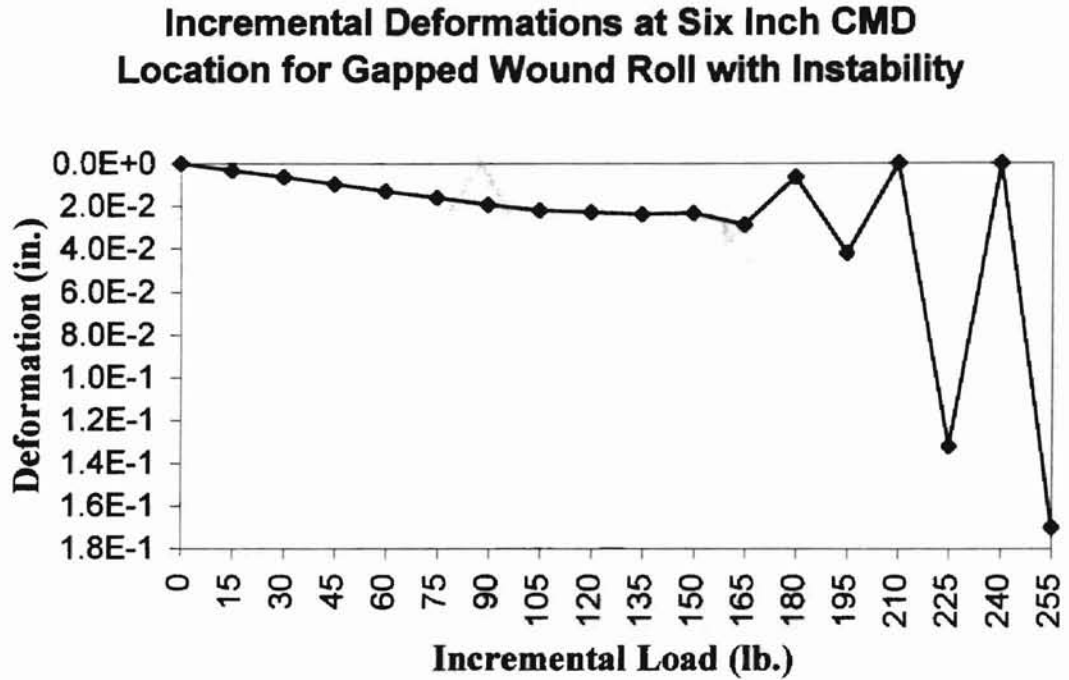


Figure # 48: The six inch CMD location becomes unstable above 165 lb.

Both figures indicate the deformations become unstable as the loads increase. As mentioned this is due to the method used to calculate the stiffness for each load step. The instability is eliminated if the program uses the convergence iteration option. The results for the convergence iteration option appear in figure # 49. When the deformations are made to converge, the gapping occurs at 615 lb. and at no time do the values go unstable. The deformations at each force increment smoothly transition into the next increment. Also the deformation increase rate undergoes an exponential decay.

Incremental Deformations Across Gapped Wound Roll with Convergence Induced Stability

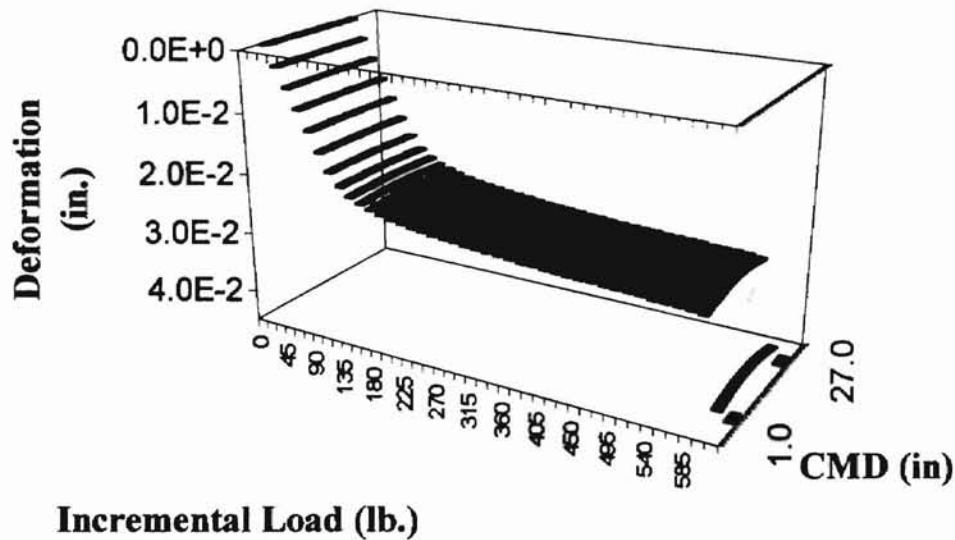


Figure # 49: The convergence iteration stabilizes the deformations.

Figure # 50 plots the wound roll deformations across the width for the last two load increments.

Deformations Across Gapped Wound Roll with Convergence Induced Stability at Final Load Steps

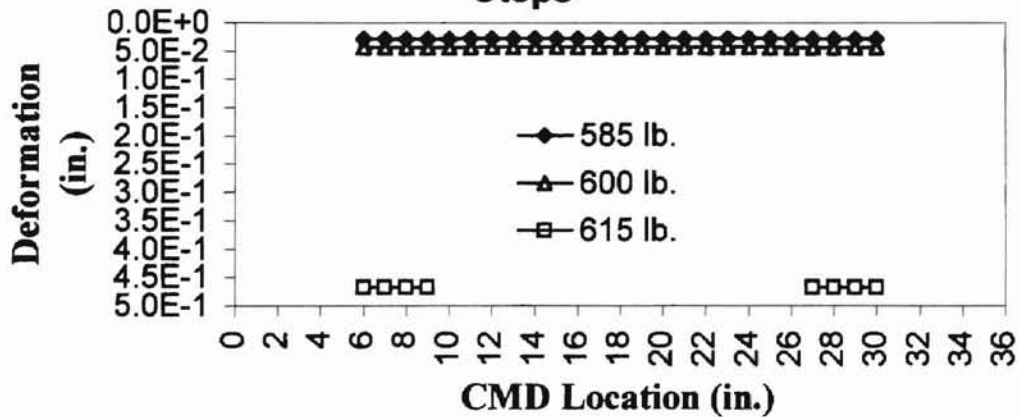


Figure # 50: The nip roller pulls away from the wound roll between the 600 lb. and 615 lb. load increments.

At 615 lb. exactly 75 % of the elements gap leaving six elements deforming 0.468 inches. The deformations at 585 lb. are all approximately 0.028 inches, while those at 600 lb. are 0.043 inches.

Figure # 51 displays the loads per width for increments up to the gapping load.

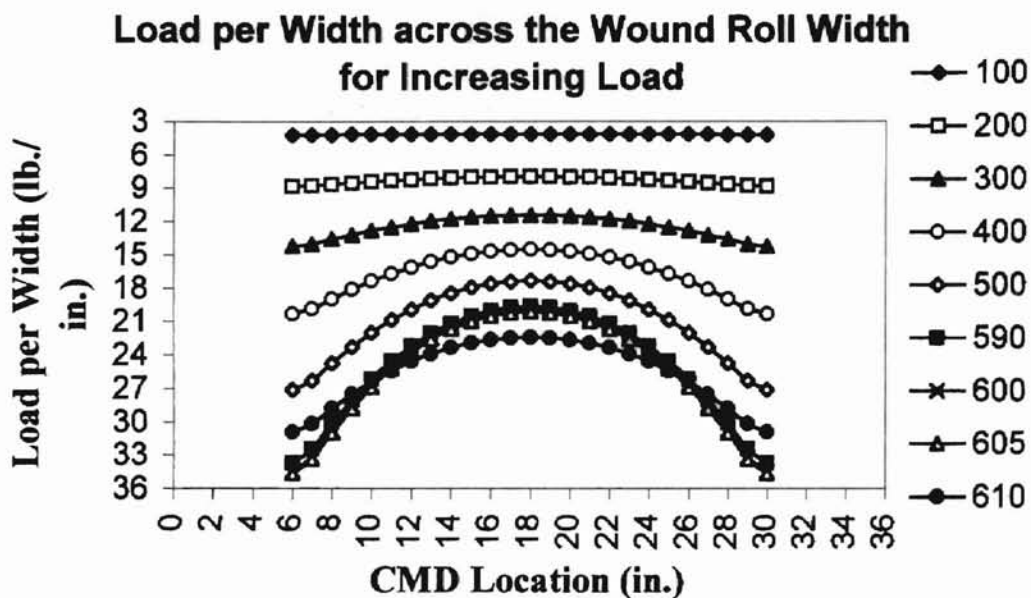


Figure # 51: The load per width saturates just before the nip gaps from the wound roll.

As the applied load approaches the gapping load, the load per width across the wound roll width flattens out. The values in the center continue to increase, while the values at the ends begin to decrease. The load per width still varies noticeably across the wound roll width.

CHAPTER V

CONCLUSIONS

This characterization of a nip impinged wound roll focuses on the loading between the nip roller and the wound roll across the entire width of contact. The analyzed loading not only includes the contact loads, but also the deformation, and the contact pressures as a function of wound roll width. The investigation includes variables such as geometry, applied loads, boundary restraints, materials and the modeling approach. The developed IMPINGE program facilitates the investigation and provides a lasting method for continued investigations.

The results represent the combination of numerous theories and models. Hertzian contact theories provide a theoretical framework governing the nip's impingement into the wound roll. Pfeiffer's pressure versus strain material relations weave real material tendencies into the theoretical framework. Finite Elements provide a proven modeling technique to analyze the nip/ wound roll interaction and maintain a methodology in the approach.

The results demonstrate that the wound roll geometry greatly affects load distributions. The loads across the width decrease as the wound roll grows in radius. This is due to the increased contact area between the nip roller and the wound roll. The wound roll width is the number one contributor to the variations in deformations, loads per width, and maximum pressure across the width. Narrow wound rolls clearly vary less across their width than do wider ones.

The web material also contributes to the magnitudes of the deformations, loads per width and maximum pressures. Stiffer materials vary much more drastically across the width than do softer ones. However, the tendencies remain the same regardless of the material.

The nip roller's geometry, material, and constraints contribute greatly to the shape of the load distributions across the wound roll. Comparison's between nip rollers with different MOIs resulting from different radii, show that stiffer nip rollers produce less variation across the wound roll width. Likewise, stiffer nips resulting from a higher modulus of elasticity result in less variation across the width. Constraining the ends of the nip roller from rotating also reduces the variation. Therefore, the stiffest most constrained nip is desired to reduce variations across the wound roll. However, such nips are more massive, and they create dynamic problems during winding. This forces the nip roller selection to be a tradeoff.

The results also show sensitivity to the modeling method. The model requires sufficient load steps to produce repeatable results. In addition, the sensitivity to the starting force greatly increases as the load increases or for stiffer materials. It is wise then to always begin the model at 0 applied load. The iteration process is absolutely necessary for all but light loads or soft materials. Also, gapping is a complex problem, that demands much attention to detail.

Overall, the IMPINGE program works well to characterize the nip induced wound roll. However, some assumptions leave it with limited application as discussed next.

ASSUMPTIONS

True to any model this investigation simplifies the analysis via assumptions. First of all the wound roll core is assumed rigid. The validity of the assumption is somewhat questionable since it varies with applied loads, wound roll width, and web material stiffness. For the case of the 615 lb. load on a core (presumed to be 36 inches in length) for the 83 μm newsprint, the core's reaction most likely plays a significant role.

The nip and wound roll are also assumed to be parallel. This is definitely a valid assumption in that they are often parallel during winding in industry. However, in industry they are also often intentionally skewed. This reduces the pressure near the ends and more importantly increases the center's pressure. The effect of the assumption then is the reduction of applications the IMPINGE program can address.

As mentioned the nip roller is assumed to be at least as wide as the wound roll. Like the skewed angle between the nip roller and wound roll, a shorter nip roller would increase the pressures in the center of a wound roll. The ends of the wound roll which were not contacted by the nip roller would most likely react similarly to a Hakiel's [3] center wound roll model with a relaxation radius. The assumption does help to eliminate ridiculous configurations like a 1 inch nip impinging a 36 inch wound roll.

The investigation does not allow for non uniform wound rolls due to thickness variations. Thus it assumes the wound roll is uniform before the load is applied. This assumption of course can never be absolutely true as no web is perfectly uniform. However the sheer number of articles addressing thickness variations show it to be a topic for separate investigation, and therefore it falls out of the scope of this investigation.

The nip roller is also assumed to have no compressible covering and to be uniform. Like the thickness assumption, this serves to reduce the scope of the investigation. Coverings or local non-uniformities do not change the nip's behavior consistently across the width. The results for these special cases are meaningless without the characterization for the uniform case as a basis for comparison.

Equation # 2 normalized the strain associated with the Pfeiffer models of the wound roll material behavior. This means the deformation in the radial modulus (E_{roll}) equations apply to a 1 inch thick stack of material. This allows comparisons with 1 inch empirical stack data, reduces the required input values, and reduces computation complexity. Truly the deformations should first be divided by their stack height to obtain the deformation per inch (strain). For this, the stack height is the radius of the wound roll minus the radius of the wound roll core.

FURTHER INVESTIGATION

Many of the assumptions are prime candidates for further investigations. The complexity of the model will increase with each eliminated assumption, but so will the usefulness and application.

This characterization opens the door for investigations into 3 D winding. The very nature of FEMs accommodate segmenting the wound roll width corresponding to local non-uniformities, and maintaining inter-relations between the segments. Thickness variations, nip abnormalities, load variations, stack heights, and so forth incorporate easily. In addition, the center winding models which "wind" the web on by laps, can be included. Ultimately, the model would be quite comprehensive.

BIBLIOGRAPHY

1. Hakiel, Z., "Nonlinear Model for Wound Roll Stresses." TAPPI Journal, Vol. 70, No. 5, pp. 113-117, May 1987
2. Hakiel, Z., "On the Effect of Width Direction Thickness Variations in Wound Rolls." Proceedings, First International Conference on Web Handling, Oklahoma State University, 1991
3. Hakiel, Z., "A Nonlinear Wound Roll Stress Model Accounting for Widthwise Web Thickness Nonuniformities." AMD, Vol. 149, ASME 1992
4. Pfeiffer, J. D., "Surface Winding to Overcome the Strain Deficiency." Proceedings of the 1990 TAPPI Finishing and Converting Conference, pp. 233-236
5. Good, J.K., Wu, Z., and Fikes, M.W.R., "The Internal Stresses In Wound Rolls with the Presence of a Nip Roller", Journal of Applied Mechanics, 1992
6. Rodal, J.A., "Paper Deformation in a Calendering Nip." TAPPI Journal, Vol. 76, No. 12, pp. 63-74, December 1993
7. Vaidyanathan, N., "A Study of Wound Roll Slippage", Ph.D. Thesis, Oklahoma State University, 1996, pp. 216-219
8. Johnson, K.L., Contact Mechanics, Cambridge: Cambridge University Press, 1985
9. Timoshenko, S.P., and Goodier, J.N., Theory of Elasticity, 3rd edition, McGraw-Hill, 1970

10. Felippa, C.A. "Lecture Notes in Introduction to Linear Finite Element Methods",
Vol. 1, pp. 10-8, pp. 7-6, 1989
11. Chandrupatla, T.R., and Belegundu, A.D., Introduction to Finite Elements in Engineering, Prentice Hall, 1991
12. Good, J.K., and Pfeiffer, J.D., "Tension Losses During Centerwinding"
Proceedings of the 1992 TAPPI Finishing and Converting Conference, Nashville,
TN, October 18-21, 1992, pp. 297-306
13. Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., Numerical Recipes, Cambridge University Press, 1992
14. Mathews, J. H., Numerical Methods for Mathematics, Science, and Engineering,
2nd ed., Prentice-Hall, 1992 pp. 61-62

APPENDIX A
IMPINGE PROGRAM CODE

PROGRAM IMPINGE

```

*****
*   This program is written by Paul Hoffeecker for the
*   MAE Research Lab at Oklahoma State University.
*****
*
*   Purpose: To explore the effect of a Nip Roller impinging on
*           a wound roll.
*
*   MARKER =1
*
*   Variable Definition/ Description:
*
*   ADELTA      increment value for halfwidth (length)
*   ALOWDRAD    allowed radius in inches or meters
*   ALOWDWID    allowed machine width in inches or meters
*   ASTART      starting value for incrementing halfwidth (length)
*   AVERLPW     average load per width value (force per length)
*   A(,)        2 d temporary Global FEM stiffness matrix
*   BEAMESM(,)  2 d array of FEM beam element stiffness matrix
*   BEAMNODE    indicates the array position of the beam portion of
*               the current element's nodes
*   B()         temporary 1 d FEM global array of nodal forces
*   CODEBEAM()  1 d array indicating type of beam element...
*               0 = stub; 1 = nip
*   CODEWINK()  1 d array indicating type of wink element...
*               0 = none; 1 = foundation
*   CONDITON() 1 d array stating gapping condition of each element
*   CONSTLFT    indicates if nip's left end is rotation constrained
*   CONSTRGH    indicates if nip's right end is rotation constrained
*   COLUMN      loop variable in array that increments the column
*   CURNTDEF    current deformation (deform<inc>) under consideration
*   CURRELEM    indicates element under current consideration
*   CURRLOCA    location of node under current consideration
*   CURRNODE    indicates node under current consideration
*   CURRSTEP    counts the number of DELTAFOR steps the program takes
*               towards the desired nip force
*   CURWINEL    indicates winler element under current consideration
*   CYCLE        counts the times through the convergence loop
*   DATAPTS     number of pressure verses strain data points
*   DATE        date of program run
*   DECISON1    Y or N response to find pressure versus strain data
*               parameters
*   DECISON2    Y or N response to input FEM node locations
*   DEFORM()    1 d array of deformations from impingement (length)
*   DELTAFOR    Nip force increment size in FEM
*   DIMENSON    indicates array size for FEM matrix solution routines
*   DISPLRES    FEM deformation convergence resolution
*   DISTANCE    generic FEM element distance from previous node to
*               current node
*   DOFFERND    degrees of freedom for each FEM node
*   DONE        logical variable indicating end of FEM convergence
*               loop
*   EFFECTE     effective modulus E of nip and wound roll combination
*   ELEMDISP    temporary holding variable for winkler element
*               displacements
*   ELEMSTIF()  1 d array of element stiffnesses
*   ELEONES     remainder NMCROSEL/ 10
*   ELTENS      whole number of NMCROSEL/ 10
*   ELLENGTH()  1 d array of FEM element lengths
*   EROLLRAD    modulus of elasticity of wound roll

```

```

*          (force per length^2)
*
* ERRORCHR      holds name of a variable if it is input out of ac-
*                ceptable range
*
* FORCERES      smallest force increment allowed
* FORCE()        1 d FEM global array of nodal forces
* FUNCMAX()     1 d function array corresponding to right (max)
*                deformation in FEM convergence bracketing
*
* FUNCMIDL()    1 d function array corresponding to middle
*                deformation in FEM convergence bracketing
*
* FUNCMIN()     1 d function array corresponding to left (min)
*                deformation in FEM convergence bracketing
*
* GBCOL        K matrix column index for current BEAMESM entry
* GBROW        K matrix row index for current BEAMESM entry
* GOBACK        stops or starts program
* GUESSK1      temporary guess for KONE
* GUESSK2      temporary guess for KTW0
* GWCOL        K matrix column index for current WINKESM entry
* GWROW        K matrix row index for current WINKESM entry
* HAFWIDTH()   1 d array of half widths (length)
* HALFCONW()   1 d array of the half widths of contacts for each
*                node across the wound roll
*
* HOLDER        temporary holding variable
* HOLDINUN     holding variable for the input unit number
* HOLDPRMT     holding variable for the TOPPROMPT variable
* INC          indicates the current increment number
* INDEXTNUM    miscellaneous array index
* INNEROR      indicates the number of INFILE input errors
* INFILE       input file name chosen by user
* INPTUNIT     unit number associated with the input file
* ITERATE      indicates current iteration number
* KEYBUNIT     unit number associated with keyboard input
* KONE         material specific fit parameter
* KTHREE       material specific fit parameter
* KTW0         material specific fit parameter
* K1ESTREV     temporary guess for KONE
* K1EST1       temporary guess for KONE
* K2ESTREV     temporary guess for KTW0
* K2EST1       temporary guess for KTW0
* K2HOLD       holding variable for KTW0
* K(,)        2 d Global FEM stiffness matrix
* LAST         tracks the last position number in the node location
*                sort
*
* LASTDFOR     holds final Nip force increment
* LEFTTELEM    number of elements from the left edge of the winding
*                machine to the wound roll (winkler foundation)
*
* LEFTSTUB     CMD from machine zero to beginning of nip roller
* LEFTWINK     CMD from machine zero to beginning of winkler
*                foundation
*
* LENGTH       length of current FEM element
* LINKNODE     indicates the K matrix position of the contact links
*                between the wound roll (wink) and the nip (beam)
*
* LOADPERW()   1 d array of nip roll load per unit width
* LOCALCOL     loop variable in array that increments the column
*                of a local array inside of a global one
*
* LOCALROW     loop variable in array that increments the row
*                of a local array inside of a global one
*
* LOCATE()     1 d array of the FEM node locations across the
*                machine width
*
* LOOP         miscellaneous loop variable
* LPWVSOD()    1 d array of the load force per node across the
*                wound roll
*
* MARKER       indicates the program's current location during

```



```

*
*           execution
*
* MAXARRAY      maximum allowed size of any array
* MAXDATA      maximum allowed # of pressure vs strain data points
* MAXELEM      maximum allowed number of elements
* MAXNODES     maximum allowed number of nodes
* MAXPRESS()   1 d array of maximum pressure across wound roll
* MAXPTS      maximum allowed number of impingement increments
* MAXRADIN    maximum allowed radius in inches
* MAXWIDIN    maximum allowed machine width in inches
* MAXDISP()   1 d array of right (max) deformations in FEM
*             convergence bracketing
* MIDLDISP()  1 d array of center (in between) deformations in FEM
*             convergence bracketing
* MINDISP()   1 d array of left (min) deformations in FEM
*             convergence bracketing
* MTRPERIN    conversion unit of meters per inch
* NATLOG()    1 d array of natural logs of pressure values
* NEXTROW     indicates the next row under consideration in an
*             array
* NIPEPOIS    combination of nip's poisson ratio and E (length^2
*             per force)
* NIPFORCE    total force of Nip roll onto wound roll
* NIPROLLE    modulus of elasticity of nip roller (force/ length^2)
* NIPROLLI    moment of inertia of nip roller (length^4)
* NIPROLLW    CMD width of nip roller
* NIPSTUBE    modulus of elasticity of stub shaft (force/ length^2)
* NIPSTUBI    moment of inertia of stub shaft (length^4)
* NIPSTUBW    CMD width of machine's stub shaft (length)
*             (force per length)
* NMCROSEL    number of FEM elements across the winding machine
* NMCROSD     number of FEM nodes across the winding machine
* NMWINKEL    number of FEM elements across the winkler foundation
* NMWINKND    number of FEM nodes across the winkler foundation
* NODEDISP()  1 d array of FEM nodal displacements
* NODEFORC    current force on FEM nodes
* NODELOAD    resultant force load on node
* NUMACROS    number of columns to print across the output files
* NUMGAPS     number of FEM elements which gap away from the nip
* NUMOFPTS    number of impingement increments to step through
* NUMSTEPS    number of DELTAFOR size force steps required to reach
*             desired nip force
* OUTFILE     ouptut file name
* OUTFILE1    ouptut file name 'PINGELPW.OUT
* OUTFILE2    ouptut file name 'PINGESUM.OUT
* OUTFILE3    ouptut file name 'PINGEINC.OUT
* OUTFILE4    ouptut file name 'PINGEDEF.OUT
* OUTFILE5    ouptut file name 'PINGEPRO.OUT
* OUTUNIT1    unit number associated with output file: PINGELPW.OUT
* OUTUNIT2    unit number associated with output file: PINGESUM.OUT
* OUTUNIT3    unit number associated with output file: PINGEINC.OUT
* OUTUNIT4    unit number associated with output file: PINGEDEF.OUT
* OUTUNIT5    unit number associated with output file: PINGEPRO.OUT
* PERCENT     percent of total FEM elements that are gapped
* PI          pi, 3.141592654
* POINT       indicates current data point
* POINTVEC()  1 d pointer vector for K matrix rows during inversion
* POISROLL    Poisson's ratio of wound roll
* POISSNIP    Poisson's ratio of nip roller
* PRESSDAT    pressure versus strain data input file name
* PRESSURE()  1 d array of material pressure data
* PRESUNIT    unit number associated with pressure vs strain data
*             input file

```

```

*      PROCEED      stops or starts program
*      RADEFF      inverse of the sum of the wound roll radius and nip
*                  roll radius inverses (length)
*      RADNIP      radius of impinging nip roller (length)
*      RADROLL     radius of wound roll (length)
*      RIGHELEM    number of elements from the right edge of the wound
*                  roll to the right edge of the winding machine
*      REPEAT      decision if FEM should iterate to find deformations
*      ROLPOIEF    combination of Roll's Poisson Ratio squared and 1
*      ROW         loop variable in array that increments the row
*      ROW()       1 d array of resultant values from matrix multiply
*      ROW3        holding variable for third row of matrix multiply
*      STARTFOR    starting force to apply to nip
*      SCRNUMIT    unit number associated with output to the screen
*      STATUS()    1 d visual array showing if an element is gapped
*      STIFF       stiffness of current element under consideration
*      STIFNESS()  1 d array of Winkler foundation stiffnesses
*                  (length^2 per force)
*      STRAIN()    1 d array of material strain data
*      SUM()       1 d array of summations
*      TEMPDISP(,) 2 d temp. holding array of forces and displacements
*      THERE       indicates existence of user's specified INFILE
*      THETAPRN    decision if want to print out rotations
*      TIME1       indicates the number of times CONTACT calls OUTDATA
*      TOPROMPT    indicates if the user needs to be prompted for input
*                  data
*      TOTALDOF   total number of degrees of freedom for the FEM
*      UNITANGL    holds the unit angle symbol
*      UNITFORC    holds the unit force symbol
*      UNITFPL     holds the unit force per length symbol
*      UNITLENG    holds the unit length symbol
*      UNITMOI     holds the unit moment of inertia symbol
*      UNITPRES    holds the unit pressure symbol
*      UNITTYPE    holds the type of unit system to be used, MKS or ENG
*      WINKDISP()  1 d array of FEM winkler foundation (wound roll)
*                  elment displacements obtained as average of
*                  their node displacements
*      WINKESM(,)  2 d array of FEM winkler element stiffness matrix
*      WINKNODE    indicates the array position of the Winkler Found-
*                  ation portion of the current element's nodes
*      WINKWIDT    CMD width of winkler foundation
*      XTRAELEM    extra elements required to accomodate nip roll change
*                  in radius from stub shaft to roller and back
*      X()         temporary 1 d array of FEM nodal displacements
*      Y()         temporary 1 d holding array of nodal forces

```

* Subroutines and Functions:

```

*      BEAMELEM:   determines beam element stiffness matrix
*      CONTACT:    impinges nip beam into winkler roll
*      ELEMNMBR:   finds the number of elements to left and right of wound
*                  roll
*      FINDK1K2:   finds pressure versus strain curve fit parameters KONE
*                  and KTWO
*      HERTZIAN:   generates values of load per width corresponding to
*                  deformations
*      INVALIDES:  gathers input data from user
*      LOADER:     calculates nip load per roll unit width via Hertzian
*                  formula
*      LUDECOMP:   decomposes matrix A into Upper and Lower triangular
*                  matrices
*      MESHER:     establishes the FEM mesh across the width of the rolls

```

```

*   OUTDATA:   writes all data to files
*   PROFILE:   generates the load per width, and max pressure across the
*              wound roll width
*   ROLLERAD:  generates wound roll's radial elastic modulus
*   SUBSTIT:   uses forward and back substitution to solve system LUX=PB
*   WINKELEM:  determines winkler foundation stiffness matrix
*   WINKSTIF:  generates winkler foundation stiffness
*
*****
*
*   DEFINE VARIABLES AND SET PARAMETERS
*
*   INTEGER MAXPTS, MAXELEM, MAXNODES, MAXDATA, MAXARRAY
*   INTEGER NUMOFPTS, NMCROSEL, NMWINKEL, NMCROSND, NMWINKND, NUMGAPS
*   INTEGER MARKER, SCRUNIT, KEYBUNIT, DOFPERND, PERCENT, TOTALDOF
*   INTEGER TIME1, ROW, COLUMN, LEFTELEM, RIGHELEM
*
*   DOUBLE PRECISION RADROLL, POISROLL, RADNIP, NIPROLLE, NIPROLLI
*   DOUBLE PRECISION NIPSTUBE, NIPSTUBI, MAXRADIN, WINKWIDT, STARTFOR
*   DOUBLE PRECISION POISSNIP, ASTART, DELTA, PI, MTRPERIN, NIPFORCE
*   DOUBLE PRECISION RADEFF, DELTAFOR, KONE, KTHREE, NIPROLLW
*   DOUBLE PRECISION NIPEPOIS, AVERLPW, ROLPOIEF, NODEFORC, FORCERES
*   DOUBLE PRECISION NIPSTUBW, LEFTSTUB, LEFTWINK, MAXWIDIN, DISPLRES
*
*   PARAMETER (MAXPTS = 2000, MAXELEM = 96, MAXNODES = 97)
*   PARAMETER (MAXDATA = 500, MAXARRAY = 6*MAXNODES)
*   PARAMETER (SCRUNIT = 6, KEYBUNIT = 5, DOFPERND = 2)
*   PARAMETER (PI=3.141592654D+00, FORCERES = 0.01D+00)
*   PARAMETER (MAXWIDIN=4.80D+01, MTRPERIN = 2.54D-02)
*   PARAMETER (MAXRADIN = 1.0D+02, DISPLRES = 1.0D-05)
*
*   INTEGER CODEBEAM(MAXELEM), CODEWINK(MAXELEM)
*   DOUBLE PRECISION NODEDISP(MAXARRAY), TEMPDISP(MAXARRAY,6)
*   DOUBLE PRECISION DEFORM(MAXPTS), LOADPERW(MAXPTS)
*   DOUBLE PRECISION HAFWIDTH(MAXPTS), STIFNESS(MAXPTS)
*   DOUBLE PRECISION MAXPRESS(MAXNODES), LPWVSND(MAXNODES)
*   DOUBLE PRECISION HALFCONW(MAXNODES), LOCATE(MAXNODES)
*   DOUBLE PRECISION WINKDISP(MAXELEM), ELLENGTH(MAXELEM)
*
*   CHARACTER DATE*20, CONDITON(MAXELEM)*7, STATUS(MAXELEM)*2
*   CHARACTER UNITTYPE*3, UNITLENG*10, UNITFORC*10, UNITFPL*10
*   CHARACTER UNITPRES*10, UNITMOI*10, UNITANGL*10, THETAPRN*1
*   CHARACTER CONSTLFT*1, CONSTRGH*1, REPEAT*1
*
*   INITIALIZE VARIABLES
*
*   INTEGERS
*   NUMOFPTS = 0
*   NMCROSEL = 0
*   NMWINKEL = 0
*   NMCROSND = 0
*   NMWINKND = 0
*   NUMGAPS = 0
*   PERCENT = 0
*   TIME1 = 1
*   LEFTELEM = 0
*   RIGHELEM = 0
*
*   DOUBLE PRECISION
*   RADROLL = 0.0D+00
*   POISROLL = 0.0D+00
*   RADNIP = 0.0D+00

```

```

NIPROLLE = 0.0D+00
NIPROLLI = 0.0D+00
NIPSTUBE = 0.0D+00
NIPSTUBI = 0.0D+00
POISSNIP = 0.0D+00
ASTART   = 0.0D+00
ADELTA   = 0.0D+00
STARTFOR = 0.0D+00
NIPFORCE = 0.0D+00
DELTAFOR = 0.0D+00
KONE     = 0.0D+00
KTWO    = 0.0D+00
KTHREE  = 0.0D+00
NIPROLLW = 0.0D+00
NIPEPOIS = 0.0D+00
AVERLPW = 0.0D+00
RADEFF  = 0.0D+00
ROLPOIEF = 0.0D+00
NODEFORC = 0.0D+00
WINKWIDT = 0.0D+00
NIPSTUBW = 0.0D+00
LEFTSTUB = 0.0D+00
LEFTWINK = 0.0D+00
*
* ARRAY
DO 100 ROW = 1, MAXARRAY
  NODEDISP(ROW) = 0.0D+00
  DO 110 COLUMN = 1, 6
    TEMPDISP (ROW,COLUMN) = 0.0D+00
110  CONTINUE
100  CONTINUE
DO 120 ROW = 1, MAXELEM
  ELLENGTH(ROW) = 0.0D+00
  WINKDISP(ROW) = 0.0D+00
  CODEBEAM(ROW) = 2
  CODEWINK(ROW) = 2
120  CONTINUE
DO 130 ROW = 1, MAXNODES
  LOCATE(ROW) = 0.0D+00
  LPWVSNOD(ROW) = 0.0D+00
  MAXPRESS(ROW) = 0.0D+00
  HALFCONW(ROW) = 0.0D+00
130  CONTINUE
DO 140 ROW = 1, MAXPTS
  DEFORM (ROW) = 0.0D+00
  LOADPERW(ROW) = 0.0D+00
  HAFWIDTH(ROW) = 0.0D+00
  STIFNESS(ROW) = 0.0D+00
140  CONTINUE
*
* CHARACTER
DATE = 'JANUARY 1, 1900'
UNITTYPE = '****'
UNITLENG = 'NO LENGTH'
UNITFORC = 'NO FORCE'
UNITFPL = 'NO FPLENG'
UNITPRES = 'NO PRESSRE'
UNITMOI = 'NO MOI'
UNITANGL = 'NO ANGLE'
THETAPRN = '*'
CONSTLFT = '*'
CONSTRGH = '*'

```

```

REPEAT = '*'
DO 150 ROW = 1, MAXELEM
  CONDITON(ROW) = 'NOT SET'
  STATUS (ROW) = '***'
150 CONTINUE
*
*
* INPUT DATA
*
MARKER = 2
CALL INVALUES(SCRNUNIT, KEYBUNIT, MAXELEM, MAXNODES, DATE,
+ RADROLL, POISSROLL, RADNIP, NIPROLLE, NIPROLLI, POISSNIP,
+ DELTA, NUMOFPTS, KONE, KTWO, NIPFORCE, DELTAFOR, NMCROSEL,
+ NMCROSND, ELLENGTH, NIPROLLW, MAXDATA, UNITTYPE, UNITLENG,
+ UNITFORC, UNITFPL, UNITPRES, UNITMOI, UNITANGL, RADEFF,
+ LOCATE, MAXWIDIN, MTRPERIN, NMWINKEL, NMWINKND, CODEBEAM,
+ CODEWINK, NIPSTUBE, NIPSTUBI, ASTART, LEFTELEM, MAXRADIN,
+ FORCERES, RIGHELEM, WINKWIDT, NIPSTUBW, LEFTSTUB, LEFTWINK,
+ KTHREE, THETAPRN, CONSTLFT, CONSTRGH, STARTFOR, REPEAT)
*
*
* INITIALIZE OUTPUT FILES
*
CALL OUTDATA(MARKER, DATE, KONE, KTWO, RADROLL, POISSROLL, RADNIP,
+ POISSNIP, NIPROLLE, AVERLPW, NIPROLLI, NMCROSEL, NIPFORCE,
+ NIPROLLW, NODEFORC, NUMGAPS, PERCENT, NUMOFPTS, HAFWIDTH,
+ DEFORM, STIFNESS, LOADPERW, CONDITON, STATUS, NMCROSND,
+ NODEDISP, DOFFPERND, TIME1, DELTAFOR, WINKDISP, LPWVSNOD,
+ MAXPRESS, UNITTYPE, UNITLENG, UNITFORC, UNITFPL, UNITPRES,
+ UNITMOI, UNITANGL, RADEFF, TEMPDISP, HALFCONW, LOCATE,
+ CODEBEAM, NIPSTUBW, LEFTSTUB, LEFTWINK, NIPSTUBE, NIPSTUBI,
+ WINKWIDT, NMWINKEL, CODEWINK, LEFTELEM, KTHREE, THETAPRN,
+ CONSTLFT, CONSTRGH, STARTFOR, REPEAT)
*
*
* DETERMINE HERTZIAN LOADS PER WIDTH VERSUS DEFORMATION
*
WRITE (SCRNUNIT,*) 'DETERMINING GENERIC LOADS PER WIDTH...'
MARKER = 3
CALL HERTZIAN(NUMOFPTS, HAFWIDTH, DEFORM, LOADPERW, NIPROLLE, PI,
+ POISSNIP, POISSROLL, ASTART, DELTA, KONE, KTWO, KTHREE,
+ ROLPOIEF, NIPEPOIS, STIFNESS, RADEFF)
*
*
* PRINT OUT HERTZIAN LOADS PER WIDTH
*
CALL OUTDATA(MARKER, DATE, KONE, KTWO, RADROLL, POISSROLL, RADNIP,
+ POISSNIP, NIPROLLE, AVERLPW, NIPROLLI, NMCROSEL, NIPFORCE,
+ NIPROLLW, NODEFORC, NUMGAPS, PERCENT, NUMOFPTS, HAFWIDTH,
+ DEFORM, STIFNESS, LOADPERW, CONDITON, STATUS, NMCROSND,
+ NODEDISP, DOFFPERND, TIME1, DELTAFOR, WINKDISP, LPWVSNOD,
+ MAXPRESS, UNITTYPE, UNITLENG, UNITFORC, UNITFPL, UNITPRES,
+ UNITMOI, UNITANGL, RADEFF, TEMPDISP, HALFCONW, LOCATE,
+ CODEBEAM, NIPSTUBW, LEFTSTUB, LEFTWINK, NIPSTUBE, NIPSTUBI,
+ WINKWIDT, NMWINKEL, CODEWINK, LEFTELEM, KTHREE, THETAPRN,
+ CONSTLFT, CONSTRGH, STARTFOR, REPEAT)
*
*
* CALCULATE THE TOTAL NUMBER OF DEGREES OF FREEDOM
*
TOTALDOF = (NMCROSND + (NMWINKND * 2)) * DOFFPERND
*

```

```

*
*   INCREMENTALLY IMPINGE NIP BEAM INTO WINKLER ROLL
*
WRITE (SCRNUNIT,*) 'IMPINGING NIP ROLL INTO WOUND ROLL...'
MARKER = 4
CALL CONTACT(MARKER, NMCROSND, NMCROSEL, DOFFPERND, PERCENT,
+   KONE, KTW0, FORCERES, NIPFORCE, NIPEPOIS, PI, NODEFORC,
+   DELTAFOR, ROLPOIEF, NODEDISP, ELLENGTH, CONDITON, STATUS,
+   NUMGAPS, TOTALDOF, DATE, RADROLL, POISROLL, RADNIP, POISSNIP,
+   NIPROLLE, AVERLPW, NIPROLLI, NIPROLLW, NUMOFPTS, HAFWIDTH,
+   DEFORM, STIFNESS, LOADPERW, TIME1, WINKDISP, LPWVSOD,
+   MAXPRESS, UNITTYPE, UNITLENG, UNITFORC, UNITFPL, UNITPRES,
+   UNITMOI, UNITANGL, RADEFF, TEMPDISP, HALFCONW, CODEBEAM,
+   CODEWINK, NIPSTUBE, NIPSTUBI, LOCATE, LEFTELEM, NMWINKND,
+   NMWINKEL, KTHREE, THETAPRN, CONSTLFT, CONSTRGH, STARTFOR,
+   REPEAT, DISPLRES)
*
*
*   DETERMINE WOUND ROLL'S CONTACT PROFILE
*
WRITE (SCRNUNIT,*) 'CALCULATING WOUND ROLL PROFILE...'
MARKER = 5
CALL PROFILE(NMCROSEL, NMCROSND, DOFFPERND, PI, RADEFF,
+   WINKWIDT, NODEDISP, LPWVSOD, MAXPRESS, ELLENGTH, NODEFORC,
+   AVERLPW, HALFCONW, CODEBEAM, CODEWINK, NIPSTUBI, NIPSTUBE,
+   NIPROLLE, NIPROLLI, LEFTELEM)
*
*
*   PRINT OUT CONTACT PROFILE
*
CALL OUTDATA(MARKER, DATE, KONE, KTW0, RADROLL, POISROLL, RADNIP,
+   POISSNIP, NIPROLLE, AVERLPW, NIPROLLI, NMCROSEL, NIPFORCE,
+   NIPROLLW, NODEFORC, NUMGAPS, PERCENT, NUMOFPTS, HAFWIDTH,
+   DEFORM, STIFNESS, LOADPERW, CONDITON, STATUS, NMCROSND,
+   NODEDISP, DOFFPERND, TIME1, DELTAFOR, WINKDISP, LPWVSOD,
+   MAXPRESS, UNITTYPE, UNITLENG, UNITFORC, UNITFPL, UNITPRES,
+   UNITMOI, UNITANGL, RADEFF, TEMPDISP, HALFCONW, LOCATE,
+   CODEBEAM, NIPSTUBW, LEFTSTUB, LEFTWINK, NIPSTUBE, NIPSTUBI,
+   WINKWIDT, NMWINKEL, CODEWINK, LEFTELEM, KTHREE, THETAPRN,
+   CONSTLFT, CONSTRGH, STARTFOR, REPEAT)
*
*
*   TERMINATE PROGRAM RUN
*
WRITE (SCRNUNIT,*) 'PROGRAM RUN SUCCESSFUL, NOW QUITING...'
STOP
END
*
*****
*****
*
*   INVALUES, GATHERS INPUT DATA FROM USER
*
*****
*
SUBROUTINE INVALUES(SCRNUNIT, KEYBUNIT, MAXELEM, MAXNODES, DATE,
+   RADROLL, POISROLL, RADNIP, NIPROLLE, NIPROLLI, POISSNIP,
+   ADELTA, NUMOFPTS, KONE, KTW0, NIPFORCE, DELTAFOR, NMCROSEL,
+   NMCROSND, ELLENGTH, NIPROLLW, MAXDATA, UNITTYPE, UNITLENG,
+   UNITFORC, UNITFPL, UNITPRES, UNITMOI, UNITANGL, RADEFF,
+   LOCATE, MAXWIDIN, MTRPERIN, NMWINKEL, NMWINKND, CODEBEAM,
+   CODEWINK, NIPSTUBE, NIPSTUBI, ASTART, LEFTELEM, MAXRADIN,

```



```

+   FORCERES, RIGHELEM, WINKWIDT, NIPSTUBW, LEFTSTUB, LEFTWINK,
+   KTHREE, THETAPRN, CONSTLFT, CONSTRGH, STARTFOR, REPEAT)
*
*   DEFINE VARIABLES
*
INTEGER NUMOFPTS, INERROR, LOOP, NMCROSEL, NMWINKEL, NMCROSND
INTEGER MAXELEM, MAXNODES, MAXDATA, RIGHELEM, XTRAELEM
INTEGER NMWINKND, CODEBEAM(MAXELEM), CODEWINK(MAXELEM), LEFTELEM
INTEGER SCRUNIT, KEYBUNIT, INPTUNIT, PRESUNIT, HOLDINUN
DOUBLE PRECISION RADROLL, POISROLL, RADNIP, NIPROLLE, NIPROLLI
DOUBLE PRECISION NIPSTUBE, NIPSTUBI, POISSNIP, ASTART, DELTA
DOUBLE PRECISION KONE, KTW, KTHREE, NIPSTUBW, LEFTSTUB, LEFTWINK
DOUBLE PRECISION NIPFORCE, DELTAFOR, RADEFF, MAXWIDIN, MTRPERIN
DOUBLE PRECISION ALOWDWID, NIPROLLW, ELLENGTH(MAXELEM), MAXRADIN
DOUBLE PRECISION LOCATE(MAXNODES), WINKWIDT, FORCERES, STARTFOR
CHARACTER INFILE*80, DATE*20, ERRORCHR*8, DECISON1*1, DECISON2*1
CHARACTER UNITTYPE*3, UNITLENG*10, UNITFORC*10, UNITFPL*10
CHARACTER UNITPRES*10, UNITMOI*10, UNITANGL*10, PRESRDAT*80
CHARACTER THETAPRN*1, CONSTLFT*1, CONSTRGH*1, REPEAT*1
LOGICAL THERE, TOPPROMPT, PROCEED, HOLDPRMT
*
*   INITIALIZE VARIABLES
*
THERE = .TRUE.
TOPPROMPT = .TRUE.
PROCEED = .FALSE.
INERROR = 0
XTRAELEM = 0
PRESUNIT = 20
*
*   PRINT OUT INTRODUCTION
*
*   CLEAR PRINT SCREEN
*
DO 200 LOOP = 1, 40
  WRITE (SCRUNIT,2000)
200 CONTINUE
*
*   PRINT OUT HEADER
*
WRITE (SCRUNIT,2010)
WRITE (SCRUNIT,2000)
WRITE (SCRUNIT,2020)
WRITE (SCRUNIT,2030)
WRITE (SCRUNIT,2000)
WRITE (SCRUNIT,2040)
WRITE (SCRUNIT,2050)
WRITE (SCRUNIT,2000)
WRITE (SCRUNIT,2920)
READ (KEYBUNIT,2925) DECISON1
IF (DECISON1.EQ.'') THEN
  PROCEED = .TRUE.
  DECISON1 = 'N'
  TOPPROMPT = .FALSE.
  INPTUNIT = 10
  INFILE = 'DATASAMP.IN'
  OPEN (INPTUNIT, FILE = INFILE, STATUS = 'OLD')
ELSE IF (DECISON1.EQ.'$') THEN
  DECISON1 = 'Y'
ELSE
  DECISON1 = 'N'
ENDIF

```

```

WRITE (SCRNUNIT,2000)
WRITE (SCRNUNIT,2060)
WRITE (SCRNUNIT,2070)
WRITE (SCRNUNIT,2080)
WRITE (SCRNUNIT,2090)
WRITE (SCRNUNIT,2100)
WRITE (SCRNUNIT,2110)
WRITE (SCRNUNIT,2120)
WRITE (SCRNUNIT,2130)
WRITE (SCRNUNIT,2140)
WRITE (SCRNUNIT,2000)
WRITE (SCRNUNIT,2150)
WRITE (SCRNUNIT,2000)
WRITE (SCRNUNIT,2160)
WRITE (SCRNUNIT,2000)
WRITE (SCRNUNIT,2200)
WRITE (SCRNUNIT,2000)
WRITE (SCRNUNIT,2010)
*
* INPUT FILE NAME, CHECK FOR EXISTENCE, IF DOESN'T EXIST...LOOP THROUGH
* UP TO 3 TIMES TO ALLOW USER TO ALTER THE PATH OR FILE NAME
*
DO WHILE (.NOT.PROCEED)
  WRITE (SCRNUNIT,2000)
  WRITE (SCRNUNIT,2000)
  WRITE (SCRNUNIT,2250)
  READ (KEYBUNIT,2900) INFILE
*
* CHECK IF USER WANTS TO BE PROMPTED
*
  IF (INFILE.EQ.'NONE'.OR.INFILE.EQ.'none'.OR.INERROR.GE.3) THEN
    PROCEED = .TRUE.
    TOPROMPT = .TRUE.
    INPTUNIT = KEYBUNIT
  ELSE
    INQUIRE (FILE = INFILE, EXIST=THEIR)
    IF (THEIR) THEN
      PROCEED = .TRUE.
      TOPROMPT = .FALSE.
      INPTUNIT = 10
      OPEN (INPTUNIT, FILE = INFILE, STATUS = 'OLD')
    ELSE
      PROCEED = .FALSE.
      TOPROMPT = .TRUE.
      WRITE (SCRNUNIT,2000)
      WRITE (SCRNUNIT,2260) INFILE
      WRITE (SCRNUNIT,2000)
      INERROR = INERROR + 1
    ENDIF
  ENDIF
END DO
*
* SET HOLDING VARIABLES
*
HOLDPRMT = TOPROMPT
HOLDINUN = INPTUNIT
*
* ENTER CURRENT DATE
*
* PROMPT IF NEEDED
IF (TOPROMPT) THEN
  WRITE (SCRNUNIT,2000)

```



```

        WRITE (SCRNUNIT,2270)
    ENDIF
*   INPUT VALUE
    READ (INPTUNIT,2910) DATE
*
*   ENTER UNIT SYSTEM TO USE
*
    PROCEED = .FALSE.
    DO WHILE (.NOT.PROCEED)
*   PROMPT IF NEEDED
        IF (TOPROMPT) THEN
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2280)
        ENDIF
*   INPUT VALUE
        READ (INPTUNIT,2911) UNITTYPE
*
*   SET UNIT STANDARDS
*
        IF (UNITTYPE.EQ.'MKS'.OR.UNITTYPE.EQ.'mks') THEN
            UNITLENG = '(m)'
            UNITFORC = '(N)'
            UNITFPL = '(N/ m)'
            UNITPRES = '(N/ m^2)'
            UNITMOI = '(m^4)'
            UNITANGL = '(radian)'
            PROCEED = .TRUE.
            TOPROMPT = HOLDPRMT
            INPTUNIT = HOLDINUN
            ALOWDWID = MAXWIDIN*MTRPERIN
            ALOWDRAD = MAXRADIN*MTRPERIN
        ELSE IF (UNITTYPE.EQ.'ENG'.OR.UNITTYPE.EQ.'eng') THEN
            UNITLENG = '(in)'
            UNITFORC = '(Lb)'
            UNITFPL = '(Lb/ in)'
            UNITPRES = '(Lb/ in^2)'
            UNITMOI = '(in^4)'
            UNITANGL = '(radian)'
            PROCEED = .TRUE.
            TOPROMPT = HOLDPRMT
            INPTUNIT = HOLDINUN
            ALOWDWID = MAXWIDIN
            ALOWDRAD = MAXRADIN
        ELSE
            ERRORCHR = 'UNITTYPE'
            WRITE (SCRNUNIT,2660) ERRORCHR
            TOPROMPT = .TRUE.
            INPTUNIT = KEYBUNIT
        ENDIF
    END DO
*
*   ENTER NIP ROLLER RADIUS
*
    PROCEED = .FALSE.
    DO WHILE (.NOT.PROCEED)
*   PROMPT IF NEEDED
        IF (TOPROMPT) THEN
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2290)
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2300) UNITLENG
        ENDIF

```

```

*      INPUT VALUE
      READ (INPTUNIT,*) RADNIP
*      CHECK ACCEPTABILITY OF INPUT VALUE
      IF (RADNIP.GT.0.0D+00.AND.RADNIP.LT.ALOWDRAD) THEN
        PROCEED = .TRUE.
        TOPROMPT = HOLDPRMT
        INPTUNIT = HOLDINUN
      ELSE
        ERRORCHR = 'RADNIP'
        WRITE (SCRNUNIT,2660) ERRORCHR
        TOPROMPT = .TRUE.
        INPTUNIT = KEYBUNIT
      ENDIF
    END DO

*
*      ENTER THE CMD DISTANCE BETWEEN THE APPLIED LOAD LOCATIONS
*      NIP ROLL WITH STUB SHAFTS OVERALL WIDTH
*

    PROCEED = .FALSE.
    DO WHILE (.NOT.PROCEED)
*      PROMPT IF NEEDED
      IF (TOPROMPT) THEN
        WRITE (SCRNUNIT,2000)
        WRITE (SCRNUNIT,2310) UNITLENG
      ENDIF
*      INPUT VALUE
      READ (INPTUNIT,*) NIPSTUBW
*      CHECK ACCEPTABILITY OF INPUT VALUE
      IF (NIPSTUBW.GT.0.0D+00.AND.NIPSTUBW.LE.ALOWDWID) THEN
        PROCEED = .TRUE.
        TOPROMPT = HOLDPRMT
        INPTUNIT = HOLDINUN
      ELSE
        ERRORCHR = 'NIPSTUBW'
        WRITE (SCRNUNIT,2660) ERRORCHR
        TOPROMPT = .TRUE.
        INPTUNIT = KEYBUNIT
      ENDIF
    END DO

*
*      ENTER THE CMD DISTANCE FROM ZERO TO NIP ROLLER
*

    PROCEED = .FALSE.
    DO WHILE (.NOT.PROCEED)
*      PROMPT IF NEEDED
      IF (TOPROMPT) THEN
        WRITE (SCRNUNIT,2000)
        WRITE (SCRNUNIT,2320) UNITLENG
      ENDIF
*      INPUT VALUE
      READ (INPTUNIT,*) LEFTSTUB
*      CHECK ACCEPTABILITY OF INPUT VALUE
      IF (LEFTSTUB.GE.0.0D+00.AND.LEFTSTUB.LE.ALOWDWID) THEN
        PROCEED = .TRUE.
        TOPROMPT = HOLDPRMT
        INPTUNIT = HOLDINUN
      ELSE
        ERRORCHR = 'LEFTSTUB'
        WRITE (SCRNUNIT,2660) ERRORCHR
        TOPROMPT = .TRUE.
        INPTUNIT = KEYBUNIT
      ENDIF
    END DO

```

```

END DO
*
* ENTER THE NIP ROLLER CMD WIDTH
*
PROCEED = .FALSE.
DO WHILE (.NOT.PROCEED)
*   PROMPT IF NEEDED
*   IF (TOPROMPT) THEN
*       WRITE (SCRNUNIT,2000)
*       WRITE (SCRNUNIT,2330) UNITLENG
*   ENDIF
*   INPUT VALUE
*   READ (INPTUNIT,*) NIPROLLW
*   CHECK ACCEPTABILITY OF INPUT VALUE
*   IF (NIPROLLW.GT.0.0D+00.AND.
+   NIPROLLW.LE.(ALOWDWID-LEFTSTUB)) THEN
*       PROCEED = .TRUE.
*       TOPROMPT = HOLDPRMT
*       INPTUNIT = HOLDINUN
*   ELSE
*       ERRORCHR = 'NIPROLLW'
*       WRITE (SCRNUNIT,2660) ERRORCHR
*       TOPROMPT = .TRUE.
*       INPTUNIT = KEYBUNIT
*   ENDIF
END DO
*
* ENTER WOUND ROLL (WINKLER FOUNDATION ROLL) RADIUS
*
PROCEED = .FALSE.
DO WHILE (.NOT.PROCEED)
*   PROMPT IF NEEDED
*   IF (TOPROMPT) THEN
*       WRITE (SCRNUNIT,2000)
*       WRITE (SCRNUNIT,2340)
*       WRITE (SCRNUNIT,2000)
*       WRITE (SCRNUNIT,2350) UNITLENG
*   ENDIF
*   INPUT VALUE
*   READ (INPTUNIT,*) RADROLL
*   CHECK ACCEPTABILITY OF INPUT VALUE
*   IF (RADROLL.GT.0.0D+00.AND.RADROLL.LT.ALLOWDRAD) THEN
*       PROCEED = .TRUE.
*       TOPROMPT = HOLDPRMT
*       INPTUNIT = HOLDINUN
*   ELSE
*       ERRORCHR = 'RADROLL'
*       WRITE (SCRNUNIT,2660) ERRORCHR
*       TOPROMPT = .TRUE.
*       INPTUNIT = KEYBUNIT
*   ENDIF
END DO
*
* DETERMINE EFFECTIVE RADIUS
*
RADEFF = 1.0D+00/((1.0D+00/RADROLL) + (1.0D+00/RADNIP))
*
* ENTER THE CMD DISTANCE FROM ZERO TO THE WOUND ROLL
* (WINKLER FOUNDATION)
*
PROCEED = .FALSE.
DO WHILE (.NOT.PROCEED)

```

```

*       PROMPT IF NEEDED
        IF (TOPROMPT) THEN
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2360) UNITLENG
        ENDIF
*       INPUT VALUE
        READ (INPTUNIT,*) LEFTWINK
*       CHECK ACCEPTABILITY OF INPUT VALUE
        IF (LEFTWINK.GE.0.0D+00.AND.LEFTWINK.LE.ALLOWDWID) THEN
            PROCEED = .TRUE.
            TOPROMPT = HOLDPRMT
            INPTUNIT = HOLDINUN
        ELSE
            ERRORCHR = 'LEFTWINK'
            WRITE (SCRNUNIT,2660) ERRORCHR
            TOPROMPT = .TRUE.
            INPTUNIT = KEYBUNIT
        ENDIF
    END DO

*
*       ENTER THE CMD WIDTH OF THE WOUND ROLL
*       (WINKLER FOUNDATION)
*
    PROCEED = .FALSE.
    DO WHILE (.NOT.PROCEED)
*       PROMPT IF NEEDED
        IF (TOPROMPT) THEN
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2370) UNITLENG
        ENDIF
*       INPUT VALUE
        READ (INPTUNIT,*) WINKWIDT
*       CHECK ACCEPTABILITY OF INPUT VALUE
        IF (WINKWIDT.GT.0.0D+00.AND.
+       WINKWIDT.LE.(ALLOWDWID-LEFTWINK).AND.
+       (WINKWIDT+LEFTWINK).GT.LEFTSTUB) THEN
            PROCEED = .TRUE.
            TOPROMPT = HOLDPRMT
            INPTUNIT = HOLDINUN
        ELSE
            ERRORCHR = 'WINKWIDT'
            WRITE (SCRNUNIT,2660) ERRORCHR
            TOPROMPT = .TRUE.
            INPTUNIT = KEYBUNIT
        ENDIF
    END DO

*
*       ENTER NIP ROLLER MODULUS OF ELASTICITY
*
    PROCEED = .FALSE.
    DO WHILE (.NOT.PROCEED)
*       PROMPT IF NEEDED
        IF (TOPROMPT) THEN
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2380)
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2400) UNITPRES
        ENDIF
*       INPUT VALUE
        READ (INPTUNIT,*) NIPROLLE
*       CHECK ACCEPTABILITY OF INPUT VALUE
        IF (NIPROLLE.GT.1.0D+05) THEN

```

```

        PROCEED = .TRUE.
        TOPROMPT = HOLDPRMT
        INPTUNIT = HOLDINUN
    ELSE
        ERRORCHR = 'NIPROLLE'
        WRITE (SCRNUNIT,2660) ERRORCHR
        TOPROMPT = .TRUE.
        INPTUNIT = KEYBUNIT
    ENDIF
END DO
*
*
*
ENTER NIP ROLLER MOMENT OF INERTIA
*
*
*
PROCEED = .FALSE.
DO WHILE (.NOT.PROCEED)
*
    PROMPT IF NEEDED
    IF (TOPROMPT) THEN
        WRITE (SCRNUNIT,2000)
        WRITE (SCRNUNIT,2410) UNITMOI
    ENDIF
*
    INPUT VALUE
    READ (INPTUNIT,*) NIPROLLI
*
    CHECK ACCEPTABILITY OF INPUT VALUE
    IF (NIPROLLI.GT.0.0D+00) THEN
        PROCEED = .TRUE.
        TOPROMPT = HOLDPRMT
        INPTUNIT = HOLDINUN
    ELSE
        ERRORCHR = 'NIPROLLI'
        WRITE (SCRNUNIT,2660) ERRORCHR
        TOPROMPT = .TRUE.
        INPTUNIT = KEYBUNIT
    ENDIF
END DO
*
*
*
ENTER NIP ROLLER POISSONS RATIO
*
*
*
PROCEED = .FALSE.
DO WHILE (.NOT.PROCEED)
*
    PROMPT IF NEEDED
    IF (TOPROMPT) THEN
        WRITE (SCRNUNIT,2000)
        WRITE (SCRNUNIT,2420)
    ENDIF
*
    INPUT VALUE
    READ (INPTUNIT,*) POISSNIP
*
    CHECK ACCEPTABILITY OF INPUT VALUE
    IF (POISSNIP.GT.0.0D+00 .AND. POISSNIP.LT.3.0D+00) THEN
        PROCEED = .TRUE.
        TOPROMPT = HOLDPRMT
        INPTUNIT = HOLDINUN
    ELSE
        ERRORCHR = 'POISSNIP'
        WRITE (SCRNUNIT,2660) ERRORCHR
        TOPROMPT = .TRUE.
        INPTUNIT = KEYBUNIT
    ENDIF
END DO
*
*
*
ENTER NIP ROLLER STUB SHAFT MODULUS OF ELASTICITY
*
*
*
PROCEED = .FALSE.

```

```

DO WHILE (.NOT.PROCEED)
*   PROMPT IF NEEDED
   IF (TOPPROMPT) THEN
       WRITE (SCRNUNIT,2000)
       WRITE (SCRNUNIT,2430) UNITPRES
   ENDIF
*   INPUT VALUE
   READ (INPTUNIT,*) NIPSTUBE
*   CHECK ACCEPTABILITY OF INPUT VALUE
   IF (NIPSTUBE.GT.1.0D+03) THEN
       PROCEED = .TRUE.
       TOPPROMPT = HOLDPRMT
       INPTUNIT = HOLDINUN
   ELSE
       ERRORCHR = 'NIPSTUBE'
       WRITE (SCRNUNIT,2660) ERRORCHR
       TOPPROMPT = .TRUE.
       INPTUNIT = KEYBUNIT
   ENDIF
END DO

*
*   ENTER NIP ROLLER STUB SHAFT MOMENT OF INERTIA
*
PROCEED = .FALSE.
DO WHILE (.NOT.PROCEED)
*   PROMPT IF NEEDED
   IF (TOPPROMPT) THEN
       WRITE (SCRNUNIT,2000)
       WRITE (SCRNUNIT,2440) UNITMOI
   ENDIF
*   INPUT VALUE
   READ (INPTUNIT,*) NIPSTUBI
*   CHECK ACCEPTABILITY OF INPUT VALUE
   IF (NIPSTUBI.GT.0.0D+00) THEN
       PROCEED = .TRUE.
       TOPPROMPT = HOLDPRMT
       INPTUNIT = HOLDINUN
   ELSE
       ERRORCHR = 'NIPSTUBI'
       WRITE (SCRNUNIT,2660) ERRORCHR
       TOPPROMPT = .TRUE.
       INPTUNIT = KEYBUNIT
   ENDIF
END DO

*
*   ENTER WOUND ROLL (WINKLER FOUNDATION ROLL) POISSONS RATIO
*
PROCEED = .FALSE.
DO WHILE (.NOT.PROCEED)
*   PROMPT IF NEEDED
   IF (TOPPROMPT) THEN
       WRITE (SCRNUNIT,2000)
       WRITE (SCRNUNIT,2450)
       WRITE (SCRNUNIT,2000)
       WRITE (SCRNUNIT,2460)
   ENDIF
*   INPUT VALUE
   READ (INPTUNIT,*) POISROLL
*   CHECK ACCEPTABILITY OF INPUT VALUE
   IF (POISROLL.GE.0.0D+00 .AND. POISROLL.LT.3.0D+00) THEN
       PROCEED = .TRUE.
       TOPPROMPT = HOLDPRMT

```

```

        INPTUNIT = HOLDINUN
    ELSE
        ERRORCHR = 'POISROLL'
        WRITE (SCRNUNIT,2660) ERRORCHR
        TOPROMPT = .TRUE.
        INPTUNIT = KEYBUNIT
    ENDIF
END DO
*
*
* GET KONE, KTW0, AND KTHREE: ALLOW USER TO ENTER THEM DIRECTLY
* OR TO ENTER THE DATA FILE CONTAINING PRESSURE VERSUS DEFORMATION
* AND GO TO FINDK1K2 ROUTINE
*
*
INERROR = 0
PROCEED = .FALSE.
DO WHILE (.NOT.PROCEED)
*
* ALL MATERIAL WITH XXX AROUND IT WILL ALLOW USERS TO HAVE
* MACHINE DETERMINE K1 AND K2 WHEN THE REMARKS ARE ELIMINATED
*
* XXX
* **
*   PROMPT IF NEEDED
*   IF (TOPROMPT) THEN
*       WRITE (SCRNUNIT,2000)
*       WRITE (SCRNUNIT,2630)
*       WRITE (SCRNUNIT,2640)
*   ENDIF
*   XXX
*   CHECK DECISION...IF USER WANTS COMPUTER TO DETERMINE K1 AND K2
*   AN '*' MEANS THE LOOP IS RETURNING AFTER AN UNSUCCESSFUL TRY
*   TO FIT KONE AND KTW0 AND THEY MUST NOW BE INPUT DIRECTLY
*   IF (DECISON1.EQ.'*') THEN
*       DECISON1 = 'N'
*
*   XXX
*   ELSE
*       READ (INPTUNIT, 2920) DECISON1
*   XXX
*   ENDIF
*
*
*   USE COMPUTER TO DETERMINE K1 AND K2
*
*   IF (DECISON1.EQ.'Y'.OR.DECISON1.EQ.'y') THEN
*
*       INPUT FILE NAME, CHECK FOR EXISTENCE, IF DOESN'T EXIST...LOOP
*       THROUGH UP TO 3 TIMES TO ALLOW USER TO ALTER THE PATH OR FILE
*       NAME. IF DOES EXIST DETERMINE K1 AND K2.
*
*       IF (TOPROMPT) THEN
*           WRITE (SCRNUNIT,2000)
*           WRITE (SCRNUNIT,2160)
*           WRITE (SCRNUNIT,2170)
*           WRITE (SCRNUNIT,2180)
*           WRITE (SCRNUNIT,2190)
*           WRITE (SCRNUNIT,2000)
*           WRITE (SCRNUNIT,2650)
*       ENDIF
*       READ (INPTUNIT,2900) PRESRDAT
*
*
*       CHECK IF FILE EXISTS
*
*       IF (INERROR.GE.3) THEN
*           WRITE (SCRNUNIT,2000)
*           WRITE (SCRNUNIT,2690)

```

```

        DECISON1 = 'N'
ELSE
    INQUIRE (FILE = PRESRDAT, EXIST=THEIR)
    IF (THEIR) THEN
        PROCEED = .TRUE.
*
*       DETERMINE K1, K2, K3; OPEN AND CLOSE PRESRDAT INPUT
*       FILE, CALL FINDK1K2 ROUTINE, AND SET K3=0.0D+00
*
        OPEN (UNIT = PRESUNIT, FILE = PRESRDAT, STATUS =
+           'OLD')
+       CALL FINDK1K2 (PRESUNIT, SCRUNIT, KONE, KTWO,
        MAXDATA)
        CLOSE (UNIT = PRESUNIT)
        KTHREE = 0.0D+00
*
*       SET FLAGS IF USER NEEDS TO INPUT KONE, KTWO, AND
*       KTHREE DIRECTLY
*
        IF (KONE.LE.0.0D+00.AND.KTWO.LE.0.0D+00) THEN
            PROCEED = .FALSE.
            TOPROMPT = .TRUE.
            INPTUNIT = KEYBUNIT
            WRITE (SCRUNIT,2000)
            WRITE (SCRUNIT,2690)
            DECISON1 = '*'
        ELSE
            WRITE (SCRUNIT,2670) KONE, UNITPRES, KTWO
            WRITE (SCRUNIT,2680) KTHREE, UNITPRES
        ENDIF
*
    ELSE
        PROCEED = .FALSE.
        WRITE (SCRUNIT,2000)
        WRITE (SCRUNIT,2260) PRESRDAT
        WRITE (SCRUNIT,2000)
        INERROR = INERROR + 1
    ENDIF
ENDIF
*
*
*       INPUT K1, K2, AND K3 DIRECTLY
*
ELSEIF (DECISON1.EQ.'N'.OR.DECISON1.EQ.'n') THEN
*
*       INPUT K1
*
        PROCEED = .FALSE.
        DO WHILE (.NOT.PROCEED)
*           PROMPT IF NEEDED
            IF (TOPROMPT) THEN
                WRITE (SCRUNIT,2000)
                WRITE (SCRUNIT,2470) UNITPRES
            ENDIF
            INPUT K1
            READ (INPTUNIT,*) KONE
            CHECK ACCEPTABILITY OF USER INPUT K1
            IF (KONE.GT.0.0D+00) THEN
                PROCEED = .TRUE.
            ELSE
                ERRORCHR = 'KONE'
                WRITE (SCRUNIT,2660) ERRORCHR
            ENDIF
        END DO
    ENDIF

```



```

        ENDIF
    END DO
*
*
*
    INPUT K2
*
*
*
    PROCEED = .FALSE.
    DO WHILE (.NOT.PROCEED)
*
*
*
        PROMPT IF NEEDED
        IF (TOPROMPT) THEN
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2480)
        ENDIF
        INPUT K2
        READ (INPTUNIT,*) KTW0
        CHECK ACCEPTABILITY OF USER INPUT K2
        IF (KTW0.GT.0.0D+00) THEN
            PROCEED = .TRUE.
        ELSE
            ERRORCHR = 'KTW0'
            WRITE (SCRNUNIT,2660) ERRORCHR
        ENDIF
    END DO
*
*
*
    INPUT K3
*
*
*
    PROCEED = .FALSE.
    DO WHILE (.NOT.PROCEED)
*
*
*
        PROMPT IF NEEDED
        IF (TOPROMPT) THEN
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2500) UNITPRES
        ENDIF
        INPUT K3
        READ (INPTUNIT,*) KTHREE
        CHECK ACCEPTABILITY OF USER INPUT K3
        IF ((KONE*KTW0)+KTHREE.NE.0) THEN
            PROCEED = .TRUE.
        ELSE
            ERRORCHR = 'KTHREE'
            WRITE (SCRNUNIT,2660) ERRORCHR
        ENDIF
    END DO
*
*
*
    ELSE
        WRITE (SCRNUNIT,2640)
    ENDIF
END DO
*
*
*
RESET FLAGS USED FOR DIRECT INPUT OF KONE, KTW0, AND KTHREE
*
*
*
TOPROMPT = HOLDPRMT
INPTUNIT = HOLDINUN
*
*
*
ENTER THE HALF WIDTH STARTING VALUE
*
*
*
PROCEED = .FALSE.
DO WHILE (.NOT.PROCEED)
*
*
*
    PROMPT IF NEEDED
    IF (TOPROMPT) THEN
        WRITE (SCRNUNIT,2000)
        WRITE (SCRNUNIT,2510)
        WRITE (SCRNUNIT,2000)
        WRITE (SCRNUNIT,2520) UNITLENG
    
```

```

        ENDIF
*       INPUT VALUE
        READ (INPTUNIT,*) ASTART
*       CHECK ACCEPTABILITY OF INPUT VALUE
        IF (ASTART.GE.0.0D+00) THEN
            PROCEED = .TRUE.
            TOPPROMPT = HOLDPRMT
            INPTUNIT = HOLDINUN
        ELSE
            ERRORCHR = 'ASTART'
            WRITE (SCRNUNIT,2660) ERRORCHR
            TOPPROMPT = .TRUE.
            INPTUNIT = KEYBUNIT
        ENDIF
    END DO

*
*   ENTER THE HALF WIDTH INCREMENT SIZE
*
    PROCEED = .FALSE.
    DO WHILE (.NOT.PROCEED)
*       PROMPT IF NEEDED
        IF (TOPPROMPT) THEN
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2530) UNITLENG
        ENDIF
*       INPUT VALUE
        READ (INPTUNIT,*) ADELTA
*       CHECK ACCEPTABILITY OF INPUT VALUE
        IF (ADELTA.GT.0.0D+00) THEN
            PROCEED = .TRUE.
            TOPPROMPT = HOLDPRMT
            INPTUNIT = HOLDINUN
        ELSE
            ERRORCHR = 'ADELTA'
            WRITE (SCRNUNIT,2660) ERRORCHR
            TOPPROMPT = .TRUE.
            INPTUNIT = KEYBUNIT
        ENDIF
    END DO

*
*   ENTER THE NUMBER OF HALFWIDTH INCREMENTS TO STEP THROUGH
*
    PROCEED = .FALSE.
    DO WHILE (.NOT.PROCEED)
*       PROMPT IF NEEDED
        IF (TOPPROMPT) THEN
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2540)
        ENDIF
*       INPUT VALUE
        READ (INPTUNIT,*) NUMOFPTS
*       CHECK ACCEPTABILITY OF INPUT VALUE
        IF (NUMOFPTS.GT.0.AND. ( (0.4D+00*RADNIP) -
+       (ASTART + FLOAT(NUMOFPTS)*ADELTA) ).GE.0.0D+00) THEN
            PROCEED = .TRUE.
            TOPPROMPT = HOLDPRMT
            INPTUNIT = HOLDINUN
        ELSE
            ERRORCHR = 'NUMOFPTS'
            WRITE (SCRNUNIT,2660) ERRORCHR
            TOPPROMPT = .TRUE.
            INPTUNIT = KEYBUNIT
        ENDIF
    END DO

```

```

        ENDIF
    END DO
*
*   ENTER THE APPLIED NIP FORCE LOAD
*
    PROCEED = .FALSE.
    DO WHILE (.NOT.PROCEED)
*       PROMPT IF NEEDED
        IF (TOPROMPT) THEN
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2550)
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2560) UNITFORC
        ENDIF
*       INPUT VALUE
        READ (INPTUNIT,*) NIPFORCE
*       CHECK ACCEPTABILITY OF INPUT VALUE
        IF (NIPFORCE.GT.0.0D+00) THEN
            PROCEED = .TRUE.
            TOPROMPT = HOLDPRMT
            INPTUNIT = HOLDINUN
        ELSE
            ERRORCHR = 'NIPFORCE'
            WRITE (SCRNUNIT,2660) ERRORCHR
            TOPROMPT = .TRUE.
            INPTUNIT = KEYBUNIT
        ENDIF
    END DO
*
*   ENTER THE NIP FORCE STEP SIZE
*
    PROCEED = .FALSE.
    DO WHILE (.NOT.PROCEED)
*       PROMPT IF NEEDED
        IF (TOPROMPT) THEN
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2570) UNITFORC
        ENDIF
*       INPUT VALUE
        READ (INPTUNIT,*) DELTAFOR
*       CHECK ACCEPTABILITY OF INPUT VALUE
        IF (DELTAFOR.GE.FORCERES) THEN
            PROCEED = .TRUE.
            TOPROMPT = HOLDPRMT
            INPTUNIT = HOLDINUN
        ELSE
            ERRORCHR = 'DELTAFOR'
            WRITE (SCRNUNIT,2660) ERRORCHR
            TOPROMPT = .TRUE.
            INPTUNIT = KEYBUNIT
        ENDIF
    END DO
*
*   ENTER THE STARTING NIP FORCE
*
    PROCEED = .FALSE.
    DO WHILE (.NOT.PROCEED)
*       PROMPT IF NEEDED
        IF (TOPROMPT) THEN
            WRITE (SCRNUNIT,2000)
            WRITE (SCRNUNIT,2580) UNITFORC
        ENDIF

```

```

*      INPUT VALUE
      READ (INPTUNIT,*) STARTFOR
*      CHECK ACCEPTABILITY OF INPUT VALUE
      IF (STARTFOR.GE.0.0D+00.OR.STARTFOR.LE.NIPFORCE) THEN
          PROCEED = .TRUE.
          TOPPROMPT = HOLDPRMT
          INPTUNIT = HOLDINUN
      ELSE
          ERRORCHR = 'STARTFOR'
          WRITE (SCRNUNIT,2660) ERRORCHR
          TOPPROMPT = .TRUE.
          INPTUNIT = KEYBUNIT
      ENDIF
      END DO

*
*      ENTER DECISION IF WANT TO PRINT OUT ROTATIONS
*
      PROCEED = .FALSE.
      DO WHILE (.NOT.PROCEED)
*          PROMPT IF NEEDED
          IF (TOPPROMPT) THEN
              WRITE (SCRNUNIT,2000)
              WRITE (SCRNUNIT,2590)
              WRITE (SCRNUNIT,2640)
          ENDIF
*          INPUT VALUE
          READ (INPTUNIT,2925) THETAPRN
*          CHECK ACCEPTABILITY OF INPUT VALUE
          IF (THETAPRN.EQ.'Y'.OR.THETAPRN.EQ.'y'.OR.
+          THETAPRN.EQ.'N'.OR.THETAPRN.EQ.'n') THEN
              PROCEED = .TRUE.
              TOPPROMPT = HOLDPRMT
              INPTUNIT = HOLDINUN
          ELSE
              ERRORCHR = 'THETAPRN'
              WRITE (SCRNUNIT,2660) ERRORCHR
              TOPPROMPT = .TRUE.
              INPTUNIT = KEYBUNIT
          ENDIF
      END DO

*
*      ENTER DECISION IF WANT TO ITERATE FOR THE DEFORMATIONS IN FEM
*
      PROCEED = .FALSE.
      DO WHILE (.NOT.PROCEED)
*          PROMPT IF NEEDED
          IF (TOPPROMPT) THEN
              WRITE (SCRNUNIT,2000)
              WRITE (SCRNUNIT,2595)
              WRITE (SCRNUNIT,2640)
          ENDIF
*          INPUT VALUE
          READ (INPTUNIT,2925) REPEAT
*          CHECK ACCEPTABILITY OF INPUT VALUE
          IF (REPEAT.EQ.'Y'.OR.REPEAT.EQ.'y'.OR.
+          REPEAT.EQ.'N'.OR.REPEAT.EQ.'n') THEN
              PROCEED = .TRUE.
              TOPPROMPT = HOLDPRMT
              INPTUNIT = HOLDINUN
          ELSE
              ERRORCHR = 'REPEAT'
              WRITE (SCRNUNIT,2660) ERRORCHR

```

```

        TOPROMPT = .TRUE.
        INPTUNIT = KEYBUNIT
    ENDIF
END DO
*
* ENTER IF LEFT NIP END IS CONSTRAINED FROM ROTATION
*
PROCEED = .FALSE.
DO WHILE (.NOT.PROCEED)
*   PROMPT IF NEEDED
    IF (TOPROMPT) THEN
        WRITE (SCRNUNIT,2000)
        WRITE (SCRNUNIT,2600)
        WRITE (SCRNUNIT,2640)
    ENDIF
*   INPUT VALUE
    READ (INPTUNIT,2925) CONSTLFT
*   CHECK ACCEPTABILITY OF INPUT VALUE
    IF (CONSTLFT.EQ.'Y'.OR.CONSTLFT.EQ.'y'.OR.
+     CONSTLFT.EQ.'N'.OR.CONSTLFT.EQ.'n') THEN
        PROCEED = .TRUE.
        TOPROMPT = HOLDPRMT
        INPTUNIT = HOLDINUN
    ELSE
        ERRORCHR = 'CONSTLFT'
        WRITE (SCRNUNIT,2660) ERRORCHR
        TOPROMPT = .TRUE.
        INPTUNIT = KEYBUNIT
    ENDIF
END DO
*
* ENTER IF RIGHT NIP END IS CONSTRAINED FROM ROTATION
*
PROCEED = .FALSE.
DO WHILE (.NOT.PROCEED)
*   PROMPT IF NEEDED
    IF (TOPROMPT) THEN
        WRITE (SCRNUNIT,2000)
        WRITE (SCRNUNIT,2610)
        WRITE (SCRNUNIT,2640)
    ENDIF
*   INPUT VALUE
    READ (INPTUNIT,2925) CONSTRGH
*   CHECK ACCEPTABILITY OF INPUT VALUE
    IF (CONSTRGH.EQ.'Y'.OR.CONSTRGH.EQ.'y'.OR.
+     CONSTRGH.EQ.'N'.OR.CONSTRGH.EQ.'n') THEN
        PROCEED = .TRUE.
        TOPROMPT = HOLDPRMT
        INPTUNIT = HOLDINUN
    ELSE
        ERRORCHR = 'CONSTRGH'
        WRITE (SCRNUNIT,2660) ERRORCHR
        TOPROMPT = .TRUE.
        INPTUNIT = KEYBUNIT
    ENDIF
END DO
*
* ENTER THE NUMBER OF FINITE ELEMENTS ACROSS THE WOUND ROLL
*   (WINKLER FOUNDATION ROLL)
*
PROCEED = .FALSE.
DO WHILE (.NOT.PROCEED)

```

```

*          PROMPT IF NEEDED
          IF (TOPROMPT) THEN
              WRITE (SCRNUNIT,2000)
              WRITE (SCRNUNIT,2620)
          ENDIF
*          INPUT VALUE
          READ (INPTUNIT,*) NMWINKEL
*          CHECK ACCEPTABILITY OF INPUT VALUE
*
*          CALL SUBROUTINE ELEMNMBR TO DETERMINE THE NUMBER OF ELEMENTS
*          TO THE LEFT AND RIGHT OF THE WOUND ROLL
*
          CALL ELEMNMBR(UNITTYPE, LEFTLEM, LEFTWINK, MTRPERIN,
+          RIGHELEM, NIPSTUBW, WINKWIDT, XTRAELEM, LEFTSTUB,
+          NIPROLLW, NMWINKEL)
          IF (NMWINKEL.GT.0.AND.NMWINKEL.LE.
+          (MAXELEM-LEFTLEM-RIGHELEM-XTRAELEM)) THEN
              PROCEED = .TRUE.
              TOPROMPT = HOLDPRMT
              INPTUNIT = HOLDINUN
          ELSE
              ERRORCHR = 'NMWINKEL'
              WRITE (SCRNUNIT,2660) ERRORCHR
              TOPROMPT = .TRUE.
              INPTUNIT = KEYBUNIT
          ENDIF
          END DO
*
*          DETERMINE THE NUMBER OF NODES ACROSS THE WIDTH
*
          NMWINKEL = NMWINKEL + XTRAELEM
          NMWINKND = NMWINKEL + 1
          NMCROSEL = NMWINKEL + LEFTLEM + RIGHELEM
          NMCROSND = NMCROSEL + 1
*
*          ENTER IF THE FINITE ELEMENTS ARE <<<NOT>>> EVENLY SPACED ACROSS THE
*          WIDTH BUT ONLY IF PROGRAM IS READING DATA FROM FILE
*
          IF (DECISON1.EQ.'') THEN
              READ (INPTUNIT,2925) DECISON2
          ELSE
              IF (TOPROMPT) THEN
                  WRITE (SCRNUNIT,2700)
                  DECISON2 = 'N'
              ENDIF
*
*          CALL SUBROUTINE MESHER TO BUILD THE FEM MESH
*
          CALL MESHER(LOCATE, DECISON2, INPTUNIT, NMCROSND, ELLENGTH,
+          NIPROLLW, NMCROSEL, CODEBEAM, CODEWINK, LEFTLEM, LEFTWINK,
+          WINKWIDT, MTRPERIN, UNITTYPE, NMWINKEL, NIPSTUBW, LEFTSTUB)
*
          CLOSE INPUT FILE
*
          CLOSE (INPTUNIT)
*****
*          FORMAT STATEMENTS
*
2000  FORMAT ( 1X, ' ')
2010  FORMAT ( 1X, 76('*'))
2020  FORMAT (2X, 'This program is written by Paul Hoffercker for the MAE

```

```

+ Research Lab')
2030  FORMAT (2X, 'at Oklahoma State University under the supervision of
+ Dr. J.K. Good.')
2040  FORMAT (2X, 'The purpose of the program is to explore the effect o
+f a Nip roller')
2050  FORMAT (2X, 'impinging on a wound roll.')
2060  FORMAT (2X, 'The program uses Hertzian contact algorithms to compu
+te generic wound')
2070  FORMAT (2X, 'roll deformations as a function of contact width and
+and the roll ')
2080  FORMAT (2X, 'modulus of elasticity. From this the wound roll stif
+fness versus ')
2090  FORMAT (2X, 'radial deformation is determined. The stiffness inse
+rts into a FEM')
2100  FORMAT (2X, 'of the wound roll represented by a Winkler foundation
+. The program')
2110  FORMAT (2X, 'outputs the wound roll expected deformations at the s
+pecified nip')
2120  FORMAT (2X, 'load. The roll modulus of elasticity, Eroll, derives
+ from the model')
2130  FORMAT (2X, 'used for pressure versus strain. The pressure model
+uses three')
2140  FORMAT (2x, 'constants, K1, K2, and K3 as shown:')
2150  FORMAT (7X, 'Pressure = K1 - K1 * exp (K2 * strain) + K3 * strain'
+)
2160  FORMAT (2X, 'This program allows the user to input K1, K2, and K3
+directly.')
2170  FORMAT (2X, 'If instead a data file containing the number of data
+points (1st')
2180  FORMAT (2X, 'entry) and then strain/pressure data pairs exists, K1
+ and K2 can')
2190  FORMAT (2X, 'be determined from the file.')
2200  FORMAT (2X, 'Please refer to the users manual for additional infor
+mation.')
2250  FORMAT (2X, 'Enter the input filename (with path) or "none" to be
+prompted for each input.')
2260  FORMAT (2X, 'File ',A80,' does not exist or has an incorrect path.
+')
2270  FORMAT (10X, 'Enter the date.')
2280  FORMAT (10X, 'Enter the desired unit system (MKS or ENG).')
2290  FORMAT ( 5X, 'NIP ROLLER DIMENSIONS...')
2300  FORMAT (10X, 'Enter the radius of the nip roll. ',A10)
2310  FORMAT (10X, 'Enter the CMD distance between the applied load lo'
+ 'cations. ',A10)
2320  FORMAT (10X, 'Enter distance from machine zero to nip roller le',
+ 'ft edge',/,10X,'(left stub shaft length). ',A10)
2330  FORMAT (10X, 'Enter the nip roller width. ',A10)
2340  FORMAT ( 5X, 'WOUND ROLL DIMENSIONS...')
2350  FORMAT (10X, 'Enter the radius of the wound roll. ',A10)
2360  FORMAT (10X, 'Enter the distance from machine zero to the wound ',
+ 'roll. ',A10)
2370  FORMAT (10X, 'Enter the wound roll width. ',A10)
2380  FORMAT ( 5X, 'NIP ROLL MATERIAL DATA...')
2400  FORMAT (10X, 'Enter the nip roller modulus of elasticity. ',A10)
2410  FORMAT (10X, 'Enter the nip roller moment of inertia. ',A10)
2420  FORMAT (10X, 'Enter the Poissons ratio of the nip roll.')
2430  FORMAT (10X, 'Enter the nip stub shaft modulus of elasticity. '
+ ',A10)
2440  FORMAT (10X, 'Enter the nip stub shaft moment of inertia. ',A10)
2450  FORMAT ( 5X, 'WOUND ROLL MATERIAL DATA...')
2460  FORMAT (10X, 'Enter the Poissons ratio of the wound roll.')
2470  FORMAT (10X, 'Enter K1. ',A10)

```

```

2480 FORMAT (10X, 'Enter K2. ')
2500 FORMAT (10X, 'Enter K3. ',A10)
2510 FORMAT ( 5X, 'GENERAL HERTZIAN CONTACT ANALYSIS...')
2520 FORMAT (10X, 'Enter the half width of contact starting value. ',
+   A10)
2530 FORMAT (10X, 'Enter the half width increment step size. ',A10)
2540 FORMAT (10X, 'Enter the number of halfwidth increments to step t',
+'hrough. ')
2550 FORMAT ( 5X, 'FINITE ELEMENT ANALYSIS...')
2560 FORMAT (10X, 'Enter the applied nip force. ',A10)
2570 FORMAT (10X, 'Enter the nip force step size. ',A10)
2580 FORMAT (10X, 'Enter the starting nip force. ',A10)
2590 FORMAT (10X, 'Should the program output rotation information?')
2595 FORMAT (10X, 'Should the FEM iterate to determine deformations?')
2600 FORMAT (10X, 'Is the nip left end constrained against rotation?')
2610 FORMAT (10X, 'Is the nip right end constrained against rotation?')
2620 FORMAT (10X, 'Enter number of finite elements desired across wou',
+'nd roll. ')
*
XXX
*2630 FORMAT (10X, 'Should the program determine K1 and K2 correspondi',
*   +'ng to material data in a file?')
*
XXX
2640 FORMAT (10X, '(Y or N) ')
2650 FORMAT (10X, 'Enter the pressure/ strain data filename (including
+'path). ')
2660 FORMAT (10X, 'Input value ',A10,' is out of acceptable range. ')
2670 FORMAT ( T7,'K1= ',F10.3,1X,A10,T30,'K2= ',F10.3)
2680 FORMAT ( T7,'K3= ',F10.3,1X,A10)
2690 FORMAT (10X, 'K1, K2, and K3 must be input directly. ')
*2700 FORMAT (10X, 'Enter Y if the finite elements are NOT evenly spaced
*   +. ')
2900 FORMAT ( A80 )
2910 FORMAT ( A20 )
2911 FORMAT ( A3 )
2920 FORMAT ( 1X, '<<Press Return>>' )
2925 FORMAT ( A1 )
*****
*
*
*   RETURN TO MAIN PROGRAM
*
*   RETURN
*   END
*****
*****
*
*   ELEMNMBR
*   DETERMINES THE NUMBER OF ELEMENTS TO THE LEFT AND RIGHT OF THE
*   WOUND ROLL
*
*****
*
*   SUBROUTINE ELEMNMBR(UNITTYPE, LEFTELEM, LEFTWINK, MTRPERIN,
+   RIGHELEM, NIPSTUBW, WINKWIDT, XTRAELEM, LEFTSTUB, NIPROLLW,
+   NMWINKEL)
*
*   DEFINE VARIABLES
*
*   INTEGER RIGHELEM, LEFTELEM, XTRAELEM, NMWINKEL
*   DOUBLE PRECISION NIPSTUBW, LEFTWINK, MTRPERIN, WINKWIDT
*   DOUBLE PRECISION LEFTSTUB, NIPROLLW
*   CHARACTER UNITTYPE*3

```



```

*
*   COUNT THE NIP ROLLER ELEMENTS TO THE LEFT AND RIGHT OF WOUND ROLL
*
IF (UNITTTYPE.EQ.'ENG'.OR.UNITTTYPE.EQ.'eng') THEN
  LEFTLEM = INT(LEFTWINK)
  RIGHELEM = INT(NIPSTUBW-LEFTWINK-WINKWIDT)
  IF ((LEFTWINK-(FLOAT(LEFTLEM))).GE.0.05D+00) THEN
    LEFTLEM = LEFTLEM + 1
  ENDIF
  IF ((NIPSTUBW-LEFTWINK-WINKWIDT-(FLOAT(RIGHELEM)))
+   .GE.0.05D+00) THEN
    RIGHELEM = RIGHELEM + 1
  ENDIF
ELSE
  LEFTLEM = INT(LEFTWINK/MTRPERIN)
  RIGHELEM = INT((NIPSTUBW-LEFTWINK-WINKWIDT)/MTRPERIN)
  IF ((LEFTWINK/MTRPERIN-LEFTLEM).GE.0.05D+00) THEN
    LEFTLEM = LEFTLEM + 1
  ENDIF
  IF (((NIPSTUBW-LEFTWINK-WINKWIDT)/MTRPERIN-RIGHELEM).GE.
+   0.05D+00) THEN
    RIGHELEM = RIGHELEM + 1
  ENDIF
ENDIF
ENDIF

*
*   IF THE EDGES OF THE NIP ROLLER DO NOT CORRESPOND TO 1 INCH ELEMENTS
*   THEN EXTRA ELEMENTS WILL BE REQUIRED TO ACCOMODATE
*
IF (LEFTSTUB.LT.LEFTWINK.AND.MOD(LEFTSTUB,1.00D+00).NE.0.0D+00)
+  LEFTLEM = LEFTLEM + 1
IF (LEFTSTUB.GT.LEFTWINK.AND.MOD((LEFTSTUB-LEFTWINK),
+  (WINKWIDT/FLOAT(NMWINKEL))).NE.0.0D+00) XTRAELEM = XTRAELEM +1
IF ((LEFTSTUB+NIPROLLW).LT.(LEFTWINK+WINKWIDT).AND.MOD(
+  (LEFTSTUB+NIPROLLW-LEFTWINK), (WINKWIDT/FLOAT(NMWINKEL)))
+  .NE.0.0D+00) XTRAELEM = XTRAELEM +1
IF ((LEFTSTUB+NIPROLLW).GT.(LEFTWINK+WINKWIDT).AND.MOD((NIPSTUBW-
+  LEFTSTUB-NIPROLLW),1.0D+00).NE.0.0D+00) RIGHELEM = RIGHELEM +1

*
*   RETURN BACK TO INVALUES ROUTINE
*
RETURN
END
*****
*****
*
*   MESHER
*   ESTABLISHES THE FEM MESH ACROSS THE WIDTH OF THE ROLLS
*
*****
*
SUBROUTINE MESHER(LOCATE, DECISON2, INPTUNIT, NMCROSND, ELLENGTH,
+  NIPROLLW, NMCROSEL, CODEBEAM, CODEWINK, LEFTLEM, LEFTWINK,
+  WINKWIDT, MTRPERIN, UNITTTYPE, NMWINKEL, NIPSTUBW, LEFTSTUB)
*
*   DEFINE VARIABLES
*
INTEGER LAST, CURRNODE, CURRELEM, INPTUNIT, NMCROSND, NMCROSEL
INTEGER CODEBEAM(NMCROSEL), CODEWINK(NMCROSEL), LEFTLEM
INTEGER NMWINKEL
DOUBLE PRECISION LEFTWINK, MTRPERIN, WINKWIDT, LEFTSTUB
DOUBLE PRECISION LOCATE(NMCROSND), ELLENGTH(NMCROSEL), NIPSTUBW

```



```

+           1.00D+00) THEN
+           DISTANCE = (LEFTWINK-LOCATE(CURRNODE-1))/
+           MTRPERIN
+           ELSE
+           DISTANCE = 1.00D+00*MTRPERIN
+           ENDIF
+           ENDIF
*
* WOUND ROLL ELEMENTS
*
+ ELSEIF (CURRNODE.GT.LEFTELEM+1.AND.CURRNODE.LE.
+ LEFTELEM+1+NMWINKEL) THEN
+ DISTANCE = LENGTH
*
* ELEMENTS RIGHT OF WOUND ROLL
*
+ ELSE
+ IF (UNITYTYPE.EQ.'ENG'.OR.UNITYTYPE.EQ.'eng') THEN
+ IF (MOD((NIPSTUBW-LOCATE(CURRNODE-1)),1.00D+00)
+ .GT.0.00D+00) THEN
+ DISTANCE = MOD((NIPSTUBW-LOCATE(CURRNODE-1)),
+ 1.00D+00)
+ ELSE
+ DISTANCE = 1.00D+00
+ ENDIF
+ ELSE
+ IF (MOD((NIPSTUBW-LOCATE(CURRNODE-1))/MTRPERIN,
+ 1.00D+00).GT.0.00D+00) THEN
+ DISTANCE = MOD((NIPSTUBW-LOCATE(CURRNODE-1))/
+ MTRPERIN,1.00D+00)
+ ELSE
+ DISTANCE = 1.00D+00*MTRPERIN
+ ENDIF
+ ENDIF
+ ENDIF
*
* INSERT NIP ROLLER'S END POINTS UNLESS THEY CORRESPOND TO AN
* ALREADY EXISTING LOCATION
*
+ IF (HOLDER.NE.0.00D+00) THEN
+ DISTANCE = HOLDER
+ HOLDER = 0.00D+00
+ ENDIF
+ IF ((LEFTSTUB-LOCATE(CURRNODE-1)).LT.DISTANCE.AND.
+ (LEFTSTUB-LOCATE(CURRNODE-1)).GT.0.00D+00) THEN
+ HOLDER = DISTANCE
+ DISTANCE = LEFTSTUB-LOCATE(CURRNODE-1)
+ HOLDER = HOLDER - DISTANCE
+ ENDIF
+ IF ((LEFTSTUB+NIPROLLW-LOCATE(CURRNODE-1)).LT.DISTANCE
+ .AND.(LEFTSTUB+NIPROLLW-LOCATE(CURRNODE-1)).GT.
+ 0.00D+00) THEN
+ HOLDER = DISTANCE
+ DISTANCE = LEFTSTUB + NIPROLLW - LOCATE(CURRNODE-1)
+ HOLDER = HOLDER - DISTANCE
+ ENDIF
*
* ADD THE CURRENT ELEMENT'S LENGTH (DISTANCE) TO THE PREVIOUS
* LOCATION AND THEN REPEAT
*
+ LOCATE(CURRNODE) = LOCATE(CURRNODE-1) + DISTANCE
235 CONTINUE

```

```

*
*   ENDIF
*
*   CALCULATE THE ELEMENT LENGTHS
*
DO 230 CURRELEM = 2, NMCROSND
    ELLENGTH(CURRELEM-1) = LOCATE(CURRELEM) - LOCATE(CURRELEM-1)
230 CONTINUE
*
*   DETERMINE CODE OF EACH ELEMENT
*
*   CODEBEAM() = 0 ...THE NIP IS ONLY THE STUB SHAFT
*   CODEBEAM() = 1 ...THE NIP IS THE FULL ROLLER
*   CODEWINK() = 0 ...THE WOUND ROLL DOESN'T EXIST HERE
*   CODEWINK() = 1 ...THE WOUND ROLL EXISTS
*
DO 245 CURRELEM = 1, NMCROSEL
    IF (CURRELEM.GT.LEFTTELEM.AND.CURRELEM.LE.LEFTTELEM+NMWINKEL)
+       THEN
        CODEWINK(CURRELEM) = 1
    ELSE
        CODEWINK(CURRELEM) = 0
    ENDIF
    IF (LOCATE(CURRELEM+1).GT.LEFTSTUB.AND.
+       LOCATE(CURRELEM+1).LE.LEFTSTUB+NIPROLLW) THEN
        CODEBEAM(CURRELEM) = 1
    ELSE
        CODEBEAM(CURRELEM) = 0
    ENDIF
245 CONTINUE
*
*   RETURN BACK TO INVALUES ROUTINE
*
*   RETURN
*   END
*****
*****
*
*   OUTDATA, WRITES ALL DATA TO FILES
*
*****
*****
*
SUBROUTINE OUTDATA(MARKER, DATE, KONE, KTWO, RADROLL, POISROLL,
+   RADNIP, POISSNIP, NIPROLLE, AVERLPW, NIPROLLI, NMCROSEL,
+   NIPFORCE, NIPROLLW, NODEFORC, NUMGAPS, PERCENT, NUMOFPTS,
+   HAFWIDTH, DEFORM, STIFNESS, LOADPERW, CONDITON, STATUS,
+   NMCROSND, NODEDISP, DOFFERND, TIME1, DELTAFOR, WINKDISP,
+   LPWVSNO, MAXPRESS, UNITTYPE, UNITLENG, UNITFORC, UNITFPL,
+   UNITPRES, UNITMOI, UNITANGL, RADEFF, TEMPDISP, HALFCONW,
+   LOCATE, CODEBEAM, NIPSTUBW, LEFTSTUB, LEFTWINK, NIPSTUBE,
+   NIPSTUBI, WINKWIDT, NMWINKEL, CODEWINK, LEFTTELEM, KTHREE,
+   THETAPRN, CONSTLFT, CONSTRGH, STARTFOR, REPEAT)
*
*   DEFINE VARIABLES
*
INTEGER MARKER, NMCROSEL, NUMGAPS, PERCENT, NUMOFPTS, NMCROSND
INTEGER DOFFERND, OUTUNIT1, OUTUNIT2, OUTUNIT3, OUTUNIT4, OUTUNIT5
INTEGER INDEXNUM, COLUMN, ROW, NUMACROS, TIME1, LOOP, NMWINKEL
INTEGER ELEONES, ELETENS, CODEBEAM(NMCROSEL), CODEWINK(NMCROSEL)
INTEGER CURRELEM, LEFTTELEM
DOUBLE PRECISION KONE, KTWO, KTHREE, RADROLL, POISROLL, RADNIP
DOUBLE PRECISION NIPROLLE, POISSNIP, NIPROLLI, NIPFORCE, NIPROLLW
DOUBLE PRECISION NODEFORC, DELTAFOR, RADEFF, LOCATE(NMCROSND)

```

```

DOUBLE PRECISION NIPSTUBW, LEFTSTUB, LEFTWINK, ELEMDISP, STARTFOR
DOUBLE PRECISION NIPSTUBE, NIPSTUBI, WINKWIDT, AVERLPW
DOUBLE PRECISION HAFWIDTH(NUMOFPTS), DEFORM(NUMOFPTS)
DOUBLE PRECISION STIFNESS(NUMOFPTS), LOADPERW(NUMOFPTS)
DOUBLE PRECISION WINKDISP(NMCROSEL), LPWVSOD(NMCROSND)
DOUBLE PRECISION MAXPRESS(NMCROSND), HALFCONW(NMCROSND)
DOUBLE PRECISION TEMPDISP(DOFPERND*NMCROSND+1,6)
DOUBLE PRECISION NODEDISP(2*DOFPERND*NMCROSND)
PARAMETER (OUTUNIT1=11, OUTUNIT2=12, OUTUNIT3=13, OUTUNIT4=14)
PARAMETER (OUTUNIT5=15)
CHARACTER OUTFILE1*12, OUTFILE2*12, OUTFILE3*12, OUTFILE4*12
CHARACTER OUTFILE5*12, CONSTLFT*1, CONSTRGH*1
CHARACTER*6 TYPE[ALLOCATABLE](:)
CHARACTER DATE*20, CONDITON(NMCROSEL)*7, STATUS(NMCROSEL)*2
CHARACTER UNITYTYPE*3, UNITLENG*10, UNITFORC*10, UNITFPL*10
CHARACTER UNITPRES*10, UNITMOI*10, UNITANGL*10, THETAPRN*1
CHARACTER REPEAT*1
*
*   ALLOCATE VARIABLE DIMENSION ARRAY
*
*   ALLOCATE (TYPE(NMCROSEL))
*
*   DEFINE NAMES OF OUTPUT FILES
*
*   OUTFILE1 = 'PINGELPW.OUT'
*   OUTFILE2 = 'PINGESUM.OUT'
*   OUTFILE3 = 'PINGEINC.OUT'
*   OUTFILE4 = 'PINGEDEF.OUT'
*   OUTFILE5 = 'PINGEPRO.OUT'
*
*   SET VARIABLES
*
*   NUMACROS = 6
*
*   OPEN AND PREPARE OUTPUT FILES
*
*   IF (MARKER.EQ.2) THEN
*       OPEN (UNIT = OUTUNIT1, FILE = OUTFILE1, STATUS = 'UNKNOWN')
*       OPEN (UNIT = OUTUNIT2, FILE = OUTFILE2, STATUS = 'UNKNOWN')
*       OPEN (UNIT = OUTUNIT3, FILE = OUTFILE3, STATUS = 'UNKNOWN')
*       OPEN (UNIT = OUTUNIT4, FILE = OUTFILE4, STATUS = 'UNKNOWN')
*       OPEN (UNIT = OUTUNIT5, FILE = OUTFILE5, STATUS = 'UNKNOWN')
*
*   PRINT OUT HEADERS
*
*   PINGELPW.OUT
*
*       WRITE (OUTUNIT1,3000)
*       WRITE (OUTUNIT1,3010) OUTFILE1
*       WRITE (OUTUNIT1,3020)
*       WRITE (OUTUNIT1,3000)
*       WRITE (OUTUNIT1,3030)
*       WRITE (OUTUNIT1,3040)
*       WRITE (OUTUNIT1,3050)
*       WRITE (OUTUNIT1,3120) DATE
*       WRITE (OUTUNIT1,3000)
*       WRITE (OUTUNIT1,3200) UNITYTYPE
*       WRITE (OUTUNIT1,3000)
*       WRITE (OUTUNIT1,3205)
*       WRITE (OUTUNIT1,3210) KONE, UNITPRES, KTWO
*       WRITE (OUTUNIT1,3215) KTHREE, UNITPRES
*       WRITE (OUTUNIT1,3220) RADROLL, UNITLENG, POISROLL

```

```

WRITE (OUTUNIT1,3000)
WRITE (OUTUNIT1,3207)
WRITE (OUTUNIT1,3220) RADNIP, UNITLENG, POISSNIP
WRITE (OUTUNIT1,3250) NIPROLLE, UNITPRES
WRITE (OUTUNIT1,3000)
WRITE (OUTUNIT1,3208)
WRITE (OUTUNIT1,3230) RADEFF, UNITLENG
WRITE (OUTUNIT1,3000)
WRITE (OUTUNIT1,3000)
WRITE (OUTUNIT1,3310)
WRITE (OUTUNIT1,3320)
WRITE (OUTUNIT1,3330) UNITLENG, UNITLENG, UNITFPL, UNITPRES
WRITE (OUTUNIT1,3350)
WRITE (OUTUNIT1,3000)

```

*
*
*

PINGESUM.OUT

```

WRITE (OUTUNIT2,3000)
WRITE (OUTUNIT2,3010) OUTFILE2
WRITE (OUTUNIT2,3020)
WRITE (OUTUNIT2,3000)
WRITE (OUTUNIT2,3060)
WRITE (OUTUNIT2,3070)
WRITE (OUTUNIT2,3120) DATE
WRITE (OUTUNIT2,3000)
WRITE (OUTUNIT2,3200) UNITTYPE
WRITE (OUTUNIT2,3000)
WRITE (OUTUNIT2,3205)
WRITE (OUTUNIT2,3210) KONE, UNITPRES, KTWO
WRITE (OUTUNIT2,3215) KTHREE, UNITPRES
WRITE (OUTUNIT2,3220) RADROLL, UNITLENG, POISROLL
WRITE (OUTUNIT2,3360) WINKWIDT, UNITLENG, NMWINKEL
WRITE (OUTUNIT2,3000)
WRITE (OUTUNIT2,3206)
WRITE (OUTUNIT2,3300) NIPSTUBE, UNITPRES, NIPSTUBI, UNITMOI
WRITE (OUTUNIT2,3000)
WRITE (OUTUNIT2,3207)
WRITE (OUTUNIT2,3220) RADNIP, UNITLENG, POISSNIP
WRITE (OUTUNIT2,3300) NIPROLLE, UNITPRES, NIPROLLI, UNITMOI
WRITE (OUTUNIT2,3362) NIPROLLW, UNITLENG
WRITE (OUTUNIT2,3000)
WRITE (OUTUNIT2,3208)
WRITE (OUTUNIT2,3370) NIPFORCE, UNITFORC, DELTAFOR, UNITFORC
WRITE (OUTUNIT2,3372) STARTFOR, UNITFORC, REPEAT
WRITE (OUTUNIT2,3360) NIPSTUBW, UNITLENG, NMCROSEL
WRITE (OUTUNIT2,3365) CONSTLFT, CONSTRGH
WRITE (OUTUNIT2,3230) RADEFF, UNITLENG
WRITE (OUTUNIT2,3000)
WRITE (OUTUNIT2,3209)
WRITE (OUTUNIT2,3373) LEFTSTUB, UNITLENG, LEFTWINK, UNITLENG
WRITE (OUTUNIT2,3000)
WRITE (OUTUNIT2,3000)
WRITE (OUTUNIT2,3380)
WRITE (OUTUNIT2,3390)
WRITE (OUTUNIT2,3395)
WRITE (OUTUNIT2,3400) UNITLENG, UNITLENG
WRITE (OUTUNIT2,3350)

```

*
*
*

PINGEINC.OUT

```

WRITE (OUTUNIT3,3000)
WRITE (OUTUNIT3,3010) OUTFILE3

```

```

WRITE (OUTUNIT3,3020)
WRITE (OUTUNIT3,3000)
WRITE (OUTUNIT3,3080)
WRITE (OUTUNIT3,3090)
WRITE (OUTUNIT3,3120) DATE
WRITE (OUTUNIT3,3000)
WRITE (OUTUNIT3,3200) UNITYTYPE
WRITE (OUTUNIT3,3000)
WRITE (OUTUNIT3,3410)
DO 300 COLUMN = 1, NMCROSEL
    WRITE (OUTUNIT3,3420)
300 CONTINUE
WRITE (OUTUNIT3,3410)
DO 310 COLUMN = 1, NMCROSEL
    WRITE (OUTUNIT3,3430)
310 CONTINUE
ELETENS = INT(NMCROSEL/10)
WRITE (OUTUNIT3,3440)
DO 320 COLUMN = 1, ELETENS, 1
    WRITE (OUTUNIT3,3450) COLUMN
320 CONTINUE
ELEONES = 9
WRITE (OUTUNIT3,3460) UNITFORC
DO 330 COLUMN = 0, ELETENS, 1
    IF (COLUMN.EQ.ELETENS) ELEONES= MOD(NMCROSEL,10)
    WRITE (OUTUNIT3,3470) (LOOP, LOOP = 1, ELEONES)
330 CONTINUE
DO 335 COLUMN = 1, NMCROSEL+7
    WRITE (OUTUNIT3,3355)
335 CONTINUE
WRITE (OUTUNIT3,3412)
*
* PINGEDEF.OUT
*
WRITE (OUTUNIT4,3000)
WRITE (OUTUNIT4,3010) OUTFILE4
WRITE (OUTUNIT4,3020)
WRITE (OUTUNIT4,3000)
WRITE (OUTUNIT4,3100)
WRITE (OUTUNIT4,3110)
WRITE (OUTUNIT4,3120) DATE
WRITE (OUTUNIT4,3000)
WRITE (OUTUNIT4,3200) UNITYTYPE
WRITE (OUTUNIT4,3000)
WRITE (OUTUNIT4,3205)
WRITE (OUTUNIT4,3360) WINKWIDT, UNITLENG, NMWINKEL
WRITE (OUTUNIT4,3000)
WRITE (OUTUNIT4,3206)
WRITE (OUTUNIT4,3300) NIPSTUBE, UNITPRES, NIPSTUBI, UNITMOI
WRITE (OUTUNIT4,3000)
WRITE (OUTUNIT4,3207)
WRITE (OUTUNIT4,3300) NIPROLLE, UNITPRES, NIPROLLI, UNITMOI
WRITE (OUTUNIT4,3362) NIPROLLW, UNITLENG
WRITE (OUTUNIT4,3000)
WRITE (OUTUNIT4,3208)
WRITE (OUTUNIT4,3370) NIPFORCE, UNITFORC, DELTAFOR, UNITFORC
WRITE (OUTUNIT4,3372) STARTFOR, UNITFORC, REPEAT
WRITE (OUTUNIT4,3360) NIPSTUBW, UNITLENG, NMCROSEL
*
* PINGEPRO.OUT
*
WRITE (OUTUNIT5,3000)

```

```

WRITE (OUTUNIT5,3010) OUTFILE5
WRITE (OUTUNIT5,3020)
WRITE (OUTUNIT5,3000)
WRITE (OUTUNIT5,3130)
WRITE (OUTUNIT5,3140)
WRITE (OUTUNIT5,3150)
WRITE (OUTUNIT5,3120) DATE
WRITE (OUTUNIT5,3000)
WRITE (OUTUNIT5,3200) UNITYTYPE
WRITE (OUTUNIT5,3000)
WRITE (OUTUNIT5,3205)
WRITE (OUTUNIT5,3210) KONE, UNITPRES, KTWO
WRITE (OUTUNIT5,3215) KTHREE, UNITPRES
WRITE (OUTUNIT5,3360) WINKWIDT, UNITLENG, NMWINKEL
WRITE (OUTUNIT5,3000)
WRITE (OUTUNIT5,3207)
WRITE (OUTUNIT5,3300) NIPROLLE, UNITPRES, NIPROLLI, UNITMOI
WRITE (OUTUNIT5,3362) NIPROLLW, UNITLENG
WRITE (OUTUNIT5,3000)
WRITE (OUTUNIT5,3000)
WRITE (OUTUNIT5,3208)
WRITE (OUTUNIT5,3370) NIPFORCE, UNITFORC, DELTAFOR, UNITFORC
WRITE (OUTUNIT5,3372) STARTFOR, UNITFORC, REPEAT
WRITE (OUTUNIT5,3360) NIPSTUBW, UNITLENG, NMCROSEL
WRITE (OUTUNIT5,3230) RADEFF, UNITLENG
WRITE (OUTUNIT5,3000)
WRITE (OUTUNIT5,3000)
WRITE (OUTUNIT5,3580)
WRITE (OUTUNIT5,3590)
WRITE (OUTUNIT5,3600) UNITLENG, UNITFPL, UNITPRES, UNITFPL,
+   UNITLENG
WRITE (OUTUNIT5,3350)
ENDIF
*
*   PRINT OUT LOAD PER WIDTH RESULTS
*
IF (MARKER.EQ.3) THEN
DO 340 ROW = 1, NUMOFPTS
WRITE (OUTUNIT1,3510) HAFWIDTH(ROW), DEFORM(ROW), LOADPERW
+   (ROW), STIFNESS(ROW)
340 CONTINUE
ENDIF
*
*   PRINT OUT INTERMEDIATE GAPPING AND DEFORMATION RESULTS
*
IF (MARKER.EQ.4) THEN
*
*   PINGEINC.OUT
*
WRITE (OUTUNIT3,3530) NODEFORC, (STATUS(COLUMN), COLUMN = 1,
+   NMCROSEL)
*
*   PINGEDEF.OUT
*
IF (TIME1.GT.NUMACROS) THEN
TIME1 = 1
WRITE (OUTUNIT4,3000)
WRITE (OUTUNIT4,3000)
WRITE (OUTUNIT4,3520)
WRITE (OUTUNIT4,3500) UNITFORC, (TEMPDISP (1,COLUMN),
+   COLUMN = 1, NUMACROS)
WRITE (OUTUNIT4,3350)

```



```

DO 350 ROW = 1, NMCROSND
  INDEXNUM = (ROW * DOFFPERND)
  WRITE (OUTUNIT4,3540) ROW, UNITLENG, (TEMPDISP
+   (INDEXNUM,COLUMN), COLUMN = 1, NUMACROS)
  IF (THETAPRN.EQ.'Y'.OR.THETAPRN.EQ.'y') THEN
+   WRITE (OUTUNIT4,3550) ROW, UNITANGL, (TEMPDISP
    (INDEXNUM+1,COLUMN), COLUMN = 1, NUMACROS)
  ENDIF
*   RESET THE ARRAY VALUES TO 0
  DO 360 COLUMN = 1, NUMACROS
    TEMPDISP(1,COLUMN) = 0.0D+00
    TEMPDISP(INDEXNUM,COLUMN) = 0.0D+00
    TEMPDISP(INDEXNUM+1,COLUMN) = 0.0D+00
360   CONTINUE
350   CONTINUE
  ENDIF
ENDIF
*
*   PRINT OUT PROFILE RESULTS IN PINGE INCrement, SUMMmary and PROfile
*
IF (MARKER.EQ.5) THEN
  WRITE (OUTUNIT3,3000)
  WRITE (OUTUNIT3,3560) NODEFORC, UNITFORC, NUMGAPS
  WRITE (OUTUNIT3,3570) PERCENT
*
*   SET ELEMENT 'TYPES' CORRESPONDING TO THEIR CODE
*
DO 369 CURRELEM = 1, NMCROSEL
  IF (CODEBEAM(CURRELEM).EQ.1) THEN
    TYPE(CURRELEM) = 'ROLLER'
  ELSE
    TYPE(CURRELEM) = 'STUB'
  ENDIF
369 CONTINUE
*   WRITE TO FILES
DO 370 ROW = 1, NMCROSEL
  IF (CODEWINK(ROW).EQ.0.OR.CONDITON(ROW).EQ.'GAPPED') THEN
    ELEMDISP = 0.0D+00
  ELSE
    ELEMDISP = WINKDISP(ROW-LEFTLEM)
  ENDIF
+   WRITE (OUTUNIT2,3620) ROW, LOCATE(ROW), TYPE(ROW),
    ELEMDISP, CONDITON(ROW)
370 CONTINUE
  WRITE (OUTUNIT2,3630) NMCROSND, LOCATE(NMCROSND), UNITLENG
  WRITE (OUTUNIT2,3000)
  WRITE (OUTUNIT2,3560) NODEFORC, UNITFORC, NUMGAPS
  WRITE (OUTUNIT2,3570) PERCENT
  WRITE (OUTUNIT2,3000)
*
DO 380 ROW = 1, NMCROSND
  INDEXNUM = (DOFFPERND * ROW) -1
+   WRITE (OUTUNIT5,3610) ROW, NODEDISP(INDEXNUM), LPWVSNO
    (ROW), MAXPRESS(ROW), AVERLPW, HALFCONW(ROW)
380 CONTINUE
*
*   CLOSE OUTPUT FILES
*
  CLOSE (OUTUNIT1)
  CLOSE (OUTUNIT2)
  CLOSE (OUTUNIT3)
  CLOSE (OUTUNIT4)

```

```

        CLOSE (OUTUNIT5)
    ENDIF
*****
*   FORMAT STATEMENTS
*
3000  FORMAT ( 1X, ' ')
3010  FORMAT ( 8X, 'This is the output file ',A12,' from the program I',
+       'MPINGE written by:')
3020  FORMAT (12X, 'Paul Hoffercker at MAE Research Lab, Oklahoma State',
+       ' University.')
3030  FORMAT ( T11, 'Each given Hertzian Contact Half Width has a corr',
+       'esponding wound roll')
3040  FORMAT ( T12, 'deformation as shown. The wound roll foundation ',
+       'stiffness and load')
3050  FORMAT ( T23, 'per width at the deformation are also given.')
3060  FORMAT ( T12, 'This summary covers info from the other files inc',
+       'luding: gapping,')
3070  FORMAT ( T17, 'and the deformation of the winkler foundation ele',
+       'ments.')
3080  FORMAT ( T11, 'The lines for each finite element visually repres',
+       'ent Nip roll con-')
3090  FORMAT ( T18, 'tact on the wound roll for the corresponding load.')
3100  FORMAT ( T11, 'The given beam and winkler deformations and angle',
+       's are for each')
3110  FORMAT ( T27, 'increasing Nip load increment, up to the desired ',
+       'Nip force.')
3120  FORMAT (28X, 'The run date is: ',A20)
3130  FORMAT ( T11, 'The deformation profile includes the nodal displa',
+       'cement, load per')
3140  FORMAT ( T10, 'width, max pressure, average load per width, and ',
+       'the corresponding')
3150  FORMAT ( T14, 'halfwidth of contact across the wound roll at the',
+       ' max load.')
3200  FORMAT ( T7, 'Units',T20,'=',7X,A3)
3205  FORMAT ( T7, 'Wound Roll...')
3206  FORMAT ( T7, 'Stub Shaft...')
3207  FORMAT ( T7, 'Nip Roller...')
3208  FORMAT ( T7, 'Overall...')
3209  FORMAT ( T7, 'CMD distance from machine zero to...')
3210  FORMAT ( T7, 'K1',T20,'=',4X,F10.3,1X,A10,T50,'K2',T66,'= ',F10.3)
3220  FORMAT ( T7, 'Radius',T20,'=',4X,F10.3,1X,A10,T50,'Poissons Ratio',
+       T66,'= ',F10.3)
3230  FORMAT ( T7, 'Radius Eff',T20,'=',4X,F10.3,1X,A10)
3215  FORMAT ( T7, 'K3',T20,'=',4X,F10.3,1X,A10)
3250  FORMAT ( T7, 'Modulus',T20,'= ',F13.3,1X,A10)
3300  FORMAT ( T7, 'Modulus',T20,'= ',F13.3,1X,A10,T50,'MOI',T66,
+       '= ',F10.3,1X,A10)
3310  FORMAT ( T2, 'Half Width',T26,'Roll',T47,'Load',T66,'Foundation')
3320  FORMAT ( T2, 'of Contact',T25,'Deform',T45,'per Width',T67,'Stiff',
+       'ness')
3330  FORMAT ( T5,A10,T26,A10,T45,A10,T66,A10)
3340  FORMAT ( 1X,120(' '))
3350  FORMAT ( 1X,83(' '))
3355  FORMAT ( ' ',\ )
3360  FORMAT ( T7,'CMD Width',T20,'=',4X,F10.3,1X,A10,T50,'# of Elemen',
+       'ts',T66,'= ',T70,I4)
3362  FORMAT ( T7,'CMD Width',T20,'=',4X,F10.3,1X,A10)
3365  FORMAT ( T7,'Rigid left',T30,A1,T50,'Rigid Right',T73,A1)
3370  FORMAT ( T7,'Total Force',T20,'=',4X,F10.3,1X,A10,T50,'Force Ste',
+       'p Size',T66,'= ',F10.3,1X,A10)
3372  FORMAT ( T7,'Start Force',T20,'=',4X,F10.3,1X,A10,T50,'Iteration',
+       ' on?',T73,A1)

```

```

3373 FORMAT ( T7,'Nip Roller',T20,'=',4X,F10.3,1X,A10,T50,'Wound Roll',
+ T66,'= ',F10.3,1X,A10)
3380 FORMAT ( 1x,'Element',T13,'Element',T27,'Nip',T39,'Element',T51,
+ 'Nip to Wound')
3390 FORMAT ( 1X,'Number',T12,'Left Node',T25,'Element',T38,'Displace',
+ T51,'Roll Contact')
3395 FORMAT ( 1X,T12,'Location',T26,'Type')
3400 FORMAT ( 12X,A10,T40,A10)
3410 FORMAT ( /,' ',\ )
3412 FORMAT ( /,\ )
3420 FORMAT ( 'E ',\ )
3430 FORMAT ( '# ',\ )
3440 FORMAT ( /,1x,'Force ',\ )
3450 FORMAT ( 19X,I1,\ )
3460 FORMAT ( /,3x,A10,' ',\ )
3470 FORMAT ( 9(I1,1X),'0 ',\ )
3500 FORMAT ( 1x,'Force',',A10,3(F9.2,2X,F9.2,3X))
3510 FORMAT ( 1X,2(F10.5,9X), 2(F14.4,8X))
3520 FORMAT ( 19X,3(3X,'BEAM',7X,'WINK',5X))
3530 FORMAT ( 1X,F10.3,T14,96A2)
3540 FORMAT ( 1X,'U',I2,1X,A10,2X,3(3X,F7.5,4X,F7.5,2X))
3550 FORMAT ( 1X,'Rot',I2,1X,A10,3(3X,F7.5,4X,F7.5,2X))
3560 FORMAT ( T7,'At a Nip Force of ',F10.3,1x,A10,TL3,I4,' gaps',
+ ' occurred across the width.')
3570 FORMAT ( T7,'The gapping occurs in ',I4,'% of the elements.')
3580 FORMAT ( T4,'Node',T17,'Node',T28,'Load Per',T45,'Maximum',
+ T61,'Average',T76,'Halfwidth')
3590 FORMAT ( T3,'Number',T15,'Displace',T30,'Width',T44,'Pressure',
+ T61,'Load/ W',T75,'of Contact')
3600 FORMAT ( T16,A10,T28,A10,T43,A10,T60,A10,T77,A10)
3610 FORMAT ( T5,I2,T12,F10.6,4X,F10.4,3(3X,F13.4))
3620 FORMAT ( 1X,I2,T14,F5.2,T26,A6,T38,F8.6,T53,A7)
3630 FORMAT ( 1X,'Node ',I2,' ... ',T14,F5.2,T20,A10)
*****
*
* DEALLOCATE VARIABLE DIMENSION ARRAY
*
* DEALLOCATE (TYPE)
*
* RETURN TO SENDING PROGRAM
*
* RETURN
* END
*****
*****
*
* HERTZIAN, GENERATES VALUES OF LOAD PER WIDTH CORRESPONDING
* TO DEFORMATIONS
*
*****
*
* SUBROUTINE HERTZIAN(NUMOFPTS, HAFWIDTH, DEFORM, LOADPERW,
+ NIPROLLE, PI, POISSNIP, POISROLL, ASTART, DELTA, KONE, KTWO,
+ KTHREE, ROLPOIEF, NIPEPOIS, STIFNESS, RADEFF)
*
* DEFINE VARIABLES
*
* INTEGER NUMOFPTS, INC
* DOUBLE PRECISION POISROLL, EROLLRAD, NIPROLLE
* DOUBLE PRECISION ROLLERAD, WINKSTIF, LOADER, NIPEPOIS, POISSNIP
* DOUBLE PRECISION ASTART, DELTA, RADEFF, PI, ROLPOIEF, KTHREE
* DOUBLE PRECISION DEFORM(NUMOFPTS), LOADPERW(NUMOFPTS), KONE

```

```

      DOUBLE PRECISION HAFWIDTH(NUMOFPTS), STIFNESS(NUMOFPTS), KTW0
*
*
*   INITIALIZE VARIABLES
*
      INC = 1
      HAFWIDTH(INC) = ASTART
      NIPEPOIS = (1.0D+00 - (POISSNIP**2))/NIPROLLE
      ROLPOIEF = 1.0D+00-POISROLL**2
*
*   LOOP THROUGH DESIRED NUMBER OF INCREMENTS
*
      DO 400 INC = 1, NUMOFPTS
*
*       CALCULATE RADIAL DEFORMATION FROM HERTZIAN HALFWIDTH
*
          DEFORM(INC) = ((HAFWIDTH(INC))**2)/(RADEFF*4.0D+00)
*
*       CALCULATE RADIAL ROLL MODULUS FROM ROLLERAD FUNCTION
*
          EROLLRAD = ROLLERAD (KONE, KTW0, KTHREE, DEFORM(INC))
*
*       CALCULATE NIP LOAD PER WIDTH FROM LOADER FUNCTION
*
          LOADPERW(INC) = LOADER (PI, EROLLRAD, DEFORM(INC), ROLPOIEF,
+             NIPEPOIS)
*
*       CALCULATE WINKLER STIFFNESS FROM WINKSTIF FUNCTION
*
          STIFNESS(INC) = WINKSTIF(PI, EROLLRAD, KTW0, DEFORM(INC),
+             ROLPOIEF, NIPEPOIS, KTHREE)
*
*       INCREMENT HALF WIDTH
*
          IF(INC.LT.NUMOFPTS) HAFWIDTH(INC+1) = HAFWIDTH(INC) + ADELTA
*
400  CONTINUE
*
*   RETURN TO MAIN PROGRAM
*
      RETURN
      END
*****
*****
*
*   ROLLERAD, FUNCTION WHICH GENERATES WOUND ROLL'S RADIAL ELASTIC
*   MODULUS
*
*****
*
      DOUBLE PRECISION FUNCTION ROLLERAD (KONE, KTW0, KTHREE, CURNTDEF)
*
*   DEFINE VARIABLES
*
      DOUBLE PRECISION KONE, KTW0, KTHREE, CURNTDEF
*
*   CALCULATE ROLLERAD
*
      ROLLERAD = KONE * KTW0 * DEXP(KTW0 * CURNTDEF) + KTHREE
*
*   SET ROLLERAD = 1 IF IT GOES NEGATIVE (DUE TO K3
*   BEING LARGE NEG.)
*

```

```

      IF (ROLLERAD.LE.0.0D+00) ROLLERAD = 0.1D+01
*
*   RETURN TO SENDING ROUTINE
*
      RETURN
      END
*****
*****
*
*   WINKSTIF, FUNCTION WHICH GENERATES WINKLER FOUNDATION STIFFNESS
*
*****
*****
*
      DOUBLE PRECISION FUNCTION WINKSTIF (PI, EROLLRAD, KTW0, CURNTDEF,
+      ROLPOIEF, NIPEPOIS, KTHREE)
*
      DEFINE VARIABLES
*
      DOUBLE PRECISION PI, EROLLRAD, KTW0, CURNTDEF, ROLPOIEF
      DOUBLE PRECISION KTHREE, NIPEPOIS, EFFECTE
*
      CALCULATE EFFECTIVE MODULUS FOR BOTH NIP AND WOUND ROLL
*
      EFFECTE = (ROLPOIEF/EROLLRAD) + NIPEPOIS
*
      CALCULATE WINKSTIF
*
      WINKSTIF = (PI/(EFFECTE**2))* (EFFECTE+ (KTWO* CURNTDEF* ROLPOIEF*
+      (EROLLRAD - KTHREE))/(EROLLRAD**2))
*
      RETURN TO SENDING ROUTINE
*
      RETURN
      END
*****
*****
*
*   LOADER, FUNCTION WHICH CALCULATES NIP LOAD PER ROLL UNIT WIDTH
*   VIA HERTZIAN CONTACT FORMULA
*
*****
*****
*
      DOUBLE PRECISION FUNCTION LOADER(PI, EROLLRAD, CURNTDEF, ROLPOIEF,
+      NIPEPOIS)
*
      DEFINE VARIABLES
*
      DOUBLE PRECISION PI, EROLLRAD, ROLPOIEF, CURNTDEF, NIPEPOIS
*
      CALCULATE LOADER
*
      LOADER = PI*CURNTDEF / ((ROLPOIEF/EROLLRAD) + NIPEPOIS)
*
      RETURN TO SENDING ROUTINE
*
      RETURN
      END
*****
*****
*
*   CONTACT, SUBROUTINE WHICH IMPINGES NIP BEAM INTO WINKLER ROLL
*

```

```

*****
*
  SUBROUTINE CONTACT(MARKER, NMCROSEL, NMCROSND, DOFFERND, PERCENT,
+   KONE, KTWO, FORCERES, NIPFORCE, NIPEPOIS, PI, NODEFORC,
+   DELTAFOR, ROLPOIEF, NODEDISP, ELLENGTH, CONDITON, STATUS,
+   NUMGAPS, TOTALDOF, DATE, RADROLL, POISROLL, RADNIP, POISSNIP,
+   NIPROLLE, AVERLPW, NIPROLLI, NIPROLLW, NUMOFPTS, HAFWIDTH,
+   DEFORM, STIFNESS, LOADPERW, TIME1, WINKDISP, LPWVSNOD,
+   MAXPRESS, UNITTYPE, UNITLENG, UNITFORC, UNITFPL, UNITPRES,
+   UNITMOI, UNITANGL, RADEFF, TEMPDISP, HALFCONW, CODEBEAM,
+   CODEWINK, NIPSTUBE, NIPSTUBI, LOCATE, LEFTLEM, NMWINKND,
+   NMWINKEL, KTHREE, THETAPRN, CONSTLFT, CONSTRGH, STARTFOR,
+   REPEAT, DISPLRES)
*
*   DEFINE VARIABLES
*
  INTEGER MARKER, NMCROSEL, NMCROSND, DOFFERND, TOTALDOF
  INTEGER PERCENT, NUMOFPTS, TIME1, LEFTLEM, NMWINKND, NMWINKEL
  INTEGER CODEBEAM(NMCROSEL), CODEWINK(NMCROSEL), CYCLE
  DOUBLE PRECISION KONE, KTWO, KTHREE, FORCERES, NIPFORCE, NIPEPOIS
  DOUBLE PRECISION NODEFORC, DELTAFOR, PI, ROLPOIEF, NIPSTUBI
  DOUBLE PRECISION ELLENGTH(NMCROSEL), NODEDISP(TOTALDOF), DISPLRES
  DOUBLE PRECISION LPWVSNOD(NMCROSND), MAXPRESS(NMCROSND)
  DOUBLE PRECISION WINKDISP(NMCROSEL), K[ALLOCATABLE](:,:)
  DOUBLE PRECISION FORCE[ALLOCATABLE](:), ELEMSTIF[ALLOCATABLE](:)
  DOUBLE PRECISION HALFCONW(NMCROSND), ROLLERAD, WINKSTIF
  DOUBLE PRECISION TEMPDISP(NMCROSND*DOFFERND+1,6), NIPSTUBE
  DOUBLE PRECISION RADROLL, POISROLL, RADNIP, POISSNIP, RADEFF
  DOUBLE PRECISION NIPROLLE, AVERLPW, NIPROLLI, NIPROLLW
  DOUBLE PRECISION NIPSTUBW, LEFTSTUB, LEFTWINK, WINKWIDT
  DOUBLE PRECISION HAFWIDTH(NUMOFPTS), DEFORM(NUMOFPTS)
  DOUBLE PRECISION STIFNESS(NUMOFPTS), LOADPERW(NUMOFPTS)
  DOUBLE PRECISION FUNCMIDL[ALLOCATABLE](:), STARTFOR, MAXDIFF, DIFF
  DOUBLE PRECISION MINDISP[ALLOCATABLE](:), MAXDISP[ALLOCATABLE](:)
  DOUBLE PRECISION MIDLDISP[ALLOCATABLE](:), LOCATE(NMCROSND)
  DOUBLE PRECISION FUNCMIN[ALLOCATABLE](:), FUNCMAX[ALLOCATABLE](:)
  CHARACTER CONDITON(NMCROSEL)*7, STATUS(NMCROSEL)*2, DATE*20
  CHARACTER UNITTYPE*3, UNITLENG*10, UNITFORC*10, UNITFPL*10
  CHARACTER UNITMOI*10, UNITPRES*10, UNITANGL*10, THETAPRN*1
  CHARACTER CONSTLFT*1, CONSTRGH*1, REPEAT*1
  INTEGER NUMSTEPS, NUMGAPS, CURRSTEP, CURRNODE
  INTEGER INDEXNUM, LOCALROW, LOCALCOL, COLUMN, BEAMNODE, WINKNODE
  INTEGER GBROW, GBCOL, GWROW, GWCOL, LINKNODE, CURRELEM, CURWINEL
  INTEGER POINTVEC[ALLOCATABLE](:)
  DOUBLE PRECISION EROLLRAD, LASTDFOR, BEAMESM(4,4), WINKESM(4,4)
  LOGICAL PROCEED, DONE
*
*   ALLOCATE VARIABLE DIMENSION ARRAYS
*
  ALLOCATE (MINDISP(NMWINKND))
  ALLOCATE (MAXDISP(NMWINKND))
  ALLOCATE (MIDLDISP(NMWINKND))
  ALLOCATE (FUNCMIN(NMWINKND))
  ALLOCATE (FUNCMAX(NMWINKND))
  ALLOCATE (FUNCMIDL(NMWINKND))
  ALLOCATE (FORCE(TOTALDOF))
  ALLOCATE (ELEMSTIF(NMCROSEL*2))
  ALLOCATE (K(TOTALDOF,TOTALDOF))
  ALLOCATE (POINTVEC(TOTALDOF))
*
*   INITIALIZE MATRICES AND VARIABLES
*

```

```

DO 500 CURRNODE = 1, TOTALDOF
    FORCE(CURRNODE) = 0.0D+00
    NODEDISP(CURRNODE) = 0.0D+00
500 CONTINUE
*
DO 505 LOCALROW = 1, 4
    DO 510 LOCALCOL = 1, 4
        BEAMESM(LOCALROW,LOCALCOL) = 0.0D+00
        WINKESM(LOCALROW,LOCALCOL) = 0.0D+00
510 CONTINUE
505 CONTINUE
*
* DETERMINE THE NUMBER OF FORCE (DELTAFOR) STEPS REQUIRED TO REACH
* DESIRED NIPFORCE, AND THE FINAL STEP SIZE REQUIRED
*
NUMSTEPS = IDINT((DNINT((NIPFORCE - STARTFOR)/FORCERES))/
+ (DELTAFOR/FORCERES))+1
LASTDFOR = DMOD((DNINT((NIPFORCE - STARTFOR)/FORCERES)),
+ (DELTAFOR/FORCERES))*FORCERES
IF (LASTDFOR.EQ.0.0D+00) THEN
    LASTDFOR = DELTAFOR
ELSE
    NUMSTEPS = NUMSTEPS + 1
ENDIF
*
* SET FIRST NODEFORC TO STARTING FORCE - DELTA FORCE, SET OTHER
* VARIABLES
*
NODEFORC = STARTFOR - DELTAFOR
CURRSTEP = 0
PROCEED = .FALSE.
*
* COMPUTE THE DISPLACEMENT RESULTING FROM THE NIP FORCE
*
DO WHILE (.NOT.PROCEED)
*
*     COUNT NUMBER OF NIP FORCE STEPS
*
*     CURRSTEP = CURRSTEP + 1
*
*     IF THIS IS LAST TIME THROUGH, END WITH REMAINDER FORCE DESIRED
*
*     IF (CURRSTEP.EQ.NUMSTEPS) THEN
*         PROCEED = .TRUE.
*         DELTAFOR = LASTDFOR
*     ENDIF
*
*     ESTABLISH MINIMUM DISPLACEMENT VECTOR IF CONVERGENCE IS DESIRED,
*     AND SET COUNTER CYCLE TO TRACK NUMBER OF LOOPS
*
*     IF (REPEAT.EQ.'Y' .OR. REPEAT.EQ.'y') THEN
*         DO 515 CURRNODE = 1, NMWINKND
*             MINDISP(CURRNODE) = NODEDISP(((CURRNODE*DOFPERND)-1) +
+
515         CONTINUE
*             CYCLE = 0
*         ENDIF
*
*     INCREMENT FORCE AT NODES
*
*     NODEFORC = NODEFORC + DELTAFOR
*

```



```

*          DIVIDE NIP FORCE BETWEEN TWO END NODES
*
FORCE(1) = NODEFORC/2.0D+00
FORCE(NMCROSD*DOFFPERND-1) = NODEFORC/2.0D+00
*
*          FIND STIFFNESSES, FEM DEFORMATIONS, AND ALLOW THE LOOP
*          THROUGH DEFORMATION DETERMINATION IF USER DESIRES
*          CONVERGENCE
*
DONE = .FALSE.
DO WHILE (.NOT.DONE)
*
*          SET OR RESET THE STIFFNESS MATRIX
*
DO 520 CURRNODE = 1, TOTALDOF
  DO 525 COLUMN = CURRNODE, TOTALDOF
    K(CURRNODE,COLUMN) = 0.0D+00
525  CONTINUE
520  CONTINUE
*
*          CALCULATE THE STIFFNESS OF THE BEAM ELEMENTS AND ASSEMBLE
*          INTO STIFFNESS MATRIX
*
DO 530 CURRELEM = 1, NMCROSEL
*
*          IF BEAM ELEMENT IS THE NIP ROLLER (CODE=1) THEN USE THE
*          CORRESPONDING E AND I, OTHERWISE USE STUB SHAFT
*          VALUES
*
IF (CODEBEAM(CURRELEM).EQ.1) THEN
  ELEMSTIF(CURRELEM) = NIPROLLE*NIPROLLI/
    (ELLENGTH(CURRELEM)**3)
+ ELSE
+   ELEMSTIF(CURRELEM) = NIPSTUBE*NIPSTUBI/
    (ELLENGTH(CURRELEM)**3)
+ ENDIF
*
*          GET ELEMENT STIFFNESS MATRIX
*
CALL BEAMELEM(ELLENGTH(CURRELEM), ELEMSTIF(CURRELEM),
+   BEAMESM)
*
*          DETERMINE GLOBAL AND LOCAL POSITIONING INDICES, AND THEN
*          FILL THE K STIFFNESS MATRIX
*
BEAMNODE = DOFFPERND*CURRELEM - 1
DO 535 LOCALROW = 1, 4
  GBROW = BEAMNODE + LOCALROW -1
  DO 540 LOCALCOL = LOCALROW, 4
    GBCOL = BEAMNODE + LOCALCOL -1
    K(GBROW,GBCOL) = K(GBROW,GBCOL) +
+     BEAMESM(LOCALROW,LOCALCOL)
540  CONTINUE
535  CONTINUE
530  CONTINUE
*
*          DETERMINE MAX AND MIN VALUES FOR BRACKETING DEFORMATIONS
*          FOR CONVERGENCE
*
IF (REPEAT.EQ.'Y' .OR. REPEAT.EQ.'y') THEN
*
*          LOAD MAX POSSIBLE DEFORMATIONS, AND FUNCTION

```



```

*           CORRESPONDING TO MIN POSSIBLE DEFORMATIONS
*
IF (CYCLE.EQ.1) THEN
  DO 545 CURRNODE = 1, NMWINKND
    MAXDISP(CURRNODE) = NODEDISP(((CURRNODE*
+      DOFFERND) -1) + NMCROSND*DOFFERND)
+      FUNCMIN(CURRNODE) = MAXDISP(CURRNODE) -
545      MINDISP(CURRNODE)
    CONTINUE
  ENDIF
*
*           LOAD FUNCTION CORRESPONDING TO MAX POSSIBLE DEFORMATIONS
*
IF (CYCLE.EQ.2) THEN
  DO 550 CURRNODE = 1, NMWINKND
    FUNCMAX(CURRNODE) = NODEDISP(((CURRNODE*
+      DOFFERND) -1) + NMCROSND*DOFFERND) -
+      MAXDISP(CURRNODE)
550    CONTINUE
  ENDIF
*
*           LOAD MIDDLE DEFORMATIONS AND FUNCTIONS
*
IF (CYCLE.GE.2) THEN
  DO 555 CURRNODE = 1, NMWINKND
    MIDLDISP(CURRNODE) = (MAXDISP(CURRNODE) +
+      MINDISP(CURRNODE))/2.0D+00
+      NODEDISP(((CURRNODE*DOFFERND)-1) + NMCROSND*
555      DOFFERND) = MIDLDISP(CURRNODE)
    CONTINUE
  ENDIF
ENDIF
*
*           CALCULATE STIFFNESS OF THE WINKLER ELEMENTS AND ASSEMBLE
*           INTO STIFFNESS MATRIX
*
NUMGAPS = 0
DO 560 CURRELEM = 1, NMCROSEL
*
*           IF WINKLER ELEMENT EXISTS (CODE=1) THEN...
*           CALCULATE DISPLACEMENT OF WINKLER ELEMENTS AS
*           AVERAGES OF NODE DISPLACEMENTS
*           CALCULATE RADIAL ROLL MODULUS FROM ROLLERAD FUNCTION
*           CALCULATE WINKLER STIFFNESS FROM WINKSTIF FUNCTION
*
IF (CODEWINK(CURRELEM).EQ.1) THEN
  CURWINEL = CURRELEM - LEFTELEM
  INDEXNUM = NMCROSEL + CURWINEL
  WINKDISP(CURWINEL) = (NODEDISP(DOFFERND*INDEXNUM +
+      3) + NODEDISP(DOFFERND*INDEXNUM +1))/ 2.0D+00
+      EROLLRAD = ROLLERAD (KONE, KTW0, KTHREE,
+      WINKDISP(CURWINEL))
+      ELEMSTIF(INDEXNUM) = ELLENGTH(CURRELEM)/420.0D+00*
+      WINKSTIF(PI, EROLLRAD, KTW0, WINKDISP
+      (CURWINEL), ROLPOIEF, NIPEPOIS, KTHREE)
*
*           GET ELEMENT STIFFNESS MATRIX
*
CALL WINKELEM(ELLENGTH(CURRELEM), ELEMSTIF
+      (INDEXNUM), WINKESM)
*

```

```

*           DETERMINE GLOBAL AND LOCAL POSITIONING INDICES, AND
*           THEN FILL THE K STIFFNESS MATRIX
*
WINKNODE = DOPPERND*INDEXNUM + 1
DO 565 LOCALROW = 1, 4
  GWROW = WINKNODE + LOCALROW -1
  DO 570 LOCALCOL = LOCALROW, 4
    GWCOL = WINKNODE + LOCALCOL -1
    K(GWROW,GWCOL) = K(GWROW,GWCOL) +
      WINKESM(LOCALROW,LOCALCOL)
  CONTINUE
570 CONTINUE
565 CONTINUE
*
* CHECK FOR GAPPING...
* IF EITHER NODE OF AN ELEMENT GAPS, THEN THE
* ELEMENT GAPS
*
LINKNODE = WINKNODE + NMWINKND*DOPPERND
IF (NODEDISP(LINKNODE).LT.0.0D+00 .OR.
+   NODEDISP(LINKNODE+2).LT.0.0D+00
*   ADDED...
+   .OR. NODEDISP(WINKNODE).LT.0.0D+00 .OR.
+   NODEDISP(WINKNODE+2).LT.0.0D+00) THEN
  CONDITON(CURRELEM) = 'GAPPED '
  STATUS(CURRELEM) = ' '
  NUMGAPS = NUMGAPS + 1
ELSE
  CONDITON(CURRELEM) = 'NO_GAP '
  STATUS(CURRELEM) = ' _ '
ENDIF
*
* CHECK NODE GAPPING TO DETERMINE LINKING
*
BEAMNODE = DOPPERND*CURRELEM - 1
IF (NODEDISP(LINKNODE).LT.0.0D+00
*   ADDED...
+   .OR. NODEDISP(WINKNODE).LT.0.0D+00) THEN
  K(BEAMNODE,LINKNODE) = 0.0D+00
  K(BEAMNODE+1,LINKNODE+1) = 0.0D+00
ELSE
  K(BEAMNODE,LINKNODE) = 1.0D+00
  K(BEAMNODE+1,LINKNODE+1) = 1.0D+00
ENDIF
IF (NODEDISP(LINKNODE+2).LT.0.0D+00
*   ADDED...
+   .OR. NODEDISP(WINKNODE+2).LT.0.0D+00) THEN
  K(BEAMNODE+2,LINKNODE+2) = 0.0D+00
  K(BEAMNODE+3,LINKNODE+3) = 0.0D+00
ELSE
  K(BEAMNODE+2,LINKNODE+2) = 1.0D+00
  K(BEAMNODE+3,LINKNODE+3) = 1.0D+00
ENDIF
*
K(WINKNODE,LINKNODE) = -1.0D+00
K(WINKNODE+1,LINKNODE+1) = -1.0D+00
K(WINKNODE+2,LINKNODE+2) = -1.0D+00
K(WINKNODE+3,LINKNODE+3) = -1.0D+00
ELSE
  CONDITON(CURRELEM) = 'NO ROLL'
  STATUS(CURRELEM) = 'xx'
ENDIF
560 CONTINUE

```

```

*
* CHECK IF PERCENTAGE OF GAPPED ELEMENTS IS 75% OR GREATER
* AND EXIT LOOPS IF IT IS
*
PERCENT = 100. * NUMGAPS / NMWINKEL
IF (PERCENT.GE.75) THEN
  PROCEED = .TRUE.
  DONE = .TRUE.
ENDIF

*
* CHECK FOR CONSTRAINED NIP ENDS, THEN FILL OUT K MATRIX
*
IF (CONSTLFT.EQ.'Y'.OR.CONSTLFT.EQ.'y') K(2,2) = 1.0D+00
IF (CURRNODE.EQ.NMCROSND*DOFFPERND.OR.
+ COLUMN.EQ.NMCROSND*DOFFPERND) THEN
  K(NMCROSND*DOFFPERND,NMCROSND*DOFFPERND) = 1.0D+00
ENDIF

*
DO 575 CURRNODE = 1, TOTALDOF -1
  DO 580 COLUMN = CURRNODE +1, TOTALDOF
    IF (CONSTLFT.EQ.'Y'.OR.CONSTLFT.EQ.'y') THEN
      IF (CURRNODE.EQ.2.OR.COLUMN.EQ.2) THEN
        K(CURRNODE,COLUMN) = 0.0D+00
      ENDIF
    ENDIF
    IF (CONSTRGH.EQ.'Y'.OR.CONSTRGH.EQ.'y') THEN
      IF (CURRNODE.EQ.NMCROSND*DOFFPERND.OR.
+ COLUMN.EQ.NMCROSND*DOFFPERND) THEN
        K(CURRNODE,COLUMN) = 0.0D+00
      ENDIF
    ENDIF
    K(COLUMN,CURRNODE) = K(CURRNODE,COLUMN)
  CONTINUE
580 CONTINUE
575

*
* SOLVE THE STIFFNESS MATRIX-FORCE COMBINATION FOR DISPLACEMENT
* OF BEAM-WINKLER COMBINATION; U = K(^-1)F
*
CALL LUDECOMP(TOTALDOF, K, POINTVEC)
CALL SUBSTIT(TOTALDOF, K, POINTVEC, FORCE, NODEDISP)

*
* CHECK FOR CONVERGENCE, BY SQUEEZING IN THE BRACKETING DOMAIN
*
IF (REPEAT.EQ.'Y' .OR. REPEAT.EQ.'y') THEN
  IF (CYCLE.GE.3) THEN
    MAXDIFF = 0.0
    DO 585 CURRNODE = 1, NMWINKND
      FIND DEFORMATION FUNCTION IN MIDDLE OF
      BRACKETED DOMAIN

      FUNCMIDL(CURRNODE) = NODEDISP(((CURRNODE*
+ DOFFPERND) -1) + NMCROSND*DOFFPERND) -
+ MIDLDISP(CURRNODE)

      IF DEF. FUNC. AT MIDDLE IS ZERO, THE DEFORMATION
      IS EXACTLY CONVERGED

      IF (FUNCMIDL(CURRNODE).EQ.0.0) THEN
        MINDISP(CURRNODE) = MIDLDISP(CURRNODE)
        MAXDISP(CURRNODE) = MIDLDISP(CURRNODE)
      ENDIF
    ENDIF
  ENDIF
ENDIF

```

```

*           ADJUST RIGHT END OF DEFORMATION INWARD
*
+           ELSE IF (FUNCMAX(CURRNODE) .GE. 0.0 .AND.
+             FUNCMIDL(CURRNODE) .GE. 0.0 .OR.
+             FUNCMAX(CURRNODE) .LT. 0.0 .AND.
+             FUNCMIDL(CURRNODE) .LT. 0.0) THEN
*             MAXDISP(CURRNODE) = MIDLDISP(CURRNODE)
*             FUNCMAX(CURRNODE) = FUNCMIDL(CURRNODE)
*
*           ADJUST LEFT END OF DEFORMATION INWARD
*
*           ELSE
*             MINDISP(CURRNODE) = MIDLDISP(CURRNODE)
*             FUNCMIN(CURRNODE) = FUNCMIDL(CURRNODE)
*           ENDIF
*
*           DETERMINE DIFFERENCE BETWEEN LEFT AND RIGHT
*           END DEFORMATIONS, AND CHECK FOR MAX DIFF
*
*           DIFF = MAXDISP(CURRNODE) - MINDISP(CURRNODE)
*           IF (DIFF .GT. MAXDIFF) MAXDIFF = DIFF
585          CONTINUE
*
*           IF MAXIMUM DIFFERENCE IS SUFFICIENTLY SMALL...IT
*           HAS CONVERGED
*
*           IF (MAXDIFF .LT. DISPLRES) DONE = .TRUE.
*           ENDIF
*           CYCLE = CYCLE + 1
*
*           IF USER DOES NOT WANT CONVERGED DEFORMATIONS, THE CONVERG-
*           ENCE LOOP IS DONE
*
*           ELSE
*             DONE = .TRUE.
*           ENDIF
*
*           END LOOP FOR CONVERGENCE
*
*         END DO
*
*       LOAD TEMPORARY ARRAY FOR PRINTING PURPOSES
*
*       TEMPDISP (1, TIME1) = NODEFORC
*       TEMPDISP (1, TIME1+1) = NODEFORC
*       DO 590 CURRNODE = 1, NMCROSND*DOFPERND
*         TEMPDISP (CURRNODE+1, TIME1) = NODEDISP(CURRNODE)
*         CURRELEM = INT((CURRNODE+1)/DOFPERND)
*         CURWINEL = CURRELEM - LEFTELEM
*         INDEXNUM = NMCROSEL + CURWINEL
*         WINKNODE = DOFPERND*INDEXNUM + 1 + MOD(CURRNODE+1, DOFPERND)
*
*         CHECK AND ADJUST 'CURRELEM'...IF AT LAST WINKLER NODE OR
*         AT LAST OVERALL NODE (NMCROSND), SEE IF ELEMENT PRE-
*         CEEDING NODE IS A WINKLER ELEMENT, SO OUTDATA WILL
*         PRINT ITS DEFORMATION
*         IF (CURWINEL .EQ. NMWINKEL+1) CURRELEM = NMWINKEL
*         IF (CODEWINK(CURRELEM) .EQ. 1) THEN
*           TEMPDISP (CURRNODE+1, TIME1+1) = NODEDISP(WINKNODE)
*         ELSE
*           TEMPDISP (CURRNODE+1, TIME1+1) = 0.0D+00
*         ENDIF
*       ENDIF

```

```

590      CONTINUE
*
*      IF AT LAST TIME THROUGH, SET TIME1 VALUE SO OUTDATA WILL PRINT
*
*      IF (PROCEED) TIME1 = 6
*
*      TIME1 = TIME1 + 2
*
*      PRINT OUT VISUAL OF FORCE INCREMENT AND CORRESPONDING GAPS
*
*      CALL OUTDATA(MARKER, DATE, KONE, Ktwo, RADROLL, POISROLL,
+         RADNIP, POISSNIP, NIPROLLE, AVERLPW, NIPROLLI, NMCROSEL,
+         NIPFORCE, NIPROLLW, NODEFORC, NUMGAPS, PERCENT, NUMOFPTS,
+         HAFWIDTH, DEFORM, STIFNESS, LOADPERW, CONDTION, STATUS,
+         NMCROSND, NODEDISP, DOFFERND, TIME1, DELTAFOR, WINKDISP,
+         LPWVSND, MAXPRESS, UNITTYPE, UNITLENG, UNITFORC, UNITFPL,
+         UNITPRES, UNITMOI, UNITANGL, RADEFF, TEMPDISP, HALFCONW,
+         LOCATE, CODEBEAM, NIPSTUBW, LEFTSTUB, LEFTWINK, NIPSTUBE,
+         NIPSTUBI, WINKWIDT, NMWINKEL, CODEWINK, LEFTELEM, KTHREE,
+         THETAPRN, CONSTLFT, CONSTRGH, STARTFOR, REPEAT)
*
*      END LOOP THROUGH FORCES
*
*      END DO
*
*      UPDATE ELEMENT DISPLACEMENTS
*
*      DO 595 CURRELEM = 1, NMCROSEL
*          IF (CODEWINK(CURRELEM).EQ.1) THEN
*              CURWINEL = CURRELEM - LEFTELEM
*              INDEXNUM = NMCROSEL + CURWINEL
*              WINKDISP(CURWINEL) = (NODEDISP(DOFFERND*INDEXNUM +3) +
+              NODEDISP(DOFFERND*INDEXNUM +1))/ 2.0D+00
*          ENDIF
595  CONTINUE
*
*      DEALLOCATE VARIABLE DIMENSION ARRAYS
*
*      DEALLOCATE (MINDISP)
*      DEALLOCATE (MAXDISP)
*      DEALLOCATE (MIDLDISP)
*      DEALLOCATE (FUNCMIN)
*      DEALLOCATE (FUNCMAX)
*      DEALLOCATE (FUNCMIDL)
*      DEALLOCATE (FORCE)
*      DEALLOCATE (ELEMSTIF)
*      DEALLOCATE (K)
*      DEALLOCATE (POINTVEC)
*
*      RETURN TO MAIN PROGRAM
*
*
*      RETURN
*      END
*****
*****
*
*      BEAMELEM, CALCULATES THE BEAM ELEMENT STIFFNESS MATRIX
*
*****
*
SUBROUTINE BEAMELEM(LENGTH,STIFF,BEAMESM)

```

```

*
*   DEFINE VARIABLES
*
INTEGER ROW, COLUMN
DOUBLE PRECISION LENGTH, STIFF, BEAMESM(4,4)
*
*   CALCULATE MATRIX ENTRIES
*
BEAMESM (1,1) = STIFF * 12.0D+00
BEAMESM (1,2) = STIFF * 6.0D+00 * LENGTH
BEAMESM (1,3) = BEAMESM (1,1) * (-1.0D+00)
BEAMESM (1,4) = BEAMESM (1,2)
BEAMESM (2,2) = STIFF * 4.0D+00 * (LENGTH**2)
BEAMESM (2,3) = BEAMESM (1,2) * (-1.0D+00)
BEAMESM (2,4) = STIFF * 2.0D+00 * (LENGTH**2)
BEAMESM (3,3) = BEAMESM (1,1)
BEAMESM (3,4) = BEAMESM (1,2) * (-1.0D+00)
BEAMESM (4,4) = BEAMESM (2,2)
*
*   FILL OUT MATRIX
*
DO 10 COLUMN = 1, 3
  DO 20 ROW = COLUMN + 1, 4
    BEAMESM (ROW, COLUMN) = BEAMESM (COLUMN, ROW)
20  CONTINUE
10  CONTINUE
*
*   RETURN TO CONTACT SUBROUTINE
*
RETURN
END
*****
*****
*
*   WINKELEM, CALCULATES THE WINKLER FOUNDATION ELEMENT STIFFNESS
*   MATRIX
*
*****
*****
*
SUBROUTINE WINKELEM(LENGTH,STIFF,WINKESM)
*
*   DEFINE VARIABLES
*
INTEGER ROW, COLUMN
DOUBLE PRECISION LENGTH, STIFF, WINKESM(4,4)
*
*   CALCULATE MATRIX ENTRIES
*
WINKESM (1,1) = STIFF * 156.0D+00
WINKESM (1,2) = STIFF * 22.0D+00 * LENGTH
WINKESM (1,3) = STIFF * 54.0D+00
WINKESM (1,4) = STIFF * (-13.0D+00) * LENGTH
WINKESM (2,2) = STIFF * 4.0D+00 * (LENGTH**2)
WINKESM (2,3) = WINKESM (1,4) * (-1.0D+00)
WINKESM (2,4) = STIFF * (-3.0D+00) * (LENGTH**2)
WINKESM (3,3) = WINKESM (1,1)
WINKESM (3,4) = WINKESM (1,2) * (-1.0D+00)
WINKESM (4,4) = WINKESM (2,2)
*
*   FILL OUT MATRIX
*
DO 10 COLUMN = 1, 3

```

```

                DO 20 ROW = COLUMN + 1, 4
                  WINKESM (ROW, COLUMN) = WINKESM (COLUMN, ROW)
20          CONTINUE
10          CONTINUE
*
*          RETURN TO CONTACT SUBROUTINE
*
          RETURN
          END
*****
*****
*
*          LUDECOMP, DECOMPOSES THE STIFFNESS MATRIX INTO UPPER AND LOWER
*          TRIANGULAR MATRICES, PA = LU
*
*****
*
          SUBROUTINE LUDECOMP(DIMENSION, A, POINTVEC)
*
          DEFINE VARIABLES
*
          INTEGER DIMENSION, POINTVEC(DIMENSION)
          INTEGER ROW, NEXTROW, COLUMN, HOLDER
          DOUBLE PRECISION A(DIMENSION, DIMENSION)
*
          INTIALIZE THE POINTER VECTOR
*
          DO 10 ROW = 1, DIMENSION
            POINTVEC(ROW) = ROW
10          CONTINUE
*
          START FACTORIZATION
*
          DO 20 ROW = 1, DIMENSION - 1
*
            LOOK FOR LARGEST ELEMENT IN COLUMN TO USE AS PIVOT AND SWITCH
            THE INDICES UNTIL ITS ROW IS ABOVE THE REST
*
            DO 30 NEXTROW = ROW + 1, DIMENSION
              IF ( ABS(A(POINTVEC(NEXTROW),ROW)) .GT.
+                ABS(A(POINTVEC(ROW),ROW)) ) THEN
                HOLDER = POINTVEC(ROW)
                POINTVEC(ROW) = POINTVEC(NEXTROW)
                POINTVEC(NEXTROW) = HOLDER
              ENDIF
30          CONTINUE
*
          CHECK FOR MATRIX SINGULARITY AND COMPLETE FACTORIZATION
*
          IF ( A(POINTVEC(ROW),ROW) .NE. 0.0D+00 ) THEN
            DO 40 NEXTROW = ROW + 1, DIMENSION
              A(POINTVEC(NEXTROW),ROW) = A(POINTVEC(NEXTROW),ROW) /
+                A(POINTVEC(ROW),ROW)
            DO 50 COLUMN = ROW + 1, DIMENSION
              A(POINTVEC(NEXTROW),COLUMN) = A(POINTVEC(NEXTROW),
+                COLUMN) - A(POINTVEC(NEXTROW),ROW) *
+                A(POINTVEC(ROW),COLUMN)
50          CONTINUE
40          CONTINUE
          ELSE
            PRINT *, 'THE MATRIX IS SINGULAR'
          ENDIF

```

```

20  CONTINUE
*
*  RETURN TO CONTACT SUBROUTINE
*
      RETURN
      END
*****
*****
*
*  SUBSTIT, SUBSTITUTION ROUTINE TO SOLVE LY = PB FOR Y
*  AND UX = Y FOR X
*
*****
*
      SUBROUTINE SUBSTIT(DIMENSION, A, POINTVEC, B, X)
*
*  DEFINE VARIABLES
*
      INTEGER DIMENSION, POINTVEC(DIMENSION)
      INTEGER ROW, COLUMN
      DOUBLE PRECISION A(DIMENSION, DIMENSION), B(DIMENSION), X(DIMENSION)
      DOUBLE PRECISION Y[ALLOCATABLE](:), HOLDER
*
*  ALLOCATE VARIABLE DIMENSION ARRAYS
*
      ALLOCATE (Y(DIMENSION))
*
*  FORWARD SUBSTITUTION
*
      Y(1) = B(POINTVEC(1))
      DO 10 ROW = 2, DIMENSION
          HOLDER = 0.0D+00
          DO 20 COLUMN = 1, ROW - 1
              HOLDER = HOLDER + A(POINTVEC(ROW), COLUMN)*Y(COLUMN)
20          CONTINUE
          Y(ROW) = B(POINTVEC(ROW)) - HOLDER
10      CONTINUE
*
*  BACK SUBSTITUTION
*
      IF ( A(POINTVEC(DIMENSION), DIMENSION) .NE. 0.0D+00 ) THEN
          X(DIMENSION) = Y(DIMENSION) / A(POINTVEC(DIMENSION), DIMENSION)
          DO 30 ROW = DIMENSION - 1, 1, -1
              HOLDER = 0.0D+00
              DO 40 COLUMN = ROW + 1, DIMENSION
                  HOLDER = HOLDER + A(POINTVEC(ROW), COLUMN)*X(COLUMN)
40          CONTINUE
          X(ROW) = (Y(ROW) - HOLDER) / A(POINTVEC(ROW), ROW)
30      CONTINUE
      ELSE
          PRINT *, 'THE MATRIX IS SINGULAR'
      ENDIF
*
*  DEALLOCATE VARIABLE DIMENSION ARRAYS
*
      DEALLOCATE (Y)
*
*  RETURN TO CONTACT SUBROUTINE
*
      RETURN
      END
*****

```



```

*****
*
*   PROFILE, FUNCTION WHICH GENERATES THE LOAD PER WIDTH, AND
*   MAX PRESSURE ACROSS THE WOUND ROLL WIDTH
*
*****
*
SUBROUTINE PROFILE(NMCROSEL, NMCROSND, DOFPERND, PI, RADEFF,
+   WINKWIDT, NODEDISP, LPWVSND, MAXPRESS, ELLENGTH, NODEFORC,
+   AVERLPW, HALFCONW, CODEBEAM, CODEWINK, NIPSTUBI, NIPSTUBE,
+   NIPROLLE, NIPROLI, LEFTLEM)
*
*   DEFINE VARIABLES
*
INTEGER NMCROSEL, NMCROSND, DOFPERND, CURRELEM, INDEXNUM, LOCALCOL
INTEGER LOCALROW, LEFTLEM, CODEBEAM(NMCROSEL), CODEWINK(NMCROSEL)
INTEGER CURRNODE
DOUBLE PRECISION PI, RADEFF, WINKWIDT, ROW(4), ROW3
DOUBLE PRECISION NODEDISP(NMCROSND*DOFPERND*2), LPWVSND(NMCROSND)
DOUBLE PRECISION MAXPRESS(NMCROSND), ELLENGTH(NMCROSEL)
DOUBLE PRECISION HALFCONW(NMCROSND)
DOUBLE PRECISION NODELOAD[ALLOCATABLE](:), LENGTH1, LENGTH2
DOUBLE PRECISION NODEFORC, AVERLPW, BEAMESM(4,4)
DOUBLE PRECISION STIFF, NIPSTUBI, NIPSTUBE, NIPROLLE, NIPROLI
*
*   ALLOCATE VARIABLE DIMENSION ARRAY
*
ALLOCATE (NODELOAD(NMCROSND))
*
*
*   ROW3 = 0.00D+00
*
*   CALCULATE THE AVERAGE LOAD PER WIDTH
*
AVERLPW = NODEFORC/ WINKWIDT
*
*   OVERWRITE BEAM DISPLACEMENTS WITH WINKLER DIPLACEMENTS FOR THE
*   ELEMENTS WHERE THE WINKLER FOUNDATION EXISTS
*
DO 800 CURRELEM = 1, NMCROSEL
    INDEXNUM = CURRELEM*DOFPERND - 1
    IF (CODEWINK(CURRELEM).EQ.1) THEN
        CURWINEL = CURRELEM - LEFTLEM
        WINKNODE = DOFPERND*(NMCROSEL + CURWINEL) + 1
        NODEDISP(INDEXNUM) = NODEDISP(WINKNODE)
        NODEDISP(INDEXNUM+1) = NODEDISP(WINKNODE+1)
    ENDIF
800 CONTINUE
INDEXNUM = (NMCROSEL+1)*DOFPERND - 1
IF (CODEWINK(NMCROSEL).EQ.1) THEN
    CURWINEL = NMCROSEL + 1 - LEFTLEM
    WINKNODE = DOFPERND*(NMCROSEL + CURWINEL) + 1
    NODEDISP(INDEXNUM) = NODEDISP(WINKNODE)
    NODEDISP(INDEXNUM+1) = NODEDISP(WINKNODE+1)
ENDIF
*
*   CALCULATE LOAD PER WIDTH FOR EACH NODE
*
DO 810 CURRELEM = 1, NMCROSEL
    CURRNODE = CURRELEM
*

```

```

*      CALCULATE BEAM ELEMENT STIFFNESS
*
      IF (CODEBEAM(CURRELEM).EQ.1) THEN
          STIFF = NIPROLLE*NIPROLLI/ (ELLENGTH(CURRELEM)**3)
      ELSE
          STIFF = NIPSTUBE*NIPSTUBI/ (ELLENGTH(CURRELEM)**3)
      ENDIF
*
*      GET ELEMENT STIFFNESS MATRIX FOR BEAM
*
      CALL BEAMELEM(ELLENGTH(CURRELEM), STIFF, BEAMESM)
*
*      RESET ARRAY
*
      DO 820 LOCALCOL = 1, 4
          ROW(LOCALCOL) = 0.0D+00
820    CONTINUE
*
*      MULTIPLY BEAM ELEMENT STIFFNESS MATRIX AND LOCAL WINKLER NODE
*      DISPLACE VECTOR TO GET LOCAL NODE FORCES AND MOMENTS
*
      INDEXNUM = DOPPERND * CURRELEM - 1
      DO 830 LOCALROW = 1, 4
          DO 840 LOCALCOL = 1, 4
              ROW(LOCALROW) = ROW(LOCALROW) + BEAMESM(LOCALROW,
+                LOCALCOL) * NODEDISP(INDEXNUM -1 + LOCALCOL)
840          CONTINUE
830      CONTINUE
*
*      SUM LOADS PER NODE VALUES FROM THE ELEMENTS ON BOTH SIDES OF
*      EACH NODE
*
      NODELOAD(CURRNODE) = ABS(ROW(1)+ROW3)
      IF (CURRNODE.EQ.1) THEN
          NODELOAD(CURRNODE) = (NODEFORC/2.0D+00) -
+            NODELOAD(CURRNODE)
      ENDIF
      ROW3 = ROW(3)
      IF (CURRNODE.EQ.NMCROSEL) THEN
          NODELOAD(NMCROSND) = (NODEFORC/2.0D+00) - ROW3
      ENDIF
      IF (NODELOAD(CURRNODE).GT.1.25*NODEFORC) THEN
          NODELOAD(CURRNODE) = 0.0D+00
      ENDIF
810    CONTINUE
*
*      CALCULATE THE LOAD PER WIDTH VERSUS NODE
*
      DO 850 CURRNODE = 1, NMCROSND
          CURRELEM = CURRNODE
          IF (CURRNODE.EQ.1) THEN
              LENGTH1 = 0.0D+00
          ELSE
              LENGTH1 = ELLENGTH(CURRELEM-1)
          ENDIF
          IF (CURRNODE.EQ.NMCROSND) THEN
              LENGTH2 = 0.0D+00
          ELSE
              LENGTH2 = ELLENGTH(CURRELEM)
          ENDIF
*
          IF (CURRNODE.GT.1.AND.CURRNODE.LE.NMCROSEL) THEN

```

```

      IF (CODEWINK(CURRELEM-1).EQ.0.AND.CODEWINK(CURRELEM).EQ.1)
+       THEN
          LENGTH1 = 0.0D+00
      ENDIF
      IF (CODEWINK(CURRELEM-1).EQ.1.AND.CODEWINK(CURRELEM).EQ.0)
+       THEN
          LENGTH2 = 0.0D+00
      ENDIF
      ENDIF
*
      LPWVSNOD(CURRNODE) = NODELOAD(CURRNODE)/
+      (LENGTH1/2.0D+00 + LENGTH2/2.0+00)
850  CONTINUE
*
*   IF THE WINKLER FOUNDATION DOESN'T EXIST AT THE CURRENT NODE,
*   ELIMINATE THE DISPLACEMENTS THERE
*
DO 860 CURRNODE = 1, NMCROSND
      INDEXNUM = CURRNODE*DOFFPERND - 1
      CURRELEM = CURRNODE
      IF (CURRNODE.EQ.1.AND.CODEWINK(CURRELEM).EQ.0) THEN
          NODEDISP(INDEXNUM) = 0.0D+00
          NODEDISP(INDEXNUM+1) = 0.0D+00
      ELSE IF (CURRNODE.EQ.NMCROSND.AND.CODEWINK(CURRELEM-1).EQ.0)
+      THEN
          NODEDISP(INDEXNUM) = 0.0D+00
          NODEDISP(INDEXNUM+1) = 0.0D+00
      ELSE
          IF (CODEWINK(CURRELEM-1).EQ.0.AND.CODEWINK(CURRELEM).EQ.0)
+          THEN
              NODEDISP(INDEXNUM) = 0.0D+00
              NODEDISP(INDEXNUM+1) = 0.0D+00
          ENDIF
      ENDIF
860  CONTINUE
*
*   CALCULATE HALF WIDTH OF CONTACT CORRESPONDING TO NODE DISPLACEMENT
*   AND THE MAXIMUM PRESSURE
*
DO 870 CURRNODE = 1, NMCROSND
      INDEXNUM = CURRNODE*DOFFPERND - 1
      IF (NODEDISP(INDEXNUM).LT.1.0D-19) THEN
          HALFCONW(CURRNODE) = 0.0D+00
      ELSE
          HALFCONW(CURRNODE) = DSQRT(4.0D+00*NODEDISP(INDEXNUM)*
+          RADEFF)
      ENDIF
      IF (HALFCONW(CURRNODE).LT.1.0D-04) THEN
          MAXPRESS(CURRNODE) = 0.0D+00
      ELSE
          MAXPRESS(CURRNODE) = -2.0D+00*LPWVSNOD(CURRNODE)/
+          (PI*HALFCONW(CURRNODE))
      ENDIF
870  CONTINUE
*
*   DEALLOCATE VARIABLE DIMENSION ARRAYS
*
      DEALLOCATE (NODELOAD)
*
      RETURN TO MAIN PROGRAM
*

```

RETURN
END

APPENDIX B

EXAMPLE OUTPUT FILE: PINGELPW.OUT

This is the output file PINGELPW.OUT from the program IMPINGE written by:
Paul Hoffeecker at MAE Research Lab, Oklahoma State University.

Each given Hertzian Contact Half Width has a corresponding wound roll
deformation as shown. The wound roll foundation stiffness and load
per width at the deformation are also given.

The run date is: March 15, 1997

Units = eng

Wound Roll...

K1 = 13.537 (Lb/ in²) K2 = 23.550

K3 = .000 (Lb/ in²)

Radius = 5.000 (in) Poissons Ratio = .010

Nip Roller...

Radius = 2.000 (in) Poissons Ratio = .300

Modulus = 10300000.000 (Lb/ in²)

Overall...

Radius Eff = 1.429 (in)

Half Width of Contact (in)	Roll Deform (in)	Load per Width (Lb/ in)	Foundation Stiffness (Lb/ in ²)
.00000	.00000	.0000	1001.6002
.02500	.00011	.1098	1006.7699
.05000	.00044	.4427	1022.3991
.07500	.00098	1.0091	1048.8529
.10000	.00175	1.8265	1086.7537
.12500	.00273	2.9209	1137.0031
.15000	.00394	4.3270	1200.8134
.17500	.00536	6.0900	1279.7513
.20000	.00700	8.2677	1375.7947
.22500	.00886	10.9321	1491.4056
.25000	.01094	14.1734	1629.6236
.27500	.01323	18.1030	1794.1831
.30000	.01575	22.8589	1989.6611
.32500	.01848	28.6117	2221.6634
.35000	.02144	35.5726	2497.0579
.37500	.02461	44.0029	2824.2692
.40000	.02800	54.2269	3213.6491
.42500	.03161	66.6478	3677.9458
.45000	.03544	81.7680	4232.8975
.47500	.03948	100.2150	4897.9869

APPENDIX C

STANDARD INPUT FILE: DATASAMP.IN

```

March 15, 1997      DATE
eng                UNIT SYSTEM
2.0D+00            NIP ROLLER DIMENSIONS...NIP ROLLER RADIUS
3.6D+01            CMD DISTANCE BETWEEN APPLIED LOAD LOCATIONS
0.0D+00            CMD DISTANCE FROM ZERO TO NIP ROLLER
3.6D+01            NIP ROLLER CMD WIDTH
5.0D+00            WOUND ROLL DIMENSIONS...RADIUS OF WOUND ROLL
0.0D+00            CMD DISTANCE FROM ZERO TO THE WOUND ROLL
3.6D+01            CMD WIDTH OF THE WOUND ROLL
1.03D+07           NIP ROLL MATERIAL DATA...NIP ROLLER MODULUS OF ELASTICITY
5.200D+00          NIP ROLLER MOMENT OF INERTIA
0.3D+00            NIP ROLLER POISSONS RATIO
1.03D+07           NIP ROLLER STUB SHAFT MODULUS OF ELASTICITY
0.785D+00          NIP ROLLER STUB SHAFT MOMENT OF INERTIA
1.0D-02            WOUND ROLL MATERIAL DATA...WOUND ROLL POISSONS RATIO
1.3537D+01         K1
2.355D+01         K2
0.00D+00          K3
0.0D+00            GENERAL HERTZIAN CONTACT ANALYSIS...HALFWIDTH STARTING VALUE
.025D+00           HALFWIDTH INCREMENT SIZE
20                NUMBER OF HALFWIDTH INCREMENTS TO STEP THROUGH
2.00D+02           FINITE ELEMENT ANALYSIS...APPLIED NIP FORCE LOAD
2.5D+01            NIP FORCE STEP SIZE
0.00D+00           STARTING NIP FORCE
N                 DECISION: WANT OUTPUT FILE TO INCLUDE ROTATIONS?
N                 DECISION: SHOULD THE FEM ITERATE FOR DEFORMATIONS?
N                 IS NIP'S LEFT END CONSTRAINED FROM ROTATING?
N                 IS NIP'S RIGHT END CONSTRAINED FROM ROTATING?
36                NUMBER OF FINITE ELEMENTS ACROSS THE WOUND ROLL

```


APPENDIX D

EXAMPLE OUTPUT FILES: PINGESUM.OUT, PINGEPRO.OUT,
AND PINGEINC.OUT

This is the output file PINGESUM.OUT from the program IMPINGE written by:
Paul Hoffecker at MAE Research Lab, Oklahoma State University.

This summary covers info from the other files including: gapping,
and the deformation of the winkler foundation elements.
The run date is: March 15, 1997

```

Units          =          eng

Wound Roll...
K1              =          13.537 (Lb/ in^2)      K2              =          23.550
K3              =          .000 (Lb/ in^2)
Radius          =          5.000 (in)            Poissons Ratio =          .010
CMD Width      =          36.000 (in)            # of Elements  =          36

Stub Shaft...
Modulus         =          10300000.000 (Lb/ in^2)  MOI              =          .785
                                                    (in^4)

Nip Roller...
Radius          =          2.000 (in)            Poissons Ratio =          .300
Modulus         =          10300000.000 (Lb/ in^2)  MOI              =          5.200

CMD Width      =          36.000 (in)            (in^4)

Overall...
Total Force     =          200.000 (Lb)           Force Step Size =          25.0(Lb)

Start Force     =          .000 (Lb)             Iteration on?   =          N
CMD Width      =          36.000 (in)            # of Elements  =          36
Rigid left     =          N                     Rigid Right     =          N
Radius Eff     =          1.429 (in)

CMD distance from machine zero to...
Nip Roller     =          .000 (in)             Wound Roll      =          .000
                                                    (in)

```

Element Number	Element Left Node Location (in)	Nip Element Type	Element Displace (in)	Nip to Wound Roll Contact
1	.00	ROLLER	.005819	NO_GAP
2	1.00	ROLLER	.005633	NO_GAP
3	2.00	ROLLER	.005450	NO_GAP
4	3.00	ROLLER	.005272	NO_GAP
5	4.00	ROLLER	.005099	NO_GAP
6	5.00	ROLLER	.004933	NO_GAP
7	6.00	ROLLER	.004775	NO_GAP
8	7.00	ROLLER	.004627	NO_GAP
9	8.00	ROLLER	.004489	NO_GAP
10	9.00	ROLLER	.004363	NO_GAP
11	10.00	ROLLER	.004248	NO_GAP
12	11.00	ROLLER	.004146	NO_GAP
13	12.00	ROLLER	.004058	NO_GAP
14	13.00	ROLLER	.003984	NO_GAP
15	14.00	ROLLER	.003924	NO_GAP
16	15.00	ROLLER	.003879	NO_GAP
17	16.00	ROLLER	.003848	NO_GAP

18	17.00	ROLLER	.003833	NO_GAP
19	18.00	ROLLER	.003833	NO_GAP
20	19.00	ROLLER	.003848	NO_GAP
21	20.00	ROLLER	.003879	NO_GAP
22	21.00	ROLLER	.003924	NO_GAP
23	22.00	ROLLER	.003984	NO_GAP
24	23.00	ROLLER	.004058	NO_GAP
25	24.00	ROLLER	.004146	NO_GAP
26	25.00	ROLLER	.004248	NO_GAP
27	26.00	ROLLER	.004363	NO_GAP
28	27.00	ROLLER	.004489	NO_GAP
29	28.00	ROLLER	.004627	NO_GAP
30	29.00	ROLLER	.004775	NO_GAP
31	30.00	ROLLER	.004933	NO_GAP
32	31.00	ROLLER	.005099	NO_GAP
33	32.00	ROLLER	.005272	NO_GAP
34	33.00	ROLLER	.005450	NO_GAP
35	34.00	ROLLER	.005633	NO_GAP
36	35.00	ROLLER	.005819	NO_GAP

Node 37 ... 36.00 (in)

At a Nip Force of 200.000 (Lb) 0 gaps occurred across the width.
The gapping occurs in 0% of the elements.

This is the output file PINGEPRO.OUT from the program IMPINGE written by:
Paul Hoffercker at MAE Research Lab, Oklahoma State University.

The deformation profile includes the nodal displacement, load per width, max pressure, average load per width, and the corresponding halfwidth of contact across the wound roll at the max load.

The run date is: March 15, 1997

Units = eng

Wound Roll...

K1 = 13.537 (Lb/ in²) K2 = 23.550
K3 = .000 (Lb/ in²)
CMD Width = 36.000 (in) # of Elements = 36

Nip Roller...

Modulus = 10300000.000 (Lb/ in²) MOI = 5.200
CMD Width = 36.000 (in) (in⁴)

Overall...

Total Force = 200.000 (Lb) Force Step Size = 25.0(Lb)
Start Force = .000 (Lb) Iteration on? N
CMD Width = 36.000 (in) # of Elements = 36
Radius Eff = 1.429 (in)

Node Number	Node Displace (in)	Load Per Width (Lb/ in)	Maximum Pressure (Lb/ in ²)	Average Load/ W (Lb/ in)	Halfwidth of Contact (in)
1	.005912	7.4375	-25.7608	5.5556	.1838
2	.005726	7.2460	-25.5028	5.5556	.1809
3	.005541	6.9622	-24.9087	5.5556	.1779
4	.005360	6.6872	-24.3256	5.5556	.1750
5	.005184	6.4230	-23.7584	5.5556	.1721
6	.005014	6.1715	-23.2114	5.5556	.1693
7	.004852	5.9342	-22.6884	5.5556	.1665
8	.004699	5.7123	-22.1933	5.5556	.1639
9	.004555	5.5069	-21.7294	5.5556	.1613
10	.004423	5.3190	-21.2997	5.5556	.1590
11	.004302	5.1493	-20.9073	5.5556	.1568
12	.004194	4.9983	-20.5545	5.5556	.1548
13	.004099	4.8666	-20.2438	5.5556	.1530
14	.004017	4.7546	-19.9773	5.5556	.1515
15	.003950	4.6626	-19.7567	5.5556	.1502
16	.003897	4.5908	-19.5835	5.5556	.1492
17	.003860	4.5393	-19.4590	5.5556	.1485
18	.003837	4.5085	-19.3839	5.5556	.1481
19	.003829	4.4982	-19.3588	5.5556	.1479
20	.003837	4.5085	-19.3839	5.5556	.1481
21	.003860	4.5393	-19.4590	5.5556	.1485
22	.003897	4.5907	-19.5835	5.5556	.1492
23	.003950	4.6626	-19.7567	5.5556	.1502
24	.004017	4.7546	-19.9773	5.5556	.1515
25	.004099	4.8666	-20.2438	5.5556	.1530
26	.004194	4.9983	-20.5545	5.5556	.1548

27	.004302	5.1493	-20.9073	5.5556	.1568
28	.004423	5.3190	-21.2997	5.5556	.1590
29	.004555	5.5069	-21.7294	5.5556	.1613
30	.004699	5.7123	-22.1933	5.5556	.1639
31	.004852	5.9342	-22.6884	5.5556	.1665
32	.005014	6.1715	-23.2114	5.5556	.1693
33	.005184	6.4230	-23.7584	5.5556	.1721
34	.005360	6.6872	-24.3256	5.5556	.1750
35	.005541	6.9622	-24.9087	5.5556	.1779
36	.005726	7.2460	-25.5028	5.5556	.1809
37	.005912	7.4375	-25.7608	5.5556	.1838

APPENDIX E

EXAMPLE OUTPUT FILE: PINGEDEF.OUT

This is the output file PINGEDEF.OUT from the program IMPINGE written by:
Paul Hoffecker at MAE Research Lab, Oklahoma State University.

The given beam and winkler deformations and angles are for each
increasing Nip load increment, up to the desired Nip force.
The run date is: March 15, 1997

```
Units          =          eng

Wound Roll...
CMD Width      =          36.000 (in)      # of Elements =          36

Stub Shaft...
Modulus        = 10300000.000 (Lb/ in^2)  MOI              =          .785
                                                    (in^4)

Nip Roller...
Modulus        = 10300000.000 (Lb/ in^2)  MOI              =          5.200
CMD Width      =          36.000 (in)      (in^4)

Overall...
Total Force    =          200.000 (Lb)      Force Step Size =          25. (Lb)

Start Force    =          .000 (Lb)         Iteration on?    =          N
CMD Width      =          36.000 (in)      # of Elements    =          36
```

	BEAM	WINK	BEAM	WINK	BEAM	WINK
Force, (Lb)	.00	.00	25.00	25.00	50.00	50.00
U 1 (in)	.00000	.00000	.00086	.00086	.00168	.00168
U 2 (in)	.00000	.00000	.00084	.00084	.00164	.00164
U 3 (in)	.00000	.00000	.00082	.00082	.00159	.00159
U 4 (in)	.00000	.00000	.00079	.00079	.00154	.00154
U 5 (in)	.00000	.00000	.00077	.00077	.00150	.00150
U 6 (in)	.00000	.00000	.00075	.00075	.00145	.00145
U 7 (in)	.00000	.00000	.00073	.00073	.00141	.00141
U 8 (in)	.00000	.00000	.00071	.00071	.00137	.00137
U 9 (in)	.00000	.00000	.00069	.00069	.00134	.00134
U10 (in)	.00000	.00000	.00067	.00067	.00130	.00130
U11 (in)	.00000	.00000	.00066	.00066	.00127	.00127
U12 (in)	.00000	.00000	.00064	.00064	.00124	.00124
U13 (in)	.00000	.00000	.00063	.00063	.00122	.00122
U14 (in)	.00000	.00000	.00062	.00062	.00120	.00120
U15 (in)	.00000	.00000	.00061	.00061	.00118	.00118
U16 (in)	.00000	.00000	.00061	.00061	.00117	.00117
U17 (in)	.00000	.00000	.00060	.00060	.00116	.00116
U18 (in)	.00000	.00000	.00060	.00060	.00115	.00115
U19 (in)	.00000	.00000	.00060	.00060	.00115	.00115
U20 (in)	.00000	.00000	.00060	.00060	.00115	.00115
U21 (in)	.00000	.00000	.00060	.00060	.00116	.00116
U22 (in)	.00000	.00000	.00061	.00061	.00117	.00117
U23 (in)	.00000	.00000	.00061	.00061	.00118	.00118
U24 (in)	.00000	.00000	.00062	.00062	.00120	.00120
U25 (in)	.00000	.00000	.00063	.00063	.00122	.00122
U26 (in)	.00000	.00000	.00064	.00064	.00124	.00124
U27 (in)	.00000	.00000	.00066	.00066	.00127	.00127
U28 (in)	.00000	.00000	.00067	.00067	.00130	.00130
U29 (in)	.00000	.00000	.00069	.00069	.00134	.00134

U30 (in)	.00000	.00000	.00071	.00071	.00137	.00137
U31 (in)	.00000	.00000	.00073	.00073	.00141	.00141
U32 (in)	.00000	.00000	.00075	.00075	.00145	.00145
U33 (in)	.00000	.00000	.00077	.00077	.00150	.00150
U34 (in)	.00000	.00000	.00079	.00079	.00154	.00154
U35 (in)	.00000	.00000	.00082	.00082	.00159	.00159
U36 (in)	.00000	.00000	.00084	.00084	.00164	.00164
U37 (in)	.00000	.00000	.00086	.00086	.00168	.00168

	BEAM	WINK	BEAM	WINK	BEAM	WINK
Force, (Lb)	75.00	75.00	100.00	100.00	125.00	125.00

U 1 (in)	.00246	.00246	.00321	.00321	.00392	.00392
U 2 (in)	.00239	.00239	.00311	.00311	.00380	.00380
U 3 (in)	.00232	.00232	.00302	.00302	.00369	.00369
U 4 (in)	.00225	.00225	.00293	.00293	.00357	.00357
U 5 (in)	.00218	.00218	.00284	.00284	.00346	.00346
U 6 (in)	.00212	.00212	.00275	.00275	.00335	.00335
U 7 (in)	.00206	.00206	.00267	.00267	.00325	.00325
U 8 (in)	.00200	.00200	.00259	.00259	.00315	.00315
U 9 (in)	.00194	.00194	.00252	.00252	.00306	.00306
U10 (in)	.00189	.00189	.00245	.00245	.00298	.00298
U11 (in)	.00185	.00185	.00239	.00239	.00290	.00290
U12 (in)	.00181	.00181	.00234	.00234	.00284	.00284
U13 (in)	.00177	.00177	.00229	.00229	.00278	.00278
U14 (in)	.00174	.00174	.00225	.00225	.00272	.00272
U15 (in)	.00171	.00171	.00221	.00221	.00268	.00268
U16 (in)	.00169	.00169	.00218	.00218	.00265	.00265
U17 (in)	.00168	.00168	.00217	.00217	.00262	.00262
U18 (in)	.00167	.00167	.00215	.00215	.00261	.00261
U19 (in)	.00167	.00167	.00215	.00215	.00260	.00260
U20 (in)	.00167	.00167	.00215	.00215	.00261	.00261
U21 (in)	.00168	.00168	.00217	.00217	.00262	.00262
U22 (in)	.00169	.00169	.00218	.00218	.00265	.00265
U23 (in)	.00171	.00171	.00221	.00221	.00268	.00268
U24 (in)	.00174	.00174	.00225	.00225	.00272	.00272
U25 (in)	.00177	.00177	.00229	.00229	.00278	.00278
U26 (in)	.00181	.00181	.00234	.00234	.00284	.00284
U27 (in)	.00185	.00185	.00239	.00239	.00290	.00290
U28 (in)	.00189	.00189	.00245	.00245	.00298	.00298
U29 (in)	.00194	.00194	.00252	.00252	.00306	.00306
U30 (in)	.00200	.00200	.00259	.00259	.00315	.00315
U31 (in)	.00206	.00206	.00267	.00267	.00325	.00325
U32 (in)	.00212	.00212	.00275	.00275	.00335	.00335
U33 (in)	.00218	.00218	.00284	.00284	.00346	.00346
U34 (in)	.00225	.00225	.00293	.00293	.00357	.00357
U35 (in)	.00232	.00232	.00302	.00302	.00369	.00369
U36 (in)	.00239	.00239	.00311	.00311	.00380	.00380
U37 (in)	.00246	.00246	.00321	.00321	.00392	.00392

	BEAM	WINK	BEAM	WINK	BEAM	WINK
Force, (Lb)	150.00	150.00	175.00	175.00	200.00	200.00

U 1 (in)	.00461	.00461	.00527	.00527	.00591	.00591
U 2 (in)	.00447	.00447	.00511	.00511	.00573	.00573
U 3 (in)	.00433	.00433	.00494	.00494	.00554	.00554
U 4 (in)	.00419	.00419	.00479	.00479	.00536	.00536
U 5 (in)	.00406	.00406	.00463	.00463	.00518	.00518
U 6 (in)	.00393	.00393	.00448	.00448	.00501	.00501
U 7 (in)	.00381	.00381	.00434	.00434	.00485	.00485

U 8 (in)	.00369	.00369	.00421	.00421	.00470	.00470
U 9 (in)	.00358	.00358	.00408	.00408	.00456	.00456
U10 (in)	.00348	.00348	.00396	.00396	.00442	.00442
U11 (in)	.00339	.00339	.00386	.00386	.00430	.00430
U12 (in)	.00331	.00331	.00376	.00376	.00419	.00419
U13 (in)	.00324	.00324	.00368	.00368	.00410	.00410
U14 (in)	.00318	.00318	.00361	.00361	.00402	.00402
U15 (in)	.00313	.00313	.00355	.00355	.00395	.00395
U16 (in)	.00309	.00309	.00350	.00350	.00390	.00390
U17 (in)	.00306	.00306	.00347	.00347	.00386	.00386
U18 (in)	.00304	.00304	.00345	.00345	.00384	.00384
U19 (in)	.00303	.00303	.00344	.00344	.00383	.00383
U20 (in)	.00304	.00304	.00345	.00345	.00384	.00384
U21 (in)	.00306	.00306	.00347	.00347	.00386	.00386
U22 (in)	.00309	.00309	.00350	.00350	.00390	.00390
U23 (in)	.00313	.00313	.00355	.00355	.00395	.00395
U24 (in)	.00318	.00318	.00361	.00361	.00402	.00402
U25 (in)	.00324	.00324	.00368	.00368	.00410	.00410
U26 (in)	.00331	.00331	.00376	.00376	.00419	.00419
U27 (in)	.00339	.00339	.00386	.00386	.00430	.00430
U28 (in)	.00348	.00348	.00396	.00396	.00442	.00442
U29 (in)	.00358	.00358	.00408	.00408	.00456	.00456
U30 (in)	.00369	.00369	.00421	.00421	.00470	.00470
U31 (in)	.00381	.00381	.00434	.00434	.00485	.00485
U32 (in)	.00393	.00393	.00448	.00448	.00501	.00501
U33 (in)	.00406	.00406	.00463	.00463	.00518	.00518
U34 (in)	.00419	.00419	.00479	.00479	.00536	.00536
U35 (in)	.00433	.00433	.00494	.00494	.00554	.00554
U36 (in)	.00447	.00447	.00511	.00511	.00573	.00573
U37 (in)	.00461	.00461	.00527	.00527	.00591	.00591



VITA

Paul Hoffecker

Candidate for the Degree of

Master of Science

Thesis: CHARACTERIZATION OF A NIP IMPINGED WOUND ROLL

Major Field of Study: Mechanical Engineering

Biographical:

Education: Graduated from Berthoud High School, Berthoud, Colorado in May 1989; received Bachelor of Science degree in Aerospace Engineering from University of Colorado at Boulder, Boulder, Colorado in May 1994. Completed the requirements for the Master of Science degree with a major in Mechanical and Aerospace Engineering at Oklahoma State University in May, 1997.

Experience: Interned and Contracted as space science researcher for the National Air and Space Museum, Smithsonian, May 1992- August 1992, and May 1993- August 1993 respectively. Employed with Colorado Space Grant Consortium (September, 1993 -January, 1996) in numerous capacities from design team member to lead of sounding rocket structure team. Also shared structures team lead responsibilities on NASA student satellite proposal. Employed by Oklahoma State University, Department of Mechanical and Aerospace Engineering as a graduate research assistant, 1996 to present.

Memberships: American Institute of Aeronautics and Astronautics