NUMERICAL METHODS FOR PREDICTING

ROLL PRESS POWDER COMPACTION PARAMETERS

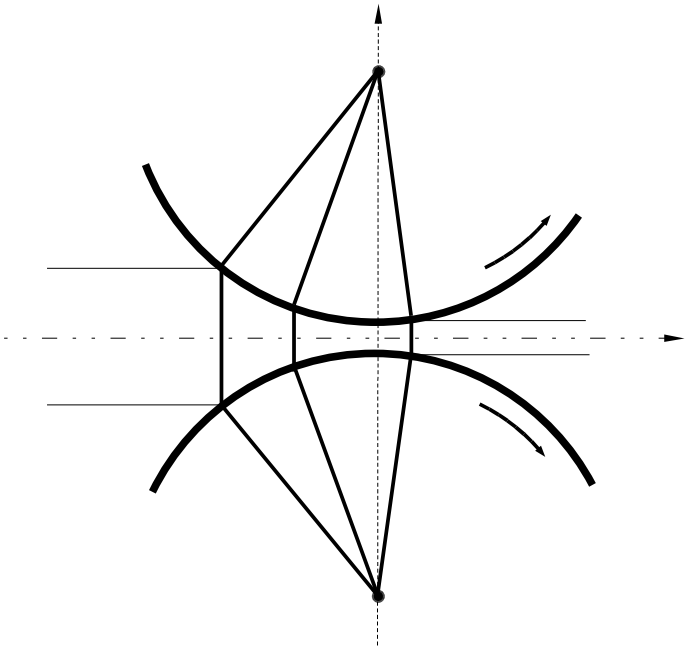
Marcin BALICKI

Adviser: Abder MICHRAFY

Ecole Des Mines D'Albi- Carmaux

France

July 15, 2003



Centre Poudres et Procédés
École des Mines d'Albi-Carmaux
Campus Jarlard - 81013 Albi
CT Cédex 09, France.
Téléphone : 05 63 49 31 22
E-mail : poudres@enstimac.fr

NUMERICAL METHODS FOR PREDICTING

ROLL PRESS POWDER COMPACTION PARAMETERS

Abstract

Large numbers of cohesive powders are compacted using a roll press. The principle of compaction is that particulate materials become a compact when they are subjected to high pressure.

Experimental work was carried out on pharmaceutical powders [1] to determine the influence of operating parameters on the roll process compaction. Theses parameters are determined for a given powder and must be measured again for any other powder. The subject of this work is to understand how to optimize the process parameters versus the mechanical behavior of powder and the roll-wall friction.

Rolling theory for granular solids has been continuously investigated for more then 40 years without a successful model representing the phenomenon. In this work, overall rolling compaction of powders is discussed followed by study of three different theoretical approaches. First the Johanson model is fully studied with the use of Matlab. Overview of the other two methods: slab method and ALE finite element method was also included. A comparison between these models is done to determine the most applicable method for further study.

Keywords: Powder Roll Compaction, mechanical behavior, friction, finite element method;

Table of Contents

1 Introduction

Roll compaction of granular solids is an industrial process used to compact a powder to a product of high homogeneous density and strength. At the end of the 19th century roll press compaction was initially developed to produce coal briquettes, but today the process has been widely adopted by mass production industries such as food processing, metallurgical, chemical, ceramic, semi-conductor as well as pharmaceutical and even more recently waste recycling [2]. The main principle behind roll press compaction is that compressible granular solids become compact when they are exposed to high stresses from applied pressure between two rolls rotating in the opposite directions.

Fig. 1.1 Pharmaceutical Roll Press (ALEXANDERWERK AG)[27]

Fig. 1.2 Chemical Roll Press (ALEXANDERWERK AG) [27]

The advantage of roll-type press application over classical (die) compaction is that high pressure is continuously exerted on moving granular solids providing an economical, high volume production of granules [3]. The cost savings come in energy consumption which is only limited to the power to drive the rolls, feeding system and any hydraulic adjustment mechanisms. Drying costs are usually minimal. Generally a roller press can produce tablets five times faster than die compaction press [4]. And in case of heavy industries such as mineral and fertilizer producers several hundred tons of material per hour can be processed [5]. However, with increase in speed comes a reduction of quality, thus making rolling compaction most suitable for products of low unit value, less accurate weight uniformity and high tolerances for compact imperfections.

Currently, there exists a lack of understanding of roll press powder compaction technology due to the complexity of the system which results in large scale use of empirical knowledge rather than scientific theory. While it is possible to achieve optimum performance using these trial-and-error techniques, the operating costs and time become a big factor especially within pharmaceutical industry where the materials value and desired quality are high. Moreover, the use of basic physical data, such as compressibility and compactibility (ability to cohere into compacts), in formulation work in order to predict

compacting behavior of particle matter is limited. An improved theoretical understanding of powders and the rolling compaction process will enable a more rational approach to the creation of powder compacts.

The driving force behind research in this field is the desire to formulate methods to help the operator of rolling compaction to correctly adopt his system to a specific powder and final product guidelines. Attempts to model the behavior of the system have been moderately successful. Most of the original work on roll press powder compaction has been adopted from sheet metal rolling but these cannot simply be transferred to roller mills especially not to briquetting rollers. There exist three models that are regarded as best suited for rolling compaction of powder: Johanson model [6-8], Slab method [8-11], and Finite Element Analysis (FEA) [8,12].

The Johanson is one of the first to attempt to describe the rolling compaction of powders. His work serves as the basis for beginning research in this field. Moreover, it can provide essential boundary conditions for other models. The slab method was originally developed and widely applied to successfully predict the pressure distribution and roll separating force in sheet metal rolling, then it was applied to metal powder rolling by Katashinskii [9]. The slab method seems to more completely describe the behavior of the system in compaction because it allows for inclusions of elements such as cohesion, friction, roll velocity; as well as their behavior as the powder is compacted. The FEA approach is the most recent modeling technique used to represent the deformation process in compaction of powders [13,14]. ABAQUS is one of the widely used and commercially available finite element analysis packages which provides the means to represent the behavior of powders. This approach is promising as it allows for inclusion of many powder parameters but, as with the slab method, this is where the difficulties arise. All the above require numerical processing with the possible exception of the slab method. The slab method and FEA methods for powder roll pressing are still in development stages paralleling the discoveries in powder characterization and process understanding.

Currently many efforts are placed on collecting experimental roll press data and characterizations of powders and compacts, which are both necessary for evaluating the quality of theoretical results [5,10] The general problem is that characteristics of particulate material are not well definable, difficult to measure and moreover they change as the powder undergoes compaction. Characterization of the flowability of a particular material consists of the relationship between shear strength and the compacting stress acting on it. This information is attained by experiment using a Jenkie shear tester or an annular shear cell tester [15]. This shear strength is dependent on bulk density which is determined by the compaction force to which the powder has been subjected. However, the applied stress in rolling compaction is not on the same order as in the available tester, therefore their measurement may just be applicable at low compaction pressures. This method also provides cohesion, which is related to the shear stress required to cause failure of the compacted to material when is subject to zero normal force. Another property is the wall friction describing the interaction between the particulate material and the surface it is in contact with, This property can also be measured by modifying the above instruments. Although the above tests attempt to measure the tribological properties, friction is still one of the major unresolved issues in representation and influence in powders especially while undergoing compaction. It is believed that friction affects, and is not constant throughout, densification which poses difficulties in

postulating a system model. Furthermore, the powder can also be characterized by the yield stress limit function such as Mohr-Coulomb yield criterion which describes the relationship between pressure and failure or shearing of the powder. Finally there should also be a assessment of the final granulate or compacted product either by physio-chemical or physio-mechanical means [16].

This paper is an initial study giving an overview of rolling compaction, followed by an extensive study of the Johanson method and a brief overview of the Slab method and FEA approach. It summarizes existing analysis to simulate and predict system behavior to gain understanding of issues involved to choose the best approach available to proceed.

2 Rolling Compaction

2.1 System Overview

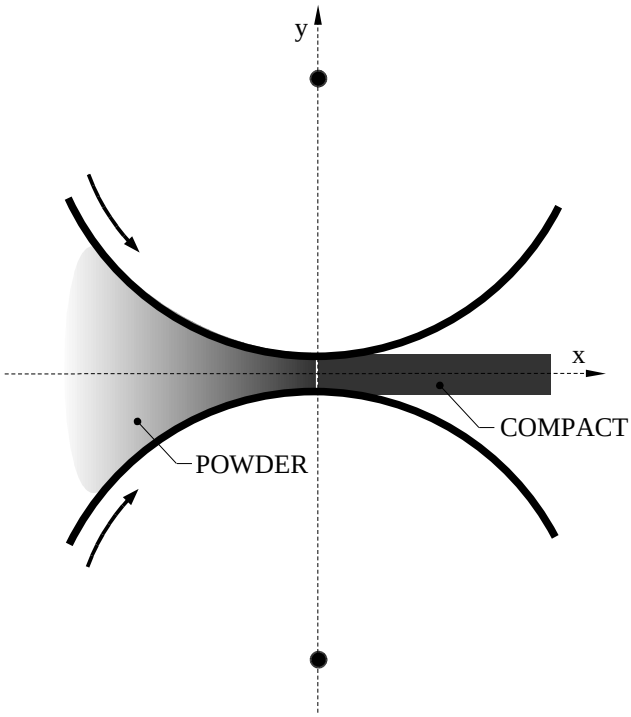


Fig. 2.1Roll Press

The roll compaction process consists of two counter rotating cylindrical rolls mounted so that their axes of rotation are parallel (Fig. 2.1). The raw material is delivered to the space between the rolls by gravity if the a hopper is used, and via a screw feeder or a combination of both [17]. Friction between material and the roller surface draws in the powder towards the narrow space separating the rolls (roll gap) where the powder is submitted to high stresses causing compaction. Depending on the roll surface materials can be shaped into dense sheets using smooth rolls, molded into strips using corrugated rolls or formed into a particular geometry matching the cavities recessed in the roll surface (briquettes) using pocketed rolls.

Feeding systems are the initial steps of the compaction process with great influence on the compaction results. The powder needs to sufficiently fill the space between the rolls in a uniform and continues fashion to yield a homogeneous compact. Fine powders exhibit poor flowability therefore, generally, screw feeders are used for that they allow greater control of feed pressure than conventional top fed system which is dependent on gravity and cohesive property of powders. Top feed systems, hoppers, tend to work best with non cohesive powders with high flowability.

Besides the feed system type, generally, a roll press can be categorized into two different types: type one that includes rollers that can be rigidly fixed to a particular height and type two that allows the adjustment of roller force. However, by adjusting the gap size in roll press of type one, the pressure exerted on the powder by the roll can be controlled. Conversely, the roll pressure adjustment roller type indirectly adjust the gap size which results from the equilibrium forces between the powder and the force of the rolls. As for

the width of the roller the diameter/width ratio is determined to minimize density variation across the roller.

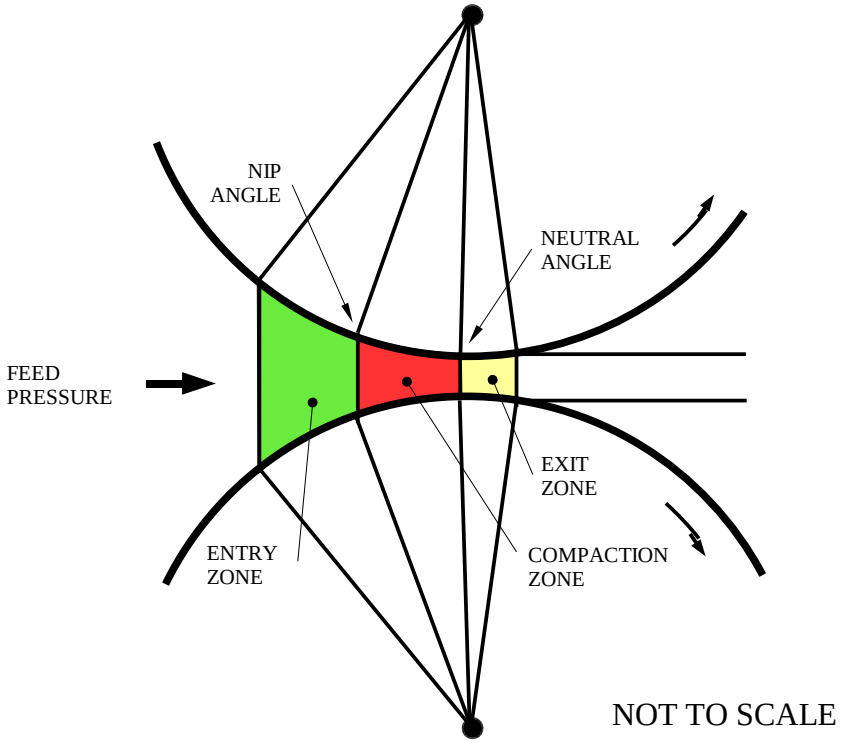


Fig. 2.2 Zones of process.

The compaction process can be divided into three zones defined by the behavior of the material (Fig. 2.2). In the first zone, feeding or entry zone, the densification is solely due to the rearrangement of particles under relatively small stresses created by the feeding method (ie. screw feeder). In the second zone, compaction zone, particles fracture and/or deform plasticly under heavy stress provided by the rolls. This region is believed to began at a point called the nip angle where the powder no longer slips and begins to stick to the wall. As a result the compaction zone is also referred to as the Nip Region. The final zone, the exit zone, is a region of a great decrease in pressure as the compact is ejected and can expand due to elasticity. As the material is pushed out of the process zone, it picks up speed and begins to move faster than the roller. This increase in speed causes slip in the opposite direction before product finally loses contact with the roller. The beginning of the ejection region is sometimes referred to as a neutral point because it sets the boundary between the region where the the material moves at the same speed as wall surface it comes in contact with and the region where the material moves faster than the roll. This angle is generally observed not to coincide with the point where the gap is the smallest but rather just prior to it due to minor slip of compact upon ejection [10]. Moreover, the natural point is also believed to be the point of maximum pressure.

It is important to mention the effects of friction on the system since it is the principal mechanism by which the powder is pulled into the roll gap. The friction forces acting on the strip are greater in the region where the roll moves faster that the material. The difference between the frictional forces in the two regions produces a net frictional force that pulls the powder into the roll gap. Generally, if the wall friction coefficient is too low, the material cannot be drawn through the roll press.

At first sight the rolling compaction may look simple by after all the parameters involved are analyzed and the powder behavior is discussed the problem contains many hurdles. The following powder roll compaction parameters can be considered in developing models and can be divided into the following categories:

Operating Parameters:

Roll Force, Roll Torque, Roll Velocity, Feed Pressure, Gravity, Inertia

Geometric Parameters:

Roll Diameter, Roll Width, Gap Size

Powder Parameters:

Internal (effective) Angle of Friction, Cohesion, Admissible Stress, Compressibility, Bulk density

Tribological Parameter:

Friction between powder and roll surface

Powder behavior characterization is one of the major research areas in pharmaceutical powders. With out good mathematical models for behavior of granular solids it is very difficult to create correct models of systems that include powders. As mentioned previously there exist a number of testing techniques for characterization or powders such as Jenike shear tester. Classical die compaction can also provide some information, especially for compact classification as well as some insight into the wall friction effects.

2.2 Desired Solution

The use of rollers continuously applies high pressure to a material efficiently producing sheets that can be further processed for storage, granulation, or final product. The process is used to alter material density, uniformity, flowability, and strength. For example, to lower cost of handling and transportation the bulk powder density can be increased as well as to minimize dust (excess 'fines') problems and improve handling [18]. It can also enhance performance characteristics of powder by reducing reactivity, solubility or permeability (ie. dosing - drug release time) [3]. With the use of briquetting roll press a required geometry can be achieved. In thermal operations, it can also improve efficiency of melting, drying or burning. A strip of compacted material can also be crushed to facilitate further processing in order to produce granules. Compaction can minimize segregative tendencies and prevent caking during storage. The advantages of agglomeration in pharmaceutical industry is that it is a dry granulation system with high volume production of granules and good control of final particle bulk density and flow properties [3].

As mentioned above, there exist many advantages of rolling compaction of powders, however, there exist very little scientific methods for designing a successful press by matching the process parameters to the granular solids to be compacted to get desired results. The number of variables, lack of good mathematical models and experimental data greatly effect the design of roller press. Currently, trail-and-error techniques are widely used to design and calibrate such systems. Thus, when a new powder is introduced to the roll-press operations many working parameters will change which will require many hours and materials to find these god compaction settings. For

example, one side effect of incorrect setting is delamination or capping caused by overcompaction. Scaling up pharmaceutical roll compaction process from laboratory to industrial magnitude can also involve several issues and technologies which will require more testing and funds. [3] More economical and simpler approach is to predict relationships between the bulk material, the press dimensions and the operating conditions by theoretical correlations, which are based on mathematical models and experimental measurements. For example, a quick approximation using a computer simulation can be useful to a predict operating parameters for preliminary trials which should be close to the optimal conditions, requiring only minor trimming by an experienced operator.

Optimal model provides relationships between the material properties, the press dimensions and operating parameters necessary to apply required pressure to a granular material to achieve a desired compact density [6,8]. This applied pressure can than be used to calculate the density or the strength of compact based on experimental data (ie. indentation tests).

Desired relation for pressure is a function of the following types of parameters:

$$PRESSURE = f(PROCESS, GEOMETRIC, MATERIAL, TRIBOLOGIC)$$

2.3 Approach

In modeling, one attempts to exclude trivial detail and include all essential features so the model describes the actual problem with sufficient accuracy and is not unnecessarily complicated. This mathematical model is an idealization, in which geometry, material properties, loads, and boundary conditions are simplified based on the analyst's understanding of what features are important and not important in obtaining the desired results. It is necessary to understand that analytical method is applied to this mathematical model rather than to the actual physical problem. When the physical phenomenon is fully understood, only then one can describe or approximate the system by selected differential equations and boundary conditions. Analytical approach is highly desired due to an immediate ability of such model to be applied in the field by roll compactor operators, however the task of finding analytical expressions relating all parameters of the process is quite difficult.

Oversimplified models tend to give us poor information, but are good starting points in understanding the problem. The first area of study was the application of sheet metal rolling theories [21,22]. Since these do not include the same issues and assumptions when powders are used (mainly material yield criterion), compaction of metal powders is the area of more interest as investigated by Tundermann or Katashinskii [9,19-22] A major breakthrough was done by Johanson who postulated a method to predict the powder behavior in the feeding zone as well as the pressure distribution in the nip region [6]. He also provided the means to calculate the boundary point between feeding and nip regions as well as roll force and roll torque. His work even incorporates application of pocketed rolls. Since this paper is an initial investigation into rolling compaction of powders the this model was used to gain more understanding of the posed issues as well as gain a general knowledge about the parameters and their effects. But as presented, this model is not the most accurate representation of the process, so other models were briefly looked at.

The slab method seems very promising because it can incorporate ample

information about the behavior of powders, much more than the Johanson model. It also has potential to provide functions which can be used without the help of computers to rapidly calculate the working parameters. This approach was original applied to sheet metal and metal powder rolling. At its core is the analysis of the nip region a small slice at a time. Just as the Johanson model the slab method only works in one plane assuming there is no out of plane strain. Force equilibrium equations are written for this small slab, including velocity, gravity, friction, and other factors. The result is a differential equation that is rewritten in terms of one principal stress by relating and simplifying the two existing principal stresses using a yield criterion which describes the behavior of powder under extreme stress that causes it to shear. The complexity of this function seems to increase with it's ability to correctly represent yield limit and therefore making the differential equation very difficult to solve without significant simplification or numerical methods.

The most modern approach, finite element analysis, extensively relies on computer's ability to perform quick and numerous calculations. It is applied with mathematical model of system behavior and experimental data for powder behavior. ABAQUS, a commercial FEA software, has recently been used to simulate classical die compaction with satisfactory results [13]. Rolling compaction seems more challenging due to more complex boundary conditions, however a group of scientists has developed FEM code for ABAQUS that simulates the problem with moderate success [8]. The advantage of using this approach is that the models can be adjusted to produce improved solutions through numerical testing and reformulation. Even though the computers today are powerful, to expend the model to three-dimensions would take much computing time but not necessarily yield more insight into the process. The FEM modeling technique appears to be even more interesting than the Slab method due its reputation as one of the closest real world modeling tools and its ability to incorporate any user defined parameter behavior with relative ease.

Noteworthy is the fact that friction is believed to play a major role in the process. All these models use a somewhat simple Mohr-Coulomb friction assumption which seems to be true for solid materials but is not verified for powders. Furthermore, great emphasis needs to be placed on the study of powder characterization, since this is central to the ability of all the theories discussed to correctly model powders behavior in a compaction environment whether it is in classical compaction or rolling compaction.

Final step would be to expend the chosen model to include material behavior models in the feeding process, exact pocket geometries in the roll surface and expand the model to three-dimensions. As these models are developed there needs to be parallel development in process parameter measurement for theory verification [10].

3 J.R. Johanson Model Study

3.1 Introduction

Although roll press compaction was predominantly applied in metal sheet or bar rolling, in the early 1960s a new emphasis was placed on rolling of powder metals or briquetting granular materials. At that time, information about the design of a roll press for powders was only of empirical nature. Thus, there existed a need for a mathematical model describing the relationships between the material properties, the press dimensions and operating parameters to aid engineers and operators in design and operation of roll presses for compaction of granular materials. Johanson was one of the first to fill this void by providing the means to determine the press dimensions and roll forces necessary to apply the required pressure to a material with specific properties which were attained experimentally [6].

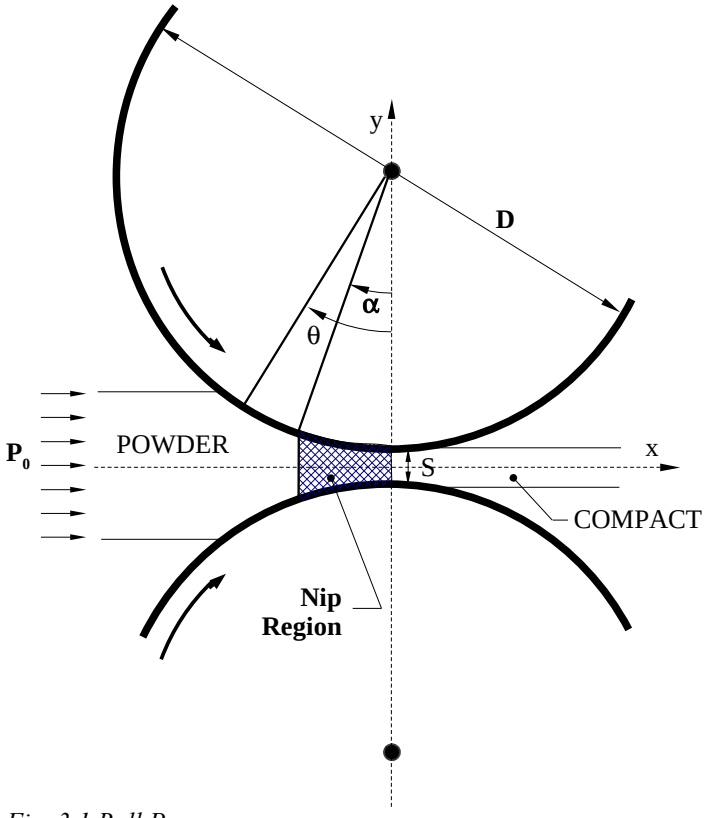


Fig. 3.1 Roll Press

Johanson model describes the stress function in relation to geometric parameters and the plastic yield laws of the material undergoing continuous shear deformation between rolls. His theory is based around the fact that the process can be divided into distinct zones where the powder behaves in a unique and determinable fashion. These regions are as follows:

1. Initially the powder is pushed towards the region between the two rolls with a constant pressure by the use of a screw feeder or gravity-fed hopper. Here the powder has the bulk material properties and does not change in density due to gravity.
2. As the powder moves into the gap, it begins to press against the wall of the roll but

not with enough pressure to prevent slipping. In this zone, also called the slip region, the powder experiences relatively little shear deformation.

3. At some point along the surface of the roll, the powder stops slipping and starts to travel with the same velocity as the roller. Johanson calls this point Nip Angle, α.
4. The nip angle marks the beginning point of the compaction zone, or the nip region (Fig. 3.1). Here the powder becomes drawn in between the two rolls which exert pressure on the powder forcing it to undergo continuous shear deformation until it is compacted to the thickness equal to the smallest part of the region between the rolls, the gap size. In this region the powder is assumed to "stick" to the roll surface or have the same velocity as the roller.

Following the compaction the material enters the exit region where its velocity is greater than that of the rollers. At this point the compact is believed to expand, although Johanson notes that this expansion is quite small and assumed to be negligible.

Johanson model can be used to calculate the maximum pressure exerted on the material being compacted. This pressure peak occurs at the smallest part of the gap ($\theta = 0$). Coupling this information with experimental pressure-density relationship, the final density of the compact can be predicted. The model also provides means for calculating roll force and torque.

In his equations, Johanson includes an approximation that describes rolls with pockets or indentations; However, for the purpose of this paper, the roll surfaces are assumed to be pocketless. Following list displays assumptions used in the work of Johanson.

- *Material is isotropic, frictional and cohesive.*
- *Material undergoes continues shear deformation into a solid mass.*
- *Rollers are rigid.*
- *Circumferences of the rollers are much larger then the contact area.*
- *Continuous plain-strain deformation.*
- *Friction between powder and roll surface is Coulomb friction.*
- *Weight of material is negligible when the press is force-fed.*

3.2 Pressure Distribution before the Nip Region

Johanson introduced Jenike's model for steady state flow in the theory of roller press compaction. The Jenike and Shield effective yield criterion *(Eq. 3.1*) applies to a region between the rolls where the powder slips against the roll surface. The yield behavior of the powder is described by the internal friction angle δ.

Eq. 3.1 $$\sin(\delta) = \frac{\sigma_1 - \sigma_2}{\sigma_1 + \sigma_2}$$

This region begins where the powder initially makes contact with roll surface, θ_0, at which point the feed pressure P_0 is the dominant source of stress (Eq. 3.2) . Notably,

one can infer that the mean normal stress ($\sigma = (\sigma_1 + \sigma_2)/2$ for any θ) remains constant for considerable distance after the application of Po, θ_0. Moreover, it increases to an extremely large value as $\theta = 0$ is approached.

Eq. 3.2 $$\sigma_0 = \frac{P_0}{(1 - \sin\delta)}$$

The feed angle θ_0 can be computed using the wall friction angle ϕ and the internal or effective friction angle δ(Eq. 3.3).

Eq. 3.3 $$\theta_0 = \frac{\phi + \arcsin\left(\frac{\sin\phi}{\sin\delta}\right)}{2}$$

The Jenike-Shield yield criterion (effective yield locus) for the slip region is represented graphically in Fig. 3.2.

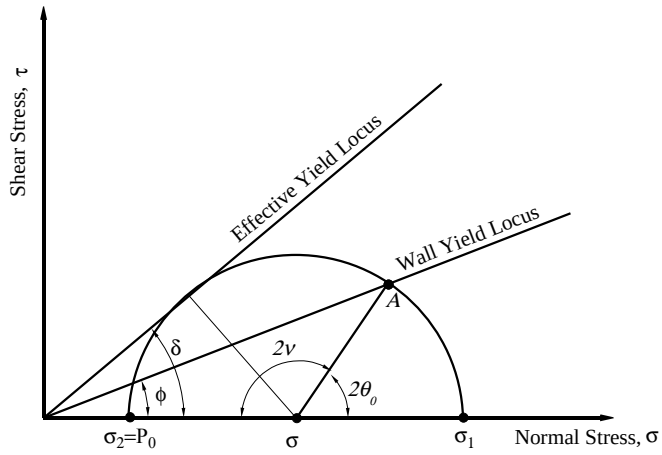


Fig. 3.2 Internal and Wall friction loci

This figure also displays the wall yield locus which describes the friction condition at the powder-roll surface contact. This friction is normally given as the slope, μ, but can also be described by the angle ϕ ($\tan\phi = \mu$).

The frictional condition for slip along the surface along with the magnitude and location of feed pressure P_0 are sufficient boundary conditions to use the Jenike-Shield shear criterion to determine the pressure distribution before the nip region ($\theta > \alpha$).

3.3 Pressure Distribution in the Nip Region

The powder that enters the nip region must be compressed to a final hight equal to the dimension of the roll gap. Johanson applies conservation of mass concept represented by Eq. 3.4 where the change in density is proportional to the change in volume.

Eq. 3.4 $$\gamma_\alpha / \gamma_\theta = V_\theta / V_\alpha$$

This means that powder with a particular mass trapped in a slice under the surface of arc-length ΔL has a volume V_α. As this same mass travels towards the gap it must be compressed to a slice of volume V_θ that is under the same sized arc-length segment ΔL (Fig. 3.3). Thus the only the volume and therefore the density changes as a finite amount of material moves through the roll gap.

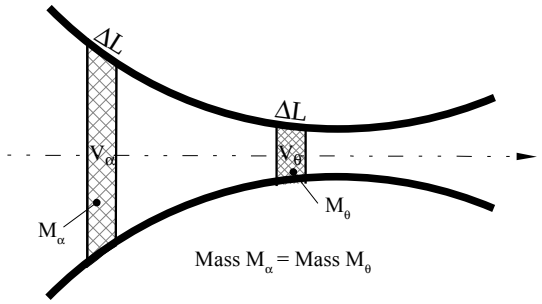


Fig. 3.3 Volume vs Mass

Pressure-Density relationship is experimentally attained using a linear compression test in a tablet pressing device and is presented as the log of density as a function of log pressure:

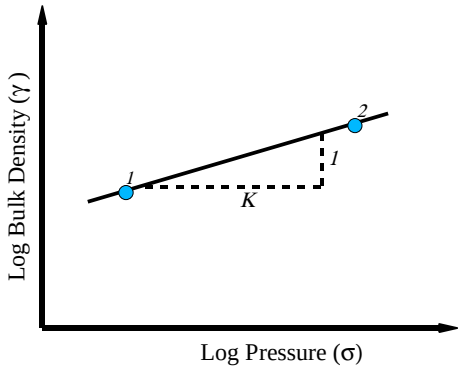


Fig. 3.4 Pressure vs Density

This relationship is often referred to as the Tablet Material Law represented by the following equation:

Eq. 3.5
$$\frac{\sigma_1}{\sigma_2} = \left(\frac{\gamma_1}{\gamma_2}\right)^K$$

The exponent K represents compressibility which is a material property and is constant for a powder with a given moisture content, temperature, and time of compaction. It describes the degree of volume reduction due to applied pressure. Due to linear behavior, attaining the compressibility is simply done by plotting a few points from pressure - density data on a log scale and calculating the slope of the line. The inverse of this derivative is the compressibility K.

It is important to mention that the words pressure and stress are interchangeable due the fact that there is only one constitutive pressure source in each region. For example, in the nip region stress is primarily due to the roll pressure thus $\sigma = P_{roll}$.

From Eq. 3.4 and Eq. 3.5 the following relationship between stress at any θ, σ_θ, and stress at the nip angle, σ_α is derived:

Eq. 3.6 $$\sigma_\theta = \sigma_\alpha \left(\gamma_\theta / \gamma_\alpha\right)^K = \sigma_\alpha \left(V_\alpha / V_\theta\right)^K$$

The volume between an segment of arc-length ΔL, roll diameter, D and roll width, W is given by :

Eq. 3.7 $$V_\theta = \left[S + D\left(1 - \cos\theta\right)\cos\theta\right] \Delta L W$$

By applying this definition to Eq. 3.6 the the pressure distribution relationship between the rolls in the nip region can be obtained, provided the nip angle α is known (Eq. 3.8.) As mentioned previously, since major principal stress in the nip region is much grater then the minor principal stress, an assumption is made that σ is equal to the pressure exerted on the powder only by the roll.

Eq. 3.8 $$\sigma_\theta = \sigma_\alpha \left[\frac{(1 + S/D - \cos\alpha)\cos\alpha}{(1 + S/D - \cos\theta)\cos\theta}\right]^K$$

3.4 Determination of Nip Angle

Johanson postulates that there must exist a point of transition where the powder stops to slip along the roll surface and begins to travel with the same velocity as the rolls. Johanson is able to mathematically describe the pressure distribution for the system where only slip occurs and where the no slip occurs. Central to his idea is the assumption that the gradient of these two pressure distributions is equal at a point: Nip Angle α. Figure 3.5 demonstrates this idea. Solid curve represents the powder undergoing continuous slip and the dashed curve represents the powder traveling with the same velocity as the rollers; thus the intersection of two curves in gives the angular position, α , or the beginning of the nip region. Note that the nip region is from angular position of 0 to angular position α, and the zone where the powder slips along the roll surface is between the angle θ and the point of where pressure gradient of slip state equals 0. This point is also the point of application of feed pressure.

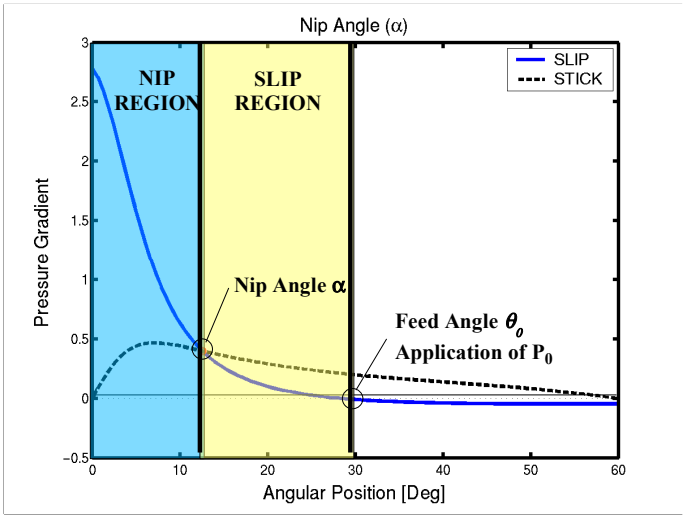


Fig. 3.5 Determination of Nip Angle

The gradient of pressure for the assumption of slip along the roll surface (Eq. 3.9) is derived by using the effective yield-locus equation (Eq. 3.1) combined with equilibrium conditions that form a solvable system of hyperbolic-type equations. However, the discussion of this method of calculation is beyond the scope of this paper.

Eq. 3.9

$$\frac{d\sigma}{dx}=\frac{4\sigma\left(\pi/2-\theta-\nu\right)\tan\delta}{\frac{D}{2}\left[\left(1+S/D-\cos\theta\right)\right]\left[\cot\left(A-U\right)-\cot\left(A+U\right)\right]}$$

Therefore, by equating Eq. 3.9 and pressure gradient equation of the case where no slip along the surface occurs (Eq. 3.10), combined with the assumption that not only the gradients but also the pressures at that point are equal, $\sigma_{slip}=\sigma_{stick}$, and solving for θ will yield the nip angle.

Eq. 3.10

$$\frac{d\sigma}{dx}=\frac{K\sigma_{\theta}\left(2\cos\theta-1-S/D\right)\tan\theta}{\frac{D}{2}\left(1+S/D-\cos\theta\right)\cos\theta}$$

The variables used in the above are derived from the yield and wall loci (Fig. 3.2) and are represented by the following equations:

Eq. 3.11

$$A=\frac{\theta+\mu+\pi/2}{2}$$

Eq. 3.12

$$U=\pi/4-\delta/2$$

Eq. 3.13

$$\nu=\frac{\pi-\arcsin\left(\frac{\sin\phi}{\sin\delta}\right)-\phi}{2}$$

3.5 Pressure Distribution Calculation

The procedure for calculating the pressure distribution in the nip region is as follows: First, the nip angle is determined using the aforementioned method. The nip region pressure distribution equation (Eq. 3.8) requires a value of pressure at the point of transition, σ_α. This value is calculated by numerically solving the differential equation for pressure gradient in the slip region (Eq. 3.9) with initial condition of $\sigma_\theta = \sigma_0$, where σ_0 is the mean normal stress at the position of application of P_0 (Eq. 3.2.) Figure 3.6 demonstrates pressure distributions for two powders, AvicelPH101 and Lactose Monohydrate in a roll press with roll diameter of 130 mm, gap size of 1mm and initial feed pressure of .06 Mpa. [see Appendix for data]

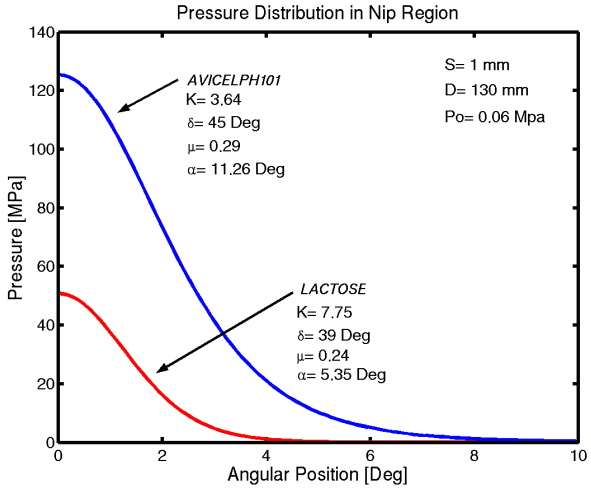


Fig. 3.6 Pressure Distribution in Nip Region

3.6 Roll Force and Torque Calculation

According to Johnson the pressure distribution is not very useful in design of a roll press process, therefore he provides means of calculating roll force and roll torque. When the pressure distribution in the nip region is known the roll force can be calculated numerically using equation 3.14 where the the P_m is the maximum pressure exerted by the rolls on the material and F is represented by equation 3.15. The nip region is the limit of integrations because the pressure there are much greater than those in entrance and exit regions. Figure 3.7 demonstrates the effects of roll gap hight variation on roll force for two different powders .

Eq. 3.14 $$RF = P_m WDF/2$$

Eq. 3.15 $$F = \int_{\theta=0}^{\theta=\alpha} \left[\frac{S/D}{(1+S/D-\cos\theta)\cos\theta} \right]^K \cos\theta \, d\theta$$

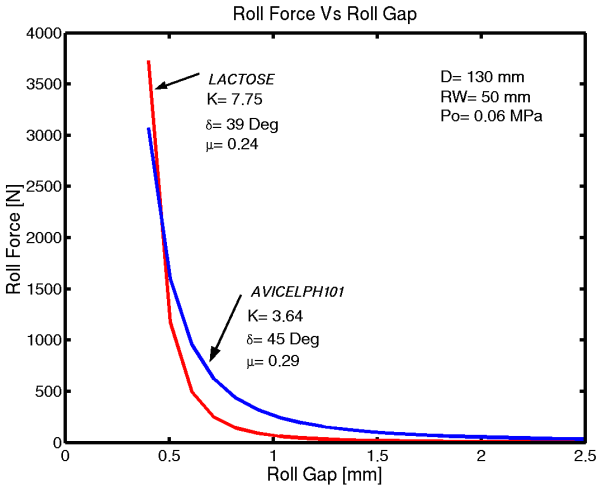


Fig. 3.7 Roll Force vs Roll Gap

In a similar manner roll torque can be calculated using the following equations:

Eq. 3.16 $$T = \int_{\theta=0}^{\theta=\alpha} \left[\frac{S/D}{(1+S/D-\cos\theta)\cos\theta} \right]^{K} \sin(2\theta)\, d\theta$$

Eq. 3.17 $$RQ = P_m W D^2 T/8$$

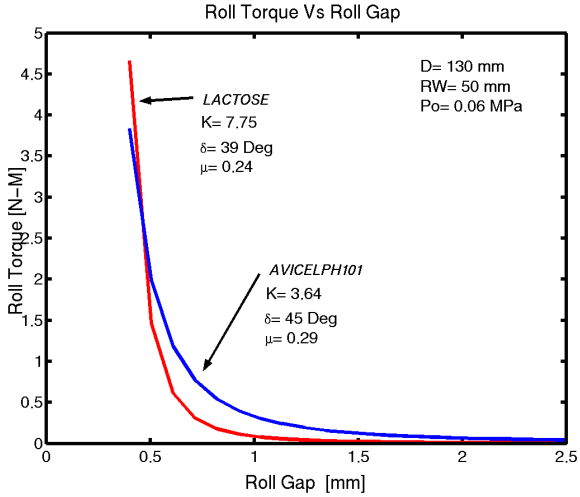


Fig. 3.8 Roll Torque vs Roll Gap

3.7 Johanson Model Application Using Matlab

Matlab mathematics software package was used to create an application implementing Johanson Model of rolling compaction of powder (Fig. 3.9). Matlab was used to numerically solve the non-linear simultaneous and also differential equations discussed in previous sections as well provided an intuitive graphical user interface to demonstrate the capabilities of Johanson Model. Moreover, the program provides a

means of creating and altering databases of powder properties and process parameters which allows for quick analysis of parameter variation on powder compaction. The program was written with modularity in mind to improve portability and efficiency. The source code contains extensive comments to aid in future development.

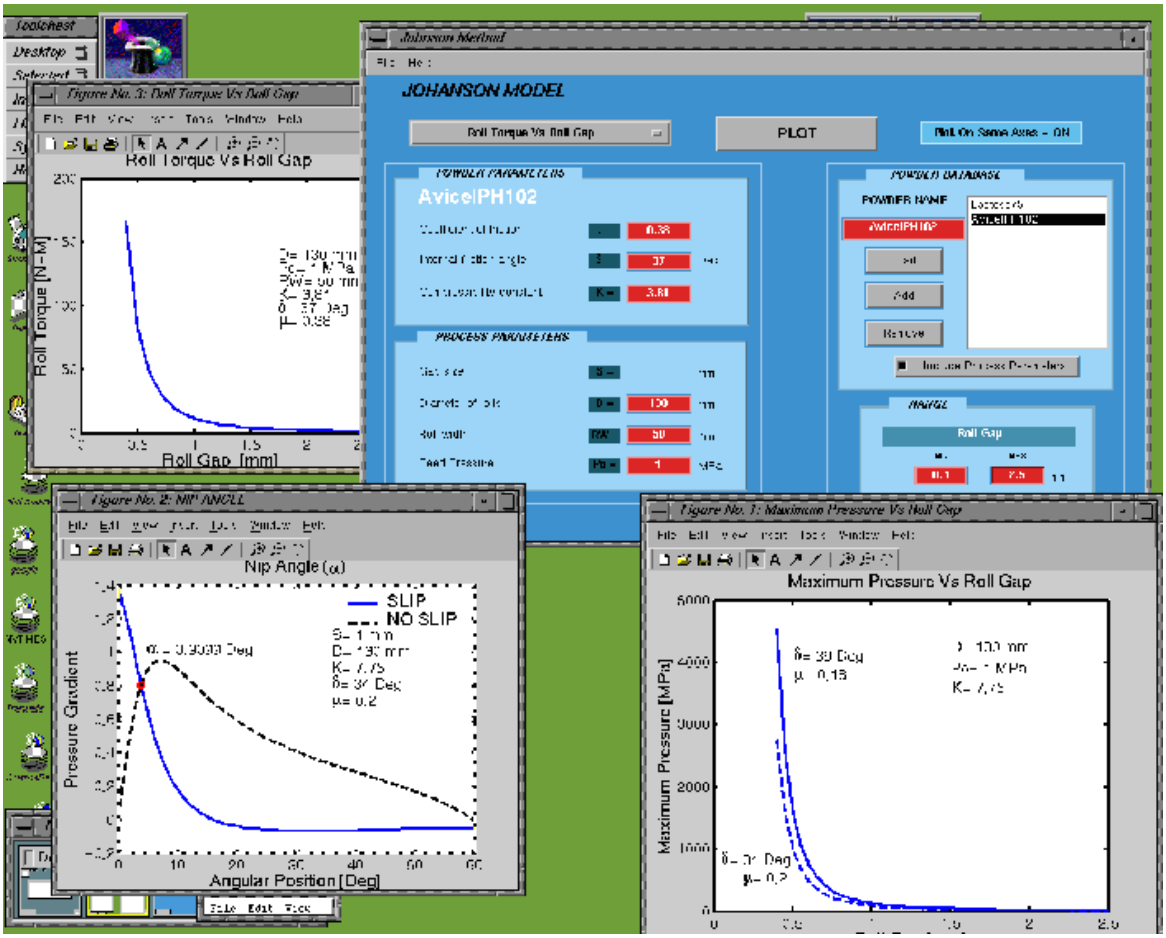


Fig. 3.9 Screen Shot

3.7.1 Example Procedure

The following figure displays the graphical user interface (GUI) which consists of system parameters, database functionality, plot selection and plot variable range (Fig. 3.10).

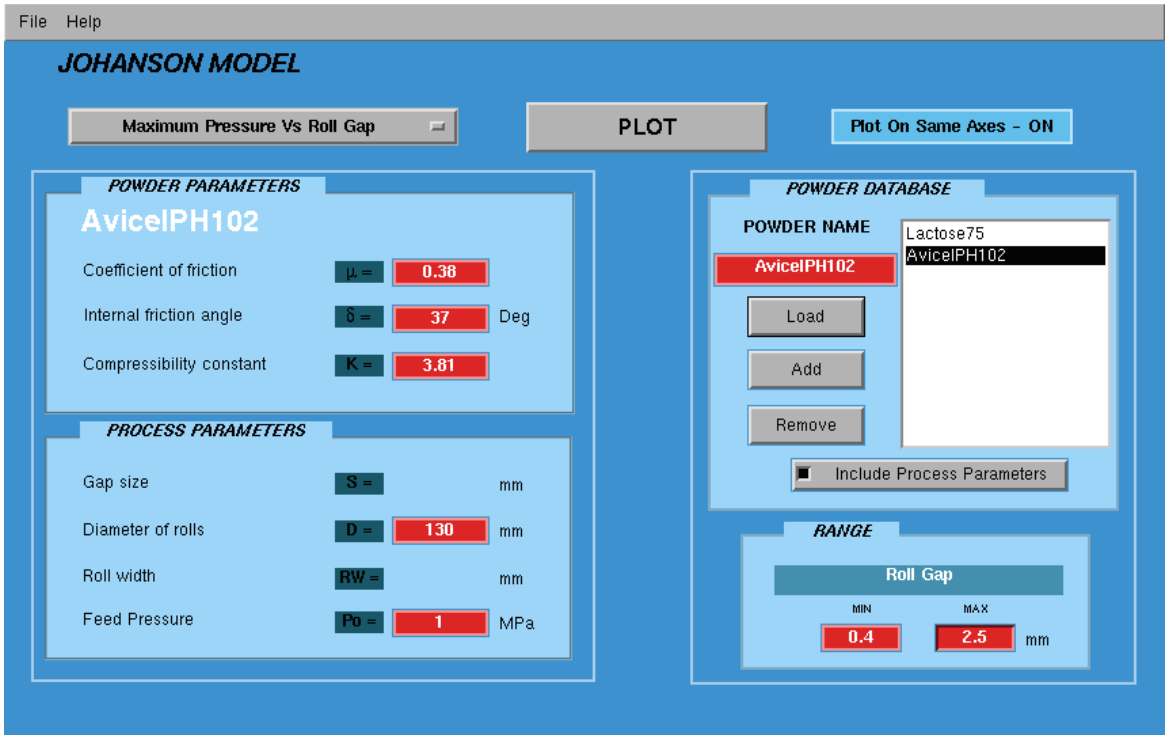


Fig. 3.10 User Interface

The GUI is run in Matlab environment (version 6.1.0.450 Release 12.1) by opening and running the 'nip.m' file. This file contains the main portion of the project: user input logic, algorithms and GUI manipulation.

After the GUI loads the user can immediately begin to experiment with the default values for AvicelPH102 loaded into input areas (Fig. 3.11).

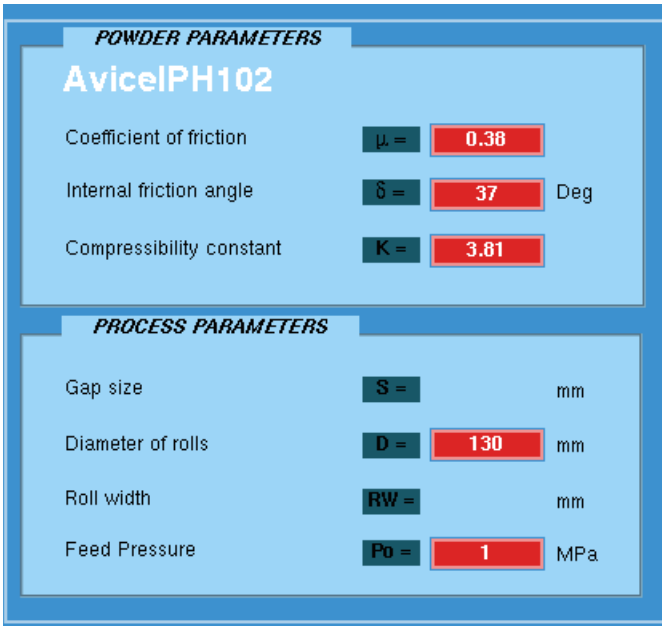


Fig. 3.11 User Input - Parameters

The parameters variables are as follows:

μ = roll surface - powder friction coefficient
δ = internal friction angle of powder
K= Compressibility constant
S= Gap size - smallest distance between rolls
D= Diameter of Rolls
P_0 =initial feed pressure (perpendicular to major principal stress)
RW= roll width

These values can be replaced with any desirable numeric value. In case of a typographical mistake (entering a combination of alpha and numeric characters) a warning window will pop up. It is also advisable to pay attention to the Matlab command line for system and program messages especially if extreme values are entered by the user. The next step is to choose the type of relationship desired from the plot pop up menu. (Fig. 3.12)

When a plot is chosen the user input fields adjust their visibility according to the required parameters for the particular operation. In the case of the Maximum Pressure vs Roll Gap, Gap size input box is hidden because it is the invariant and Roll Width is not used in this calculation (Fig. 3.11).

Fig. 3.12 Plot Menu

After choosing the appropriate parameter values the user should adjust the range of the invariant which will limit the time of calculation and display only the desired results (Fig. 3.13). The range variables are :

rMin = minimum limit
rMax = maximum limit

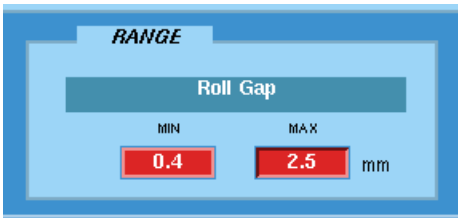


Fig. 3.13 Variable Limits

To began the process the PLOT button should be pressed. At this point the program collects user inputs and runs the algorithm displaying the results in a separate Matlab Figure window which can be edited, saved or printed (Fig. 3.14).

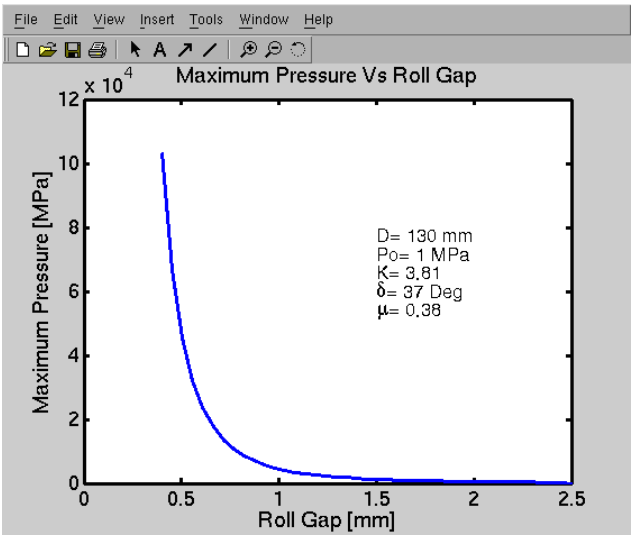


Fig. 3.14 Result

If another plot of the same type is desired the results can be plotted on the same axes as the last plot for comparison (Fig. 3.15).

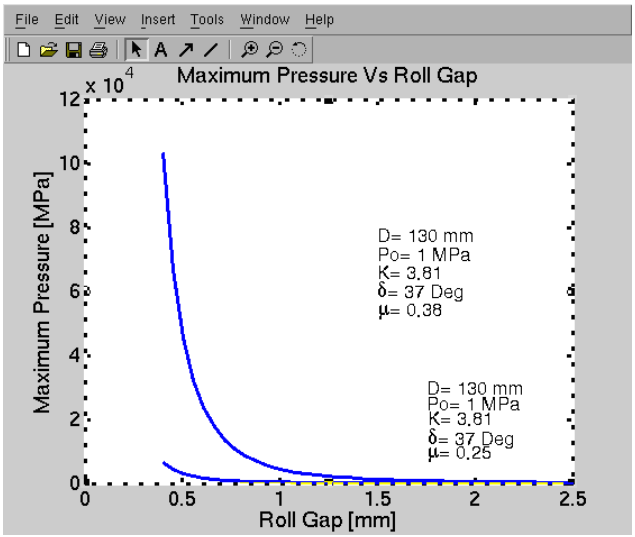


Fig. 3.15 Two Results

To achieve this effect, after altering the parameters, the button labeled 'Plot On the Same Axes' should be pressed displaying 'ON' (Fig. 3.16).

Fig. 3.16 Plot Location Button

This effectively finds a figure containing the same type of plot and displays the new data on its axes. The program also creates data labels for each curve displaying all the respective parameters entered by the user. If multiple curves are on the same axes, these labels may need some user interaction to clarify which data label belongs to which curve.

3.7.2 Powder Database

To improve the utility of the software a database functionality was implemented. The software allows the user to create a database of all input variables: powder properties and process parameters. Using the provided interface this information can be altered, saved and loaded at anytime (Fig. 3.17).

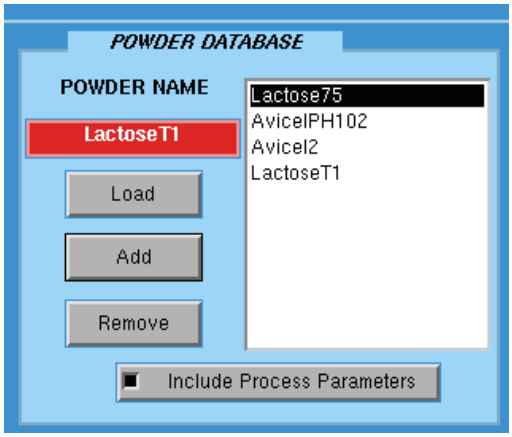


Fig. 3.17 Powder Database

The database data is stored in Matlab variable format file with a suffix of .pdb ; However, there may be a need for powder database use outside of the Matlab environment, thus an export function was created that creates a ASCII text version (.txt) of the currently opened database. This data file is space delimited as in the following example:

Lactose75 0.18 39 7.75 1 130 0.1 50

AvicelPH102 0.38 42 4.75 1 130 0.1 50

Avicel2 0.32 28 4.31 1 130 0.1 50

The fields correspond to parameters in the following fashion:

Name FricCoefficient InternalFrictAngle Compressibility RollGap RollDia P_0 *RollWidth*

Load database file function is located under the menu heading FILE (Fig. 3.18). The program automatically filters the current directory for files ending with a .pdb. After the .pdb database file is opened, loading a particular powder into the user input boxes is done by highlighting the desired powder from the powder database list and clicking on LOAD (Fig. 3.17). If include process parameters option is checked the powder properties as well as system properties will be loaded into their respective input boxes.

Fig. 3.18 File Menu

To add a the current configuration of user inputs for powder parameters type in or alter the current NAME located in a red box above the LOAD button and click ADD. This creates an instance of the powder NAME in the powder list. This procedure also saves the process parameters along with powder parameters.

To remove the powder select the NAME of the powder from the powder list and click REMOVE. This change is only permanently affected after the current powder database is saved. Otherwise reloading the powder database from the FILE menu restores the list to the last saved state.

Furthermore , another database utility stems from the fact that Matlab naturally allows the user to edit, save and print Figures, providing another method for archiving records of experimentation.

3.7.3 Printing results

Matlab print preview function was chosen as a preferred method to print the state of the user interface because it allows the user to rotate, align and fill the page with the GUI. Moreover each figure/axes is fully editable allowing for adding text, lines, arrows and changing colors, fonts, line types, etc. This figures can also be saved for later editing, exported as an image (tiff or jpg) file or just printed.

3.7.4 Algorithm

This section describes the internal workings of the program by using pressure distribution calculation as the example. This example is probably the most useful in the fact that the roll press operator is able to adjust the gap size The following flow chart summarizes the internal program procedure from the time the user presses the PLOT button to the final display of the results.

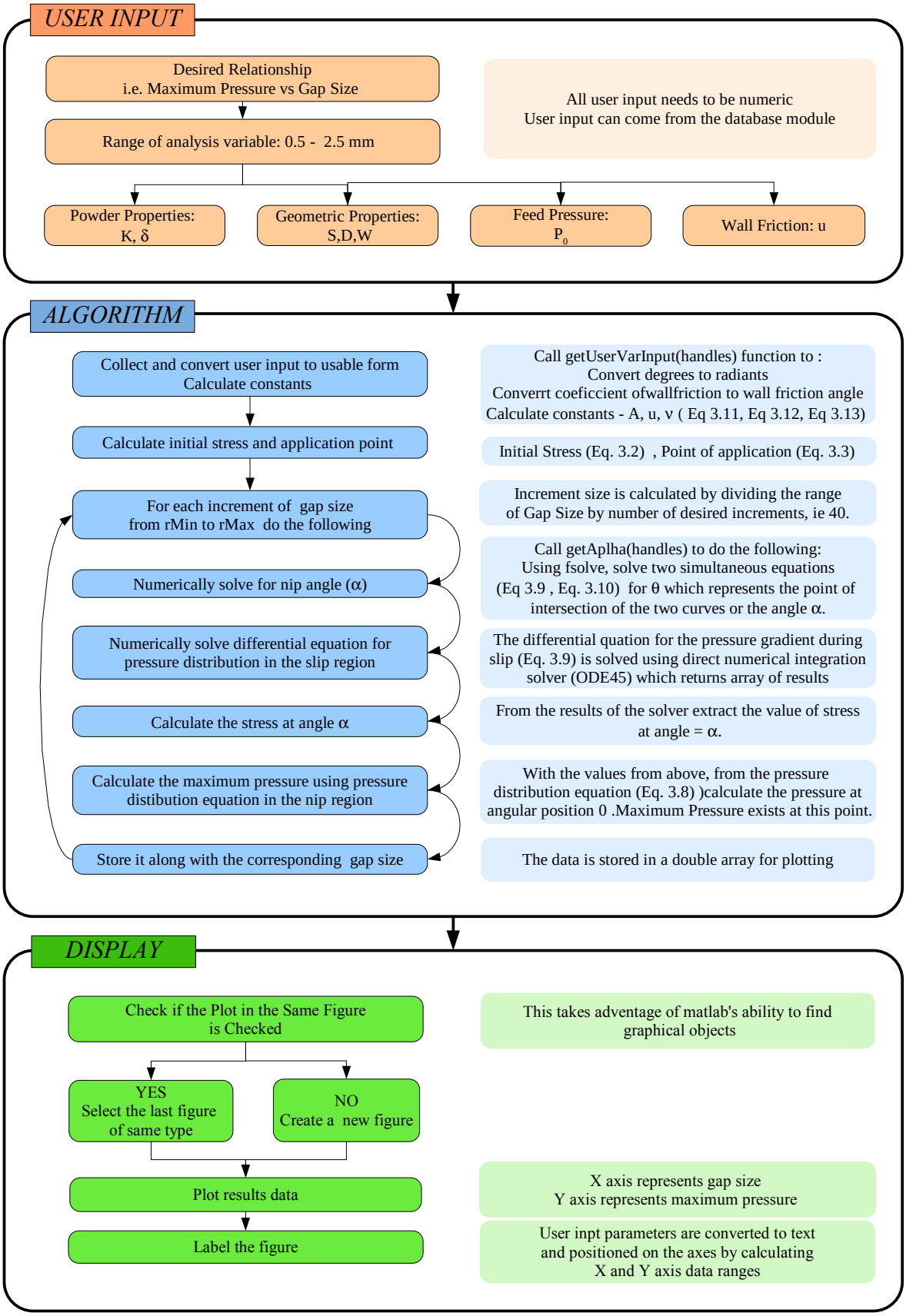


Fig. 3.19 Example Program Flow Chart

First step after the user has initiated the procedure by pressing the PLOT button is the collection of user input data. The program queries all the input fields but only uses the ones required by the particular relationship chosen and in this case Maximum Pressure vs Gap Size. The user inputs are then converted to usable format required by the model, for example angles need to be in radiants and coefficient of friction needs to be converted to angular format. The following function was written for this purpose.

Note: '*%*' denotes a comment

```
function [] = getUserVarInput(handles)
%set global variables used in functions.
global d2r r2d D S W d K v u RW Po rMin rMax
%get all the variables
%angle variables are converted to radiants
d2r=pi/180;      % conversion factor degrees into radiants
r2d=180/pi;      % conversion factor radiants into degrees

%%%%%%%%
% S=Gap size(between rolls)
% D=Diameter of Rolls
% K=Compressibility constant
% d=delta - internal friction angle of powder
% Po=initial pressure at the entrance (perpendicular to major principal stress
% W=phi - wall (roll-powder) friction angle
                    % (attained from friction  coefficient tan(phi)=mu)
% v=cute angle between the tangent to the roll surface and the direction of
% major principal stress which can be calculated using W and d
% RW=roll width
% rMin = min range value
% rMax = max range value
%%%%%%%%
% The next functions grab the string input variables and convert them to
% double precision numbers
S = str2double(get(handles.varS,'String'));
D = str2double(get(handles.varD,'String'));
K = str2double(get(handles.varK,'String'));
d = str2double(get(handles.vard,'String'))*d2r;
Po = str2double(get(handles.varPo,'String'));
RW=str2double(get(handles.varRW,'String'));
W=str2double(get(handles.varW,'String'));
%w at this moment is the friction coefficient so take atan to get the wall
%friction angle in radiants:
W=atan(W);
% calculate v which is used all equations
u=(pi/4)-(d/2);
v=(pi-asin(sin(W)/sin(d)) -W)/2;
rMin = str2double(get(handles.rMin,'String'));
rMax = str2double(get(handles.rMax,'String'));
```

The next step is to calculate the initial stress on the powder at the beginning of the application of the feed pressure as defined by Eq. 3.2 and its location defined by Eq. 3.3.

```
%initial condition
so=[Po/(1-sin(d))];
%point of application of feed pressure ( equivalent to Eq. 3.3)
x0=(pi/2)-v
```

At this point the algorithm has all the necessary data to began the application of the

Johanson Model to calculate the pressure distribution at different gap sizes. The following code is the main loop iterating through 40 different gap sizes. The iterator size is calculated by dividing by 40 the difference between invariant range values, *rMax* and *rMin*. This iterator quantity may be different for other calculations (plot types) to attain the best balance between calculation time and result resolution.

```
xspan=[x0,0];                    %integration limits from Po application to 0
i=1;                             %iterator for array indexing
x=0;                             %maximum pressure is at 0
step=(rMax-rMin)/40;             %Calculate the iterator size
for M=rMin:step:rMax
    S=M;                         %Current Gap size
    a=getAplha(handles);                %calculate the α
    sol=ode45(@slipODE,sspan,so);       %solve the slip differential equation
                                        % using Runga-Kutta and
                                        %initial condition of so at point x0
    sa=deval(sol,a);             %from solution find stress value at  position α
    N(i,1)=S;                    %save the current gap size
                                 %using the initial stress at α calculate the
                                 %maximum pressure (x=0) and save it
   N(i,2)=sa*((1+S/D-cos(a)*cos(a))/(1+S/D-cos(x)*cos(x)))^K;
    i=i+1;                       %increment the array index
end
```

The *getAplha* function accepts the argument *handles* which is passed for formality; it contains information about the GUI, although it is not used in this function. The required variables are simply passed from function to function by their global definition. And since each function refreshes all the variables there is no problem with unwanted variable values. The **fsolve** equation solver from the Matlab Optimization Toolbox finds a root (zero) of a system of nonlinear equations. In this case the two nonlinear equations are the pressure gradient with slip (Eq. 3.9) and the pressure gradient without slip (Eq. 3.10). It returns the point of the intersection of the curves representing the equations which is the angle α in radiants.

```
function a=getAplha(handles)
global d2r r2d D S W d K v u RW Po rMin rMax
InitialGuess = [.1;.1];              %this value seems to work well
Options = optimset('Display','off'); %do not display progress
% call fsolve which returns a vectory of which the first component is the
% angle α and the second part is the pressure gradient  value
XY = fsolve(@SlipStickFun, InitialGuess, Options);
a=XY(1);
```

The **fsolve** method is based on the nonlinear least squares algorithm. The advantage of using a least squares method is that if the system of equations is never zero due to small inaccuracies, or because it just does not have a zero, the algorithm still returns a point where the residual is small. However, if the Jacobian of the system is singular, the algorithm may converge to a point that is not a solution of the system of equations. More specifically, by default **fsolve** chooses the large-scale optimization algorithm which is a subspace trust region method and is based on the interior-reflective Newton method. Each iteration involves the approximate solution of a large linear system using the method of preconditioned conjugate gradients [23].

The above mentioned equations are contained in a function called SlipStickFun in

a SlipStickFun.m file.

```
function F=SlipStickFun(V)
global d2r r2dD S D W d K v u
x=V(1); y=V(2);
F=[y-(4.*(pi/2-x-v).*tan(d))./((D/2.*(1+S/D-cos(x))).*(cot(((x+v+pi/2)/2)-u)-
        cot(((x+v+pi/2)/2)+u)));
        y-(K.*(2.*cos(x)-1-S/D).*tan(x))./((D/2).*((1+S/D-cos(x)).*cos(x)))];
```

The equations are stored in a vector where any multiplication, division or power operations need to have '.' in front of them to allow for element by element matrix operation. Without this symbol the matrix multiplication would involve the inner products between rows and columns.

It is important to note that the stress at θ (σ_θ) is equal in both equations at point of intersection and therefore can be omitted. The pressure gradient values are represented by *y* while angular position θ is represented by *x*.

After the angle α is attained the value is used in calculating the pressure at such angle by solving the differential equation of the pressure distribution with slip (Eq. 3.9.) The initial condition is σ_0 which is the mean normal stress at the position of application of P_0 (Eq. 3.2.) Matlab's ode45 numerical differential equation solver was applied It is based on an explicit Runge-Kutta formula, the Dormand-Prince pair. It is a one-step solver in computing $y(t_n)$, it needs only the solution at the immediately preceding time point, $y(t_{n-1})$. In general, ode45 is the best function to apply as a "first try" for most problems [23]. The ode45 requires a argument in a function format similar to the SlipStickFun.m file.

```
function dsdx=slipODE(s,x)
global d2r r2d S D B d K v u
dsdx=[(4.*s.*(pi/2-x-v).*tan(d))./((D/2.*(1+S/D-cos(x))).*(cot(((x+v+pi/2)/2)-u)-
                cot(((x+v+pi/2)/2)+u)))];
```

After the algorithm finishes the results need to be plotted. First the program calls the *newFigure* function which was created to check whether the user desires to have the data plotted on the same axes. The function searches all the figures open for a figure with the name of the current type of plot; in this case to 'Maximum Pressure Vs Roll Gap'. If such figure exists it is chosen to be the current figure. If more than one exist the last one created is chosen. However if no match is found a new figure with the name of the desired type is created and set to current. Following the data is plotted by using the plot command:

```
%call the figure choosing function.
figure(newFigure(handles,'Maximum Pressure Vs Roll Gap'));
plotA=plot(N(:,1),N(:,2));
```

The following code labels the axes, and uses the **text** function to display the user input data:

```
title('Maximum Pressure Vs Roll Gap');
xlabel('Roll Gap [mm]');
ylabel('Maximum Pressure [MPa]');
xData = get(plotA,'XData');                 % Get the plotted data
yData = get(plotA,'YData');
TPosX=(max(xData))*0.6;                     % returns the max value in x direction
TPosY=(max(yData))*0.8;                     % returns the max value in y direction
Tsp=TPosY*.07;                  % creates a variable for vertical text spacing
```

```
%print user variables on the graph
text(TPosX,TPosY-TSp,0,     ['D= ',num2str(D), ' mm']);
text(TPosX,TPosY-TSp*2,0,   ['Po= ',num2str(Po), ' MPa']);
text(TPosX,TPosY-TSp*3,0,   ['K= ',num2str(K)]);
text(TPosX,TPosY-TSp*4,0,   ['\delta= ',num2str(d*r2d), ' Deg']);
text(TPosX,TPosY-TSp*5,0,   ['\mu= ',num2str(tan(W))]);
```

Figure 3.6 demonstrates pressure distributions for two powders, AvicelPH102 with lubricated wall and Lactose in a roll press with roll diameter of 130.0 mm, gap size of 1.0 mm and initial feed pressure of 1.0 Mpa. Similar methodology was used to represent variations of pressures based on other parameters.

3.8 System Behavior - Parameter Variation Analysis

The compactibility of a granular solid is defined as the ability to be transformed into a compact of a certain mechanical strength. Thus, it is an essential and fundamental property of a tablet mass and a determining factor for successful tablet production. Compactibility is normally assessed by the relationship between compact strength and a process variable, usually the maximum force or pressure applied on the powder during the compaction; Although, other process related factors can effect the mechanical strength of compacted powder [24].

The model provides reasonably accurate values of basic operating parameters such as roll force and roll torque for granular solids that exhibit high coefficient of friction against the roll surface and middle and high values of compressibility constant. Very compressible materials (low K values) and high compaction pressures (ie. 100 Mpa) yield results that do not correlate with experimental data; with errors over 50 % [8].

Although powder properties such as internal friction angle and compressibility are not independent of each other, their effects on the system can be used to predict system's behavior for similar powders. Wherever possible experimentally gathered powder properties were used in the Johanson Model to compare the effects of parameter variation. The experimental data was collected using classical die compaction for compressibility and annular shear tester for internal friction and wall friction [25].

3.8.1 Nip Angle

The nip angle is an important boundary information for other models for rolling compaction of powders. It defines one of the limits for the region where most of the compaction is believed to take place. Therefore it is important to become familiar with the process and powder parameters influence on the behavior of this angle. Although it is very difficult to measure the nip angle, literature exists and states that the model provides a good estimate of the nip angle. The results are in agreement with experimental data especially for gravity fed roller presses [8].

Generally, for a set of parameters, if only internal friction angle increases the nip angle increases and as a result more powder is drawn into the space between the rolls yielding a higher density compact requiring greater roll pressure, Fig. 3.20.

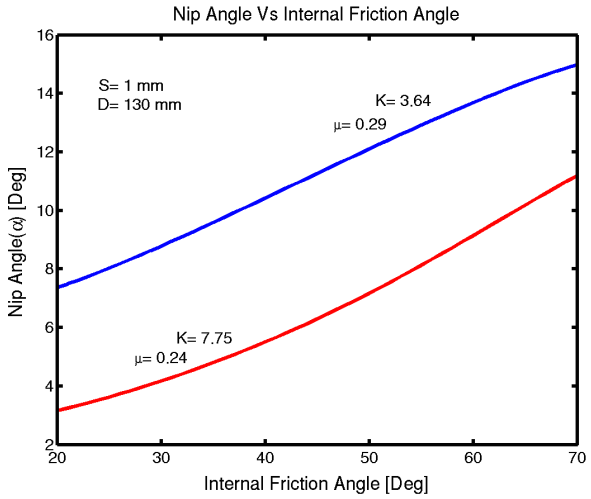


Fig. 3.20 Nip Angle vs Internal Friction

Similar effect, but not nearly as strong, is observed by increasing the coefficient of friction between the roll wall and powder as shown in Fig. 3.21.

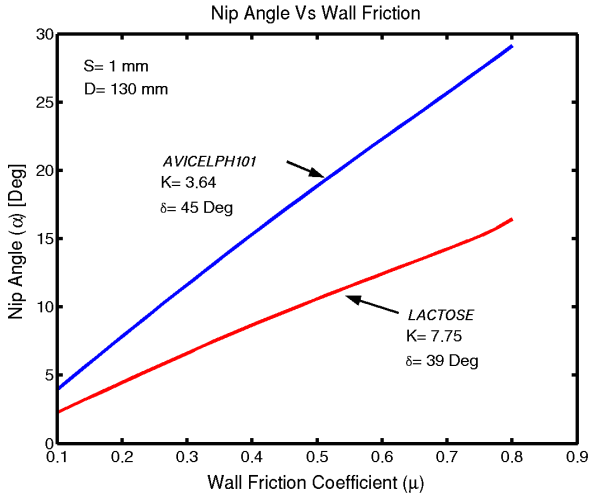


Fig. 3.21 Nip Angle vs Wall friction

In Fig. 3.22, it is observed that the nip angle decreases significantly for materials with low compressibility, or high K value. As for very compressible materials, according to Johanson there may not exist a region in which slip occurs.

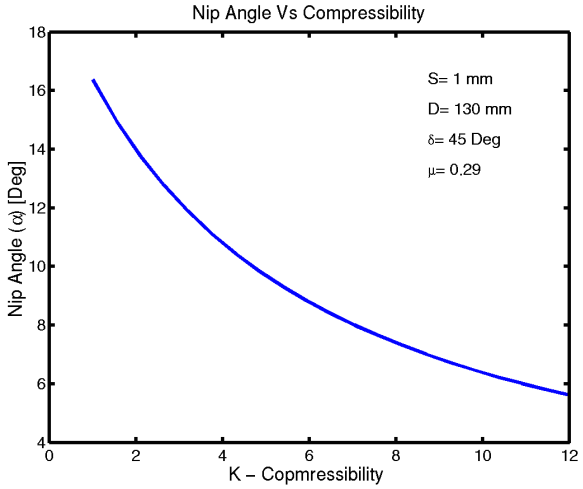


Fig. 3.22 Nip Angle vs Compressibility

Surprisingly, the nip angle does not significantly depend on the magnitude of dimensional parameters, roll gap and roll diameter. As demonstrated in the following figure, the nip angle fluctuates only a few tens of a degree while the press undergoes a considerable change in roll gap to roll diameter ratio.

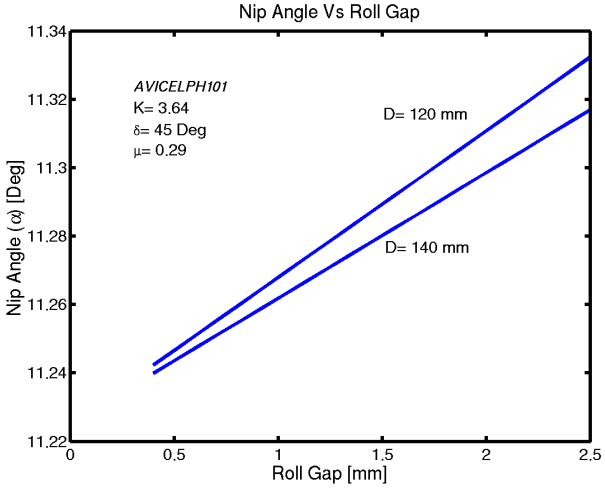


Fig. 3.23 Nip Angle vs Roll Gap

3.8.2 Maximum pressure

The pressure distributions for different powders have similarly shaped curves over the time of roll contact with maximum pressure appearing in the narrowest part of the gap; angular position 0, Fig. 3.6. Therefore, one can assume that maximum pressure is a good representation of the general trend of pressure distribution in the nip region. It is also the important factor in predicting final compact properties.

Maximum pressure is directly affected by the feed pressure [1]. This is especially evident in a system that requires relatively low compacting force, where the change of feed pressure significantly increases the pressure exerted on the powder, as displayed in

Fig. 3.24.

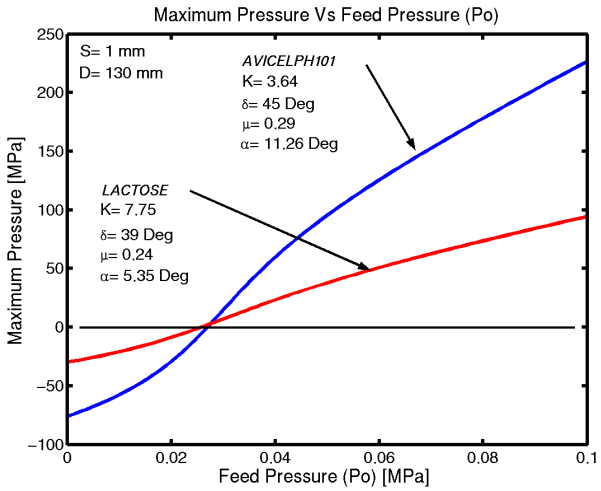


Fig. 3.24 Maximum Pressure vs. Feed Pressure

Generally, the feed pressure for a roll press with rollers of 130 mm diameter is in the range of 0.04-0.06 Mpa [1]. The pressure results in that date seem to be reasonable. As for the instance when the feed pressure is taken below 0.0275 MPa, the resulting maximum pressure (Gap=1mm) becomes negative. This phenomenon may be related to the fact that the powder needs to be fed into the roll gap with a pressure above a certain limit, otherwise compaction will not occur.

Fig. 3.25 displays effects of gap size parameter which according to Johanson is the most influential press dimension in determining the pressure in the nip region, while the roll diameter has much less of an effect. As the gap size becomes smaller the pressure rises exponentially to very extreme values which will unlikely occur in reality due to buildup of powder prior to the nip region (overfeeding). On the other hand, if the gap is too large too little powder may enter the nip region where poor compaction is expected.

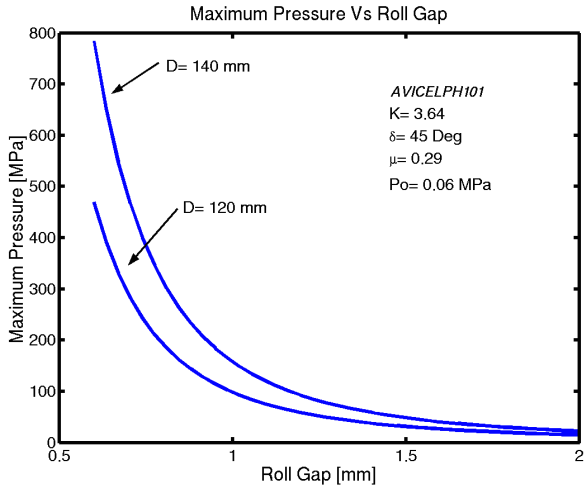


Fig. 3.25 Maximum Pressure vs. Roll Gap Size

Using the information from the above graph the roll press operator is able to predict the behavior of the powder. If he or she is using a roll press type that allows for

adjusting the gap size the desired compact density can be achieved by adjusting the separation height of the rigid rollers until the optimum pressure required to compact the powder is achieved. As mentioned before this pressure is obtained experimentally using tablet compaction instruments. In the case of the other type of press which is equipped with a hydraulic roll force adjustment (lacks direct gap size adjustment) the same graph can be used to deduct the gap size which may be utilized in verification of process calibration.

Fig. 3.26 shows the effects of friction between roll surface and powder. Rough rolls increase the force that drags the granular solid into the region of compaction, and as a result of Johanson's assumption of mass conservation, a denser compact is expected resulting from an increase in maximum pressure. Occasionally there is a problem with material gluing to the roll surface resulting in unwanted film and negatively influencing press performance. In this case the roll surface may need a mechanical cleaner, or a lubricant directly applied to the wall surface. The other solution is to add a lubricant such as Magnesium Stearate to the bulk powder in concentrations of 0.5 - 1.0 percent.

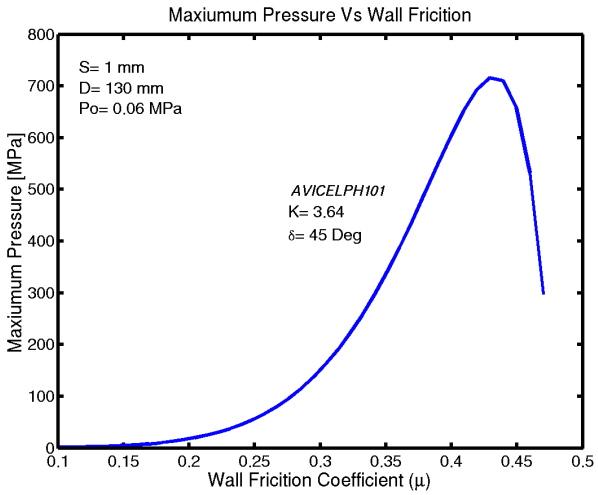


Fig. 3.26 Maximum Pressure vs. Wall Friction Coefficient

The curve in the above plot reverses its trend between coefficient values of 0.4 and 0.5. This is a model phenomenon which can be explained via the relationship between rate of change of force and the rate of change of contact surface area. As wall friction increases the nip angle increases (Fig. 3.21). The nip angle is one of the boundary points of the contact surface; the other ones are the neutral angle 0 and the roll width which stay constant for this simulation. Therefore, if the nip angle increases the contact surface increases and if the rate of this area increase is greater than the rate of increase of roll force, the pressure (Force/Area) decreases.

The next graph (Fig. 3.27) displays the effects of compressibility factor variation on maximum pressure as well as the effect of increasing the internal angle of friction.

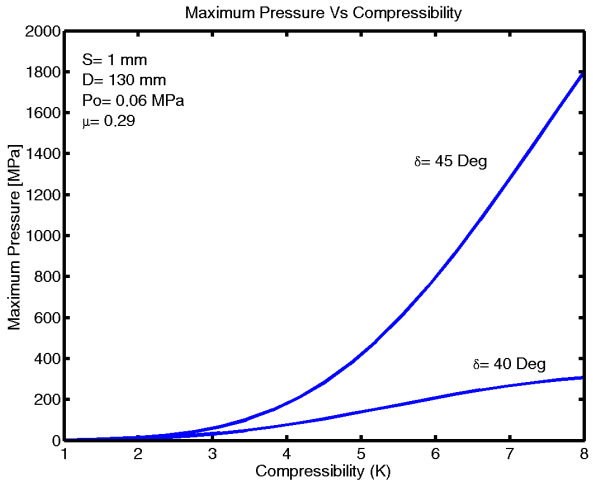


Fig. 3.27 Maximum Pressure vs. Material Compressibility

3.9 Conclusion

Compactibility is often assessed by the relationship between compact strength and maximum pressure applied to the powder during compaction. In many cases, due to simplifications made while modeling powder behavior resulted in the model's inability to represent the system correctly. However, the most important result of the Johanson model is that it provides the value of Nip Angle, which is often used as a boundary condition for other analytical and numerical models.

It is clear that a wide variation in press output exist when different materials are used in the same compaction process configuration. The most influential press dimension in determining the maximum pressure is the roll gap (S). Moreover, increasing powder-roll friction increases the effectiveness of the process. As for the nip angle, if the roll gap-roll diameter ration (*S/D*) is much less than 1, the nip angle seems to only depend on the following material properties: compressibility constant (K), effective angle of friction (δ), and wall friction (μ). One can further deduce that the feed pressure has no effect on the nip angle at all, however in reality this is probably not the case.

The discrepancies between Johanson method and experimental results may be due to the use of simple material material yield criterion for both slip and no slip regions, especially at the point of transition where high stress values are permitted and the bulk material law remains unchanged [7]. The other disadvantage of the model is that it only uses the mean shear stress ($\sigma = (\sigma_1 + \sigma_2)/2$), but it is often desirable to work with the principal stresses for more correct stress distribution. Moreover, mass conservation assumption is rather simple and it does not accurately describe the heterogeneous flow of material undergoing compaction in the nip region [1].

The rate at which a load is applied to material can affect the results obtained; in rolling compaction the roller speed is directly linked to this phenomenon. In his method, Johanson does not consider velocity of the rollers which has been demonstrated to affect the compaction process [1]. However, the variation of roll velocity relative to the feed pressure (screw feeder velocity) is tolerable if it is within a certain range where the strip of compact produced exhibits enough cohesion and mechanical strength [5]. Furthermore,

the relationship between these two parameters creates an envelope of possible values which will have minor effects on the compaction as long as compact is created.

Moreover, the model does not account for the change of the friction coefficient of wall surface-powder interaction which is directly dependent on the change of the density that varies greatly as the powder becomes compacted [8,25] . It also doesn't incorporate the effects of powder cohesion and its change with density.

In conclusion, the Johanson model is considered the corner stone of the study of rolling compaction of powders. However, due to aforementioned assumptions and simplifications it does not provide sufficient simulation of the process. Nonetheless, it is one of the first models to allow engineers to analyze the process and allow an operator to predict initial step parameters in a iterative roll press calibration procedure.

3.10 List of Symbols

D - roll diameter

F - force factor

K - compressibility constant for granular solid

P_m - vertical pressure at angular position 0

P_0 - horizontal feed pressure

RF - roll separating force

RT - roll torque for one roll

S - roll gap

T - torque factor

V_θ - volume between arc length segments ΔL at position θ

V_α - volume position at $\theta = \alpha$

V_m - volume position at $\theta = 0$

W - roll width

α - nip angle

γ - bulk density of granular solid

γ_θ - bulk density at θ

γ_α - bulk density at $\theta = \alpha$

γ_m - bulk density at $\theta = 0$

ΔL - element of arc length

δ - internal (effective) angle of friction

ϕ - wall friction angle (roll surface - powder) $\tan \phi = \mu$

μ - coefficient of friction (roll surface - powder) $\arctan \mu = \phi$

ν - acute angle between direction of σ_1 and tangent to roll surface

σ - mean normal stress in granular solid

σ_1 - major principal stress (y)

σ_2 - minor principal stress (x)

σ_0 - mean normal stress at a point of application of P_0

σ_α - stress at nip angle

θ - angular position

θ_0 - feed angle

4 Slab Method

4.1 Introduction

The slab method was developed by Siebel and von Karment for sheet metal forming [8]. It has then been adopted to rolling compaction of metal powder rolling by Katashinskii [9]. The main difference between the different applications of slab method to roll compaction is the type of material yield criterion is implemented which in case of sheet metal is the Von Mises yield criterion.

The objective of the slab method is to provide an analytical model for powder compaction which takes into account powder behavior and equilibrium equations representing forces exerted on a the powder in the nip region. The advantage of this model is that it provides information about the two principal stresses unlike the Johanson model which only describes the mean stress. It also has potential to include characteristics of powders such as cohesion and global friction between particles. However, as more complex powder behavior characteristics are implemented the ability to analytically solve the resulting differential equation decreases considerably. If possible such analytical relationship would provide an operator the pressure exerted on the powder which compared with experimental data can provide predictable compact properties such as density and strength.

4.2 Modeling Rolling Compaction with Slab Method

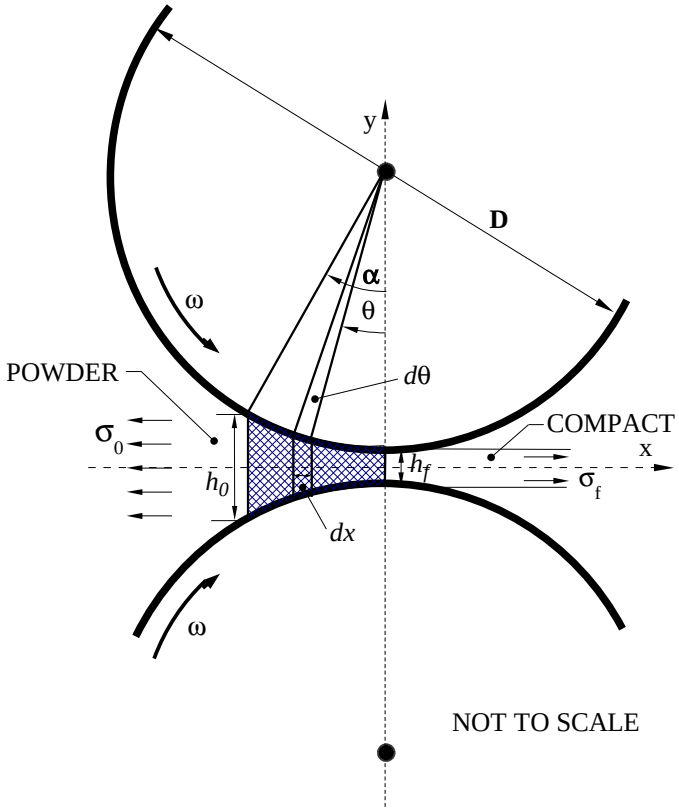


Fig. 4.1 Slab Method

Although the rolling process is carried out at low speeds, it is natural to assume

that static analysis model is a suitable approach. The compaction zone is divided into differentially small slices bounded at top by the roller surfaces (Fig. 4.2). These boundaries can be described as straight lines tangent to the roller contours due to strip's small width, *dx*. If the forces of inertia acting on the material being formed are ignored, only the stresses acting on the four sides of the trapezoidal slab need to be taken into account. At the material-surface contact point friction exists which creates shear stress, τ. From the stresses, the forces are attained by multiplying the direct and tangential components by their respective surface areas. Than by resolving the forces into horizontal and vertical forces equilibrium equations can be written.

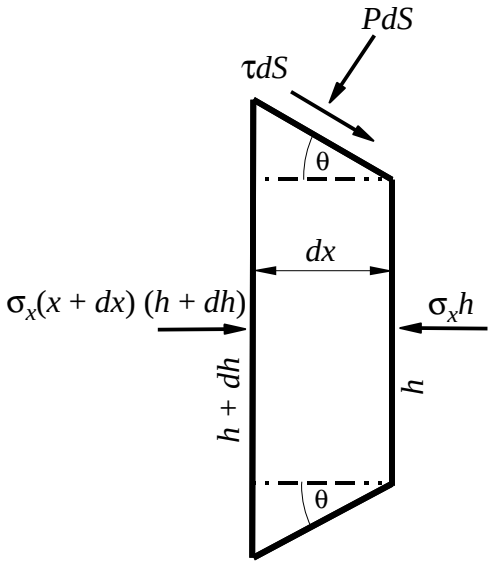


Fig. 4.2 Free Body Diagram of Slab (Entry)

The general problem includes the following aspects:

Process Parameters :

Pressure of feeder (P_0), Angular Velocity of Rollers (ω)

Roller Pressure (P), Nip Angle (α)

Geometric Parameters :

Diameter (D), Gap size (h_f)

Powder Parameters :

Cohesion (β), Internal friction angle (δ),

Relative Density (ρ =Apparent/True), Yield function

Tribologic Parameters :

Powder - roll surface friction definition (μ)

These elements can also be categorized into groups of unknown and given data:

Unknown parameters :

Stress in x and y (σ_x, σ_y) , ρ , P as a function of position

Given Parameters :

β, μ, α, initial stress at entrance ($P_0=\sigma_0$), H_f, Yield Function, ω, D

The desired function is

$$P=f\{P_0, \omega, D, h_f, Powder(\beta, \delta, Yield\ Function), \mu\}$$

Eq. 4.1Desired Relationship Equation

The general approach is to define the assumptions, write down the constitutive equilibrium equations, apply boundary conditions and yield criterion to formulate a governing differential equation.

The assumption for the following example are:

- *Rollers are rigid.*
- *Plane sections remain plane as they pass through the rolls (plane strain).*
- *Stresses are distributed uniformly within slab elements.*
- *Friction force in the contact area follows Coulomb's law.*
- *The circumference of the roller is much larger than the contact area.*
- *Roll pressure is equal to the principal stress in the y direction*

The boundary conditions include nip angle which can be obtained from Johanson model or measured experimentally; However, either of these experimental methods are currently not very verifiable. At the entrance the $\theta = \alpha$ and $h = h_0$ while at exit $\theta = 0$ and $h = h_f$. As for pressure, the major source of stress is the pressure (P_0) produced by the feeder from the entry end and by the compact from the exit end.

The friction condition between the powder and roll surface is assumed to be constant for sheet metal rolling analysis. However this assumption is not believed to be correct as internal friction, cohesion and wall friction properties are dependent on the density of granular material. A key issue exists here because wall friction affects the pressure exerted on the powder and yield (densification) is a function of pressure. And as mentioned before friction is a function of density making the wall friction directly related to the pressure and vice versa.

The yield criterion is a major element of modeling compaction as it is the definitive element by which we classify materials. There exist a number of different criterion that represent yield limits of materials. The following few have been used in metal and powder research:

- **Von Mises (used for solids)**

$$\sigma_x^2+\sigma_y^2-\sigma_x\sigma_y=Y^2$$

Eq. 4.2 Von Mises Yield Criterion

Elliptical in nature along the x axis, but for dense materials it approaches

straight lines. This model is predominantly used in modeling of metallic materials because they exhibit little change in volume.

- **Kuhn - Downey (used for porous media)**

$$\sigma_x^2+\sigma_y^2-2\,\nu(\rho)\sigma_x\sigma_y=Y^2$$

Eq. 4.3 Kuhn - Downey Yield Criterion

$$\nu(\rho)$$

Eq. 4.4 Poisson ratio as a function of density.

Takes in account the change in density. Slab method implementation by Dec [8]paper uses this yield function which has traditionally been applied to metal powder modeling. Note that as ν goes to 0.5 and ρ increases to 1 this model's behavior approaches Von Mises model.

- **Drucker - Prager/Cap model (used for porous media)**

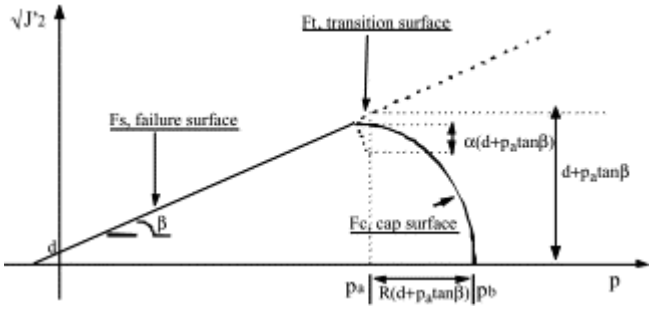


Fig. 4.3 Drucker-Prager Cap model.[13]

This model is an extension of Drucker-Prager yield function for admissible stresses. It consists of three equations each used in different states of the material as demonstrated by the three surfaces in Fig. 4.3. For this reason this model is very difficult to implement in the Slab method. Although it may be possible to use along with numerical integration. Drucker-Prager Cap model describes a cohesive powder more precisely than the previous models.

4.3 Example Slab Method Formulation

Using the above assumptions the following calculation procedure can be used to begin the initial formulation of the slab method as applied to rolling compaction. Due to the plain strain and roller diameter assumptions the maximum principle stress is equal to roll pressure for small angles ($\sigma_y = P$). The following procedure is applied to the nip region.

Horizontal forces:

$$(h+dh)\sigma_x(x+dx)-h\,\sigma_x(x)\approx$$

$$\approx \sigma_x(x)(h+dh)+\sigma_x(dx)(h+dh)-\sigma_x(x)h$$
$$\approx \sigma_x(x)h+\sigma_x(x)dh+\sigma_x(dx)h+\sigma_x(dx)dh-\sigma_x(x)h$$
$$\approx \sigma_x\frac{dh}{dx}+\frac{d\sigma_x}{dx}h+\underbrace{\frac{d\sigma_x}{dx}dh}_{zero}$$
$$\approx \sigma_x\frac{dh}{dx}+h\frac{d\sigma_x}{dx}$$
$$\approx \frac{d(\sigma_x h)}{dx}$$

Adding the horizontal components of the roll pressure and shear the following expression is derived for the equilibrium forces in the x direction.

$$\sum F_x=0\Rightarrow\frac{d(\sigma_x h)}{dx}-PdS\sin(\theta)-\tau\,dS\cos(\theta)$$

With the following geometric relations:

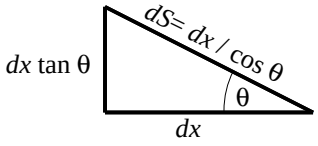


Fig. 4.4 Geometric Relations

a substitution for contact surface *dS* is made to yield:

$$\Rightarrow\frac{d(\sigma_x h)}{dx}-P\frac{dx}{\cos\theta}\sin(\theta)-\tau\frac{dx}{\cos\theta}\cos(\theta)$$

and simplifying :

$$\Rightarrow\frac{d(\sigma_x h)}{dx}-P\,dx\tan(\theta)-\tau\,dx$$

Now with the assumption of Coulomb friction between the powder and the roll :

$$\boxed{\tau=\mu P}$$

Eq. 4.5 Coulomb Friction Relation

The expression can be further simplified

$$\frac{d(\sigma_x h)}{dx}=P\,dx\tan(\theta)-\mu\,P\,dx$$
$$\frac{d(\sigma_x h)}{dx}=P(\tan(\theta)-\mu)dx$$

and finally since the slab is symmetric about the x axis the equation representing the balance of forces is:

$$\boxed{\frac{d(\sigma_x h)}{dx} = 2P(\tan(\theta) - \mu)dx}$$

Eq. 4.6 Force Equilibrium Equation in X direction

Now similarly the vertical force equalibrum expression is derived :

$$\sum F_y = 0 \Rightarrow PdS\cos(\theta) + \tau\, dS\sin(\theta) - P_y$$

$$\Rightarrow P\frac{dx}{\cos\theta}\cos(\theta) + \tau\frac{dx}{\cos\theta}\sin(\theta) - P_y$$

substitute the surface *dS* in terms of q and simplify:

$$\Rightarrow P\,dx + \tau\,dx\tan(\theta) - P_y$$

use the coulomb friction relation to write the expression in terms of *P* and θ :

$$\Rightarrow P\,dx + \mu P\,dx\tan(\theta) - P_y$$

and by factoring out the roll pressure we get:

$$P_y = P(1 + \mu\tan(\theta))dx$$

Expanding the equationg to include the whole slab (symmetry about the x axis) yields:

$$P_y = 2\mathrm{P}(1 + \mu\tan(\theta))dx$$

For small angles P_y is approximately equal to σ_y:

$$\sigma_y = 2P(1 + \mu\tan(\theta))dx$$

Rearranging the equation gives the expression for roll pressure in the y direction:

$$\boxed{P = \frac{\sigma_y}{2(1 + \mu\tan(\theta))dx}}$$

Eq. 4.7 Force Equilibrium Equation in Y direction

Combining equilibrium equations (Eq. 4.6 and Eq. 4.7) :

$$\frac{d(\sigma_x h)}{dx} = 2(\tan(\theta) - \mu)dx\frac{\sigma_y}{2(1 + \mu\tan(\theta))dx}$$

and simplifying yields the force equilibrium expression for the whole slab:

$$\frac{d(\sigma_x h)}{dx} = \frac{\sigma_y(\tan(\theta)-\mu)}{(1+\mu\tan(\theta))}$$

Eq. 4.8 Force Equilibrium Equation for Slab

dx in terms of angle θ:

$$dx = R\cos(\theta)\,d\theta$$

substituting for *dx* yields:

$$\frac{d(\sigma_x h)}{d\theta} = \sigma_y R\cos(\theta)\frac{(\tan(\theta)-\mu)}{(1+\mu\tan(\theta))}$$

The above equation needs one more step to convert it to usable form of one unknown parameter (σ). Yield equation provides this required relationship between σ_x and σ_y. For this there are a number of options varying in complexity and level of correctness for a given material.

Moreover for further investigation the equation relating hight and angle θ is :

$$h = h_f R(1-\cos(\theta))$$

and for small angles this expression can be approximated to:

$$h = h_f R\,\theta^2$$

The resulting differential equation combined with boundary conditions may or may not be solvable by hand therefore a Runga- Kutta or other numerical methods can be used.

4.4 Conclusions

The slab method displays much promise to solve this half-century old problem of rolling powder compaction. However, the method does not provide information about the tapped powder zone prior to nip region. It also assumes that the boundary condition information for the nip angle is available. There exists very little evidence for experimental measurement of the nip angle as well as verified application of the Johanson method of attaining this data. The proper characterization of friction is central to the success of simulation procedures, especially in powder compaction process. It is the main mechanism by which the powder undergoes compaction. If the friction between the roller and the powder is too low, even with the help of a screw feeder, the powder will not be drawn through the roll stand. It was shown by Michrafy that density variations in compacts made via the granular material compaction are mainly attributed to friction [14]. The friction phenomenon in powder compaction emanates mainly from two sources: inter-particle friction and powder-die wall friction. These two powder properties are directly related to density. As a powder is continuously shearing due to applied pressure its tribologic properties change and as a result effect the transformation of pressure from roll through friction. Iterative solving may be necessary to apply experimental data to the model.

Most recent work applying the slab method to powder compaction is done by

Schonert and Sander [10]. Unique to their approach is use of the normal to shear stress relation (transmission coefficient) in a powder. This coefficient serves as a substitute for a yield criterion by representing the relative amount of pressure transmitted from the σ_x to σ_y.

The major hope of using the slab method is that it may provide a mathematical formula that can be used without the need of a computer, perhaps only a four function calculator.

4.5 List of Symbols

dx - slab width

dS - slab surface in contact with roll wall

D - roll diameter

P_0 - feed pressure in the x direction

P_x - pressure in the x direction

P_y - pressure in the y direction

P - roll pressure

R - roll radius

h - height of powder section

dh - change in height from slab to slab

h_f - height of compact equal to roll gap height

h_0 - height of powder at start of compaction

Y - yield

α - nip angle

β - cohesion

δ - internal (effective) angle of friction

ϕ - wall friction angle (roll surface - powder) $\tan \phi = \mu$

μ - coefficient of friction (roll surface - powder) $\arctan \mu = \phi$

ν - Poisson ratio

ρ - bulk density of granular solid

σ_x - major principal stress in x direction

σ_y - major principal stress in y direction

σ_0 - horizontal stress at the beginning of compaction

σ_f - horizontal stress at the end of compaction

σ_α - stress at nip angle

θ - angular position

$d\theta$ - small change in angular position corresponding to dS

τ - shear stress

ω - roll velocity

5 Finite Element Method

5.1 Introduction

There exist numerous types of finite methods available, the most prominent are:

FEM - Finite Element Method for Continuum Media (solids)

FDM - Finite Difference Method (differential equation solver)

CVM - Control Volume Method = **C**FD Control Fluid Dynamics = **FDM** with integration on the control value.

While all these may be great modeling methods each one is suitable for a particular application. The FEM suits the problem at hand and is implemented in the ABAQUS commercial FEA package. [12] The finite element method seems to be the most versatile approach because it takes account of substantial information about geometry, powder behavior, and frictional conditions. For this fact nearly realistic computer experiments are possible.

ABAQUS is composed of two different packages: ABAQUS/Explicit and ABAQUS/Standard (implicit). The main difference between these two procedures is that explicit methods require a small time increment size that depends solely on the highest natural frequencies of the model and is independent of the type and duration of loading. Simulations generally take on the order of 10,000 to 1,000,000 increments, but the computational cost per increment is relatively small. Implicit methods do not place an inherent limitation on the time increment size; increment size is generally determined from accuracy and convergence considerations. Implicit simulations typically take orders of magnitude fewer increments than explicit simulations. However, since a global set of equations must be solved in each increment, the cost per increment of an implicit method is far greater than that of an explicit method [26]. Although, some equations can't be solved by explicit integration and thus requiring the use of implicit method.

The explicit dynamics method was originally developed to analyze high-speed dynamic events that can be extremely costly to analyze using implicit programs. The explicit dynamics procedure is often used for quasi-static simulations involving complex non-linear effects found in complex contact conditions. The explicit central difference method is used to integrate the equations in time; therefore discrete mass matrix used in the equilibrium equations plays a crucial role in both computational efficiency and accuracy. Quazi-static analysis incorporates rate-independent material behavior where time is not very important. The mass scaling is used to adjust the mass of the model artificially to reduce the solution time. ABAQUS provides automatic mass scaling but it is often more desirable to choose one by the user, although it is more like art than science.

5.2 Modeling Rolling Compaction with ABAQUS

The ABAQUS file is an instructions code providing all the necessary information for the program to create the model and run the simulation.

- Node definition
- Element definition
- Surface definition

- Assign element properties (rigid body, material, etc)
- Define material properties
- Define boundary conditions
- Define contact surfaces
- Define step properties (number of increments, dynamic explicit, etc)
- Define output results (all, strain, velocity, reaction force, etc)

General application in ABAQUS starts with the definition of the objects involved. The mesh is first defined, and since the press is symmetric about the x axis, symmetry is assumed to decrease design and computation time. The elements are created from four nodes in a systematic fashion. Generally for a two dimensional model, they are plane-strain continuum elements with reduced integrations (CPE4R). Following, is the definition of the material and then the boundary conditions (roll is defined as a rigid surface, dynamic explicit, velocity of roll, friction etc.)

The last step is to define the desired data and acquisition frequency as well as the time limit of the procedure.

Two approaches may be used to solve the rolling compaction; method based on the Lagrange approach and the another which is based on the Lagrange-Euler technique. Some information is given in the following.

5.3 Lagrange Approach

A pure Lagrange method is based on the fact that the elements in a mesh represent the powder. The mesh moves as if it was the powder, and as the powder undergoes stress and strain the elements directly represent the result. The example of this technique assumes that the material is formed into a slab and given a velocity. This block then travels into the roll gap where it is compressed. Because the complexities of the powder behavior, this technique was applied to a thick metal sheet.

To prevent the effects of impact from the initial contact between powder and roll, the linear velocity of the powder is taken equal to the associated velocity of the roll.

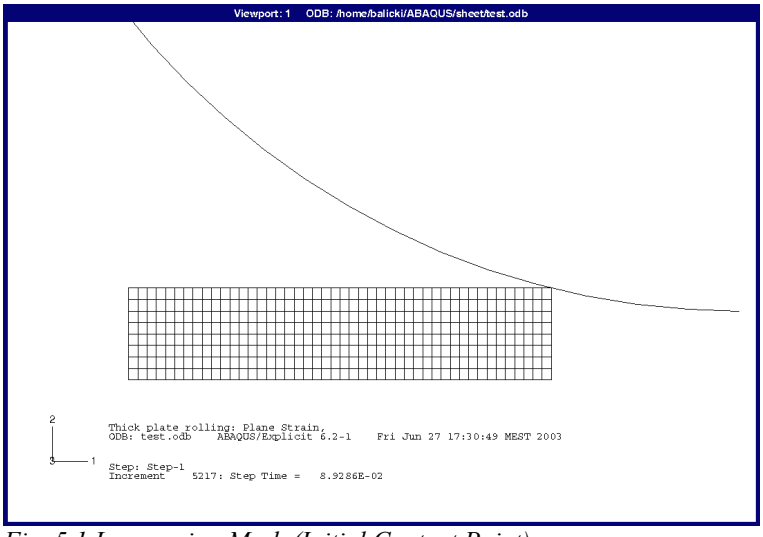


Fig. 5.1 Lagrangian Mesh (Initial Contact Point)

Following figure is the result of simulation of rolling of a thick metal sheet. It represents irreversible strains in the sheet after a steady state is reached.

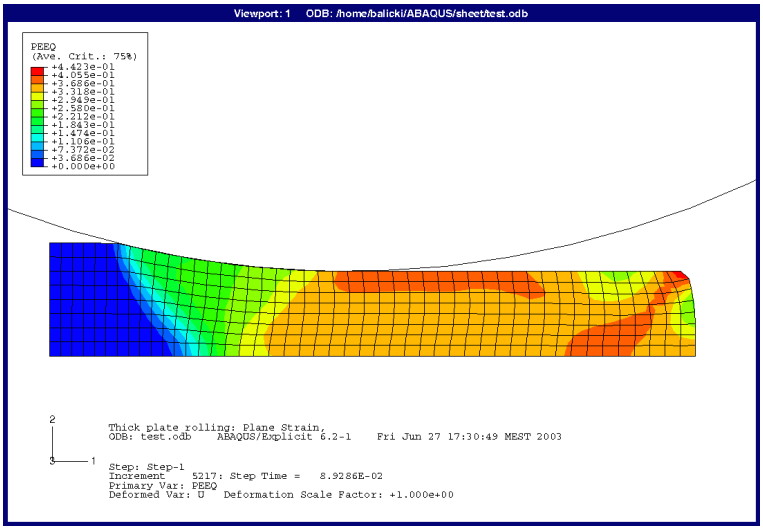


Fig. 5.2 Sheet Metal Rolling - Steady State Reached

This approach is not very applicable to powder compaction due to its inability to address severe mesh distortion exhibited by highly compressible materials. The following approach was investigated.

5.4 Euler-Lagrange Approach

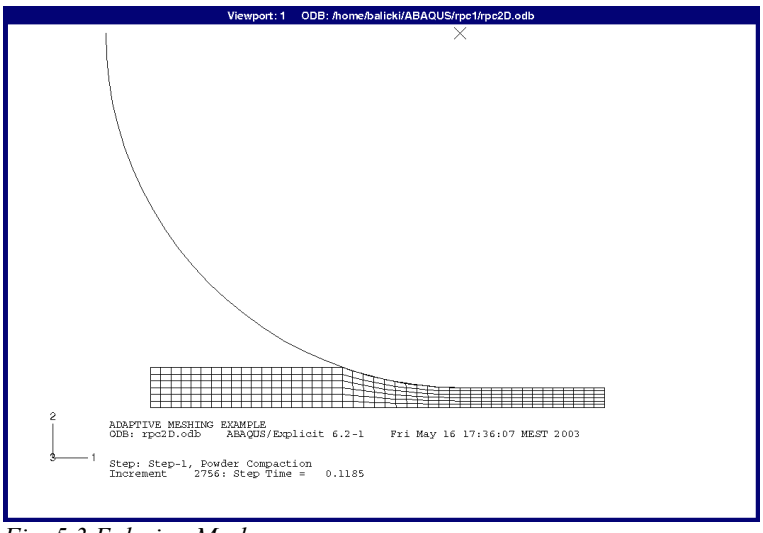


Fig. 5.3 Eulerian Mesh

Adaptive meshing is an ABAQUS/Explicit tool that makes it possible to maintain high-quality mesh through out an analysis, even when large deformations occur by allowing the mesh to move independently of the material [12]. This technique combines the features of the often used pure Lagrangian analysis in which the mesh represents the material and Eulerian analysis in which the mesh is fixed spatially while the material flows through it. This method is often referred to as the Arbitrary Lagrangian-Eulerian (ALE) analysis is extensively used in fluid flow analysis. Central to this method is the fact

that the mash is remapped when necessary to prevent the analysis from terminating as a result of severe mesh distortion.

Initially, the mesh is created representing the approximate space to be occupied by the powder traveling between the rolls. As the powder becomes compacted topologically similar mesh throughout the analysis is maintained: no elements are created or destroyed. However, the element shape and position may be altered.

The top surface of the mesh is set to move or slide freely (Lagrangian boundary) while the left and right surfaces are defined as inflow and outflow (Eulerian boundaries) to allow the material to flow through the mesh. The bottom of the mesh is set to be a symmetric boundary. The mesh follows the contour of the rolls inside the nip region as illustrated by the next figure.

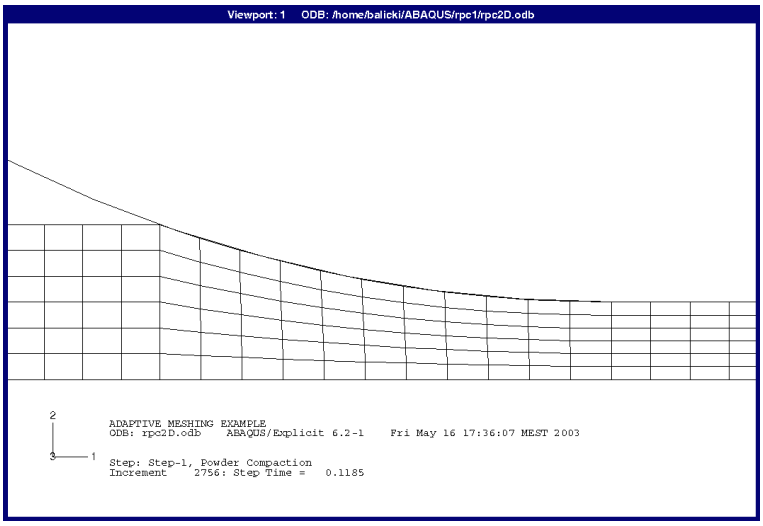


Fig. 5.4 Contact Area

A few investigative simulations were conducted. The next figure is an example of a failed run due to extreme distortion of the mesh causing the program to terminate prematurely. The initial point of powder-roll contact (nip angle) is the node farthest away from the gap that the roll surface is in contact with. This contact point may shift during the simulation.

When the model is not configured correctly it is clear that the simulation is not correct. However with investment of time and correct input data this approach may be successful.

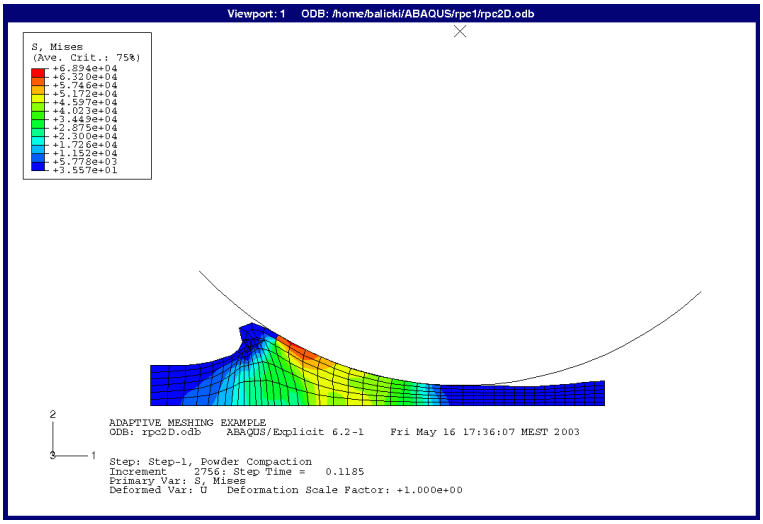


Fig. 5.5 Failed Simulation Attempt (Material = Powder)

5.5 Conclusion

These form of approach seems to be the most promising but not very practical. Sadder Kadiri, in his work on die compaction, has successfully utilized ABAQUS/Standard with Drucker-Prager / Cap powder behavior model.[25] It is believed that the same result can be attained with the explicit method.

According to literature [8], finite element method offers most attractive approach because incorporates complex information about geometry, powder behavior, and tribologic conditions. Although it requires expensive computer and software resources, it has promising ability to closely model powder rolling compaction, and as a result improve in process and equipment optimizing that may outweigh the costs of initial investment.

More experimental data is necessary to tweak the FEM model to match the real results. After the simulation is correct for a particular powder, a new powder should be introduced and examined in the same FEM model to see if the overall settings are good to determine which parameters are powder specific. Theoretically similar results may be attained for different combinations of input factors.

Once again the ability of this method to predict the operational parameters of a roll press heavily relies on the input information provided by the user from experimental data. The material behavior is at the center of this problem.

6 Conclusions

This report is an initial investigation into the nature of rolling compaction of powders in the theoretical research area. The Johanson model was utilized to gain general understanding of parameters involved and their behavior with system alteration. Two other approaches were investigated briefly to determine the direction of theoretical roll compaction research conducted at the *Centre Poudres et Procédés* at the *Ecole des Mines d'Albi-Carmaux*. It has been concluded that the finite element method is the most promising approach at the moment. Nevertheless, the slab method investigation should not be abandoned as it has potential to be more useful in the industrial setting. On a further note, these methods need to be explored in parallel with powder characterization studies considering that particular materials are at the center of the problematic issues in modeling their compaction in a roll press.

6.1 Process Remarks

The following are process remarks taken from experiments with Johanson model and literature. Initial feeding pressure has a definite effect on the final stress on the powder, moreover the applied stress heterogeneity is depended on the heterogeneity of feed pressure. For example, when using a screw feeder local pressure fluctuations are observed in the feeding zone corresponding to the period of the screw in sinusoidal fashion [3,5]. Moreover, fine powders have poor dearation ability especially while using pressure to feed the press. The air is squeezed between the particles and has difficulties to leave the mass of the powder being compacted which negatively influences feeding and therefore compaction exhibited by variation of density and strength of compact. Moreover there exists a danger of compact delamination or even explosion due to highly pressurized air bubbles escaping the compact in the exit region. In such case vacuum dearation system is suggested to possibility reduce these problems. Furthermore, there may exist a problem of powder leakage, where the loose material exits the space between the rolls. This uncompacted powder needs to be recycled for the process to be efficient [5].

As for process variables the gap size seems to be the most influential process parameter for it greatly changes the pressure exerted on the powder. Wall friction is another parameter that could be altered to adjust the compaction pressure by the use of lubricants or roughing of rolls. Roll velocity variation effects the roller minimally relative to the feed pressure as long as the compact is produced. With a constant feed pressure, for low roller speeds overcompaction occurs while for high speeds no strip is formed [1].

It is also important to mention that flow of powder in the nip region is not straight forward. The particles move at different speeds depending on their position in the compaction zone. This fact is often omitted by assumption made to simplify this flow to a constant movement of particles relative to the x axis [5].

According to Michrafy the density affects the wall friction as the powder is being compacted [14]. This friction is a function of density which is dependent on the pressing force which, in fact, is partially transmitted by the same contact friction. This circular relationship may be difficult to represent in a analytical approach and is more suitable for a numerical method. Moreover, this wall friction variation is not very easy to measure or represent. At the moment the behavior of this parameter is not verifiable in rolling compaction but seems to decrease in die compaction. In general, under densification internal friction angle increases and cohesion increases and wall friction decreases.

The other major issue in the rolling compaction area is the unavailability of accurate and thorough force/strain measurements techniques. For example the measurement of the pressure at the surface of the roll, or the pressure distribution along the axis of rotation are two major goals of rolling compaction research teams. Shear stresses on the wall surface are also quite difficult to measure due to residue buildup and heterogeneous feed pressure distribution. Generally, a hydraulic system is used to maintain the bearing blocks of a movable roller; such system is often susceptible to oscillations and irregularities complicating measurement acquisition and disrupting compact homogeneity. Not very well understood part of the process is the feeding zone where the stresses exerted by the rollers is very small, (ie. less than 0.1 MPa) and is not measurable by piezoelectric transducers placed on the roll surface [5]. Another parameter which is yet to be measured experimentally is the nip angle which is often used as a boundary condition. With out these quantitative results the accuracy of the modeling technique is only compared to the final compact and does not give us much knowledge about the phenomenon of powder behavior undergoing compaction in a roll press.

6.2 Powder Characterization

Before rolling compaction model is created a few issues need to be analyzed. One is the fact that currently there exist limited ability in defining the behavior of powders. All the models mentioned in this paper use many mechanical property theories of Jenike regarding the flow of bulk solids which are fairly good at the beginning of compaction and change significantly in compression zone especially when the material is closer to the solid state. However, this is all that is available at the moment and is applied as a result.

Internal angle of friction, bulk density and real density (using helium) are the few parameters believed to be fairly constant. Other parameters such as compressibility, contact friction are not so easily definable thus providing much of the headaches for the research community. In terms of internal friction and particle cohesion, there exist two major methods of attaining this information: Jenike shear tester and the annular ring shear tester. A preference is given the utilization of the ring (annular) cell shear tester rather than Jenike shear tester due its reliability and ease of use. As for measuring the friction coefficient between the powder and a contact surface both of the above mentioned tests can altered to take such data. It is difficult to estimate the quality of the information provided by these experiments for powder-wall surface behavior, but they are the only available quantitive source.

Another issue is the shear yield limit representation. There exists a number of different yield criterion but their usage varies due to their complexity and assumptions taken. Drucker-Prager/Cap model appears to be the most extensive in describing yield behavior of powder[13]. The data for this model is collected from diametrical compression, simple compression and compaction in an instrumental die [8].

The other end of the process, the final compact characterization, is also problematic. Unlike a tablet compact, the hardness testing is not easy for a slab of compact because a wide area with heterogeneous strength and density is observed whether a gravity feed hopper or a screw feeder are used. For this reason the compact needs to be tested along many points along the width and the length of the sheet but the most optimum positions are yet to be studied. Furthermore, the choice of hardness testing method is not clear either because standard tableting tensile strength test is generally no applicable, so a measurement such as the indentation test could be employed. Similarly, there exists very

little evidence for high quality density measurement.

6.3 Models

Rolling compaction of powders is not a simple problem as it was theoretically studied for over fort years and is still one of the major research topics in powders. Johanson was one of the first investigators into rolling compaction of non-metallic granular media; his work serves as the cornerstone of research in powder compaction field. It was chosen as the initial step in this study for its simplicity and recognition. In previous studies the model has been found moderately successful with good results for powders exhibiting low compressibility [11]. The model also gives insight into the general trends in parameter variation which can be used in estimation of minor system adjustments or comparison of compaction effects of same family powders.

The slab method is more complicated than the Johanson model because it can include more mechanical properties of the powder (cohesion, wall friction function, etc) and analyzes two principle stresses present. Currently for the method to be successful numerical differential equation solver is used due to high complexity of the governing equation. Schonert was able to successfully apply the slab method but his work only presents results for a very hard powder with low compressibility [10]. The difficulty of this approach arise in the application of yield criterion and number of factors included such as cohesion, friction, gravity, and inertia.

ABAQUS software is rather expensive and generally has a large learning curve. It may be a bit cumbersome to use it in an industrial setting, therefor a custom FEA program specific to the case of rolling compaction would be of best interest. However, this step is only reachable after a known general FEM modeling package is applied successfully to the problem. The advantage for creation of such tool is that for almost every powder a new mesh and new powder behavior need to be defined and automating or improving the current methods is desirable.

6.4 Future Considerations

Besides the Johanson's limited success, none of the above mentioned models have investigated the granular solid behavior in the region where the powder is mainly compacted by the feed pressure just prior to the compaction zone. The correct representation of powder in the this feeding zone needs to be investigated further because the results may improve the boundary conditions required for the models studied.

All the models discussed here assume Mohr-Coulomb friction which is quite a reliable method for classifying friction between solid materials but has not been studied much for granular solids. Wall friction in powders - solid interactions is believed to be related to internal friction, perhaps this type of approach could be beneficial in postulating boundary information.

Another technique that has not been discussed yet is the use of particle simulation where computer model of each powder particle in a bulk is created and accounted for through out compaction. This concept is rather difficult to achieve at the moment due to limitations of computing systems and little knowledge of the micro interactions between particles. This approach would incorporate the heterogeneous flow of particles in the nip region. This approach may be the leading method of simulating powder but that may not happen for ten or twenty years.

The models discussed may converge some day to simulate reality very closely; however, this will largely depend on our abilities to more precisely characterize granular solids.

7 Acknowledgments

Thank you to Abder Michrafy for being a great mentor and a friend. For shearing his mathematics knowledge and scientific research and presentation techniques.

Thank you to Sadder Kadiri for answering my long and speculative questions in the field of powder compaction and providing experimental data for my program.

Thank you to Stefan Haas for the long discussions about powders as well as philosophizing about social aspects of Europe and US.

Thanks to David Leach(Gary) for being a great mate, a source of comic relief and a motivating gym partner.

Thank you to Maxime for being a dear friend and a great neighbor with an oven.

Thank you to Maryse for being the most energetic and helpful administrative person I ever met.

8 References

[1] O. Simon, P. Guigon, *Interaction between Feeding and Compaction During Lactose Compaction in a Laboratory Roll Press*, PhD Thesis, Université de Technologie de Compiègne, (2000)

[2] O. Simon, P. Guigon "Correlation between powder-packing properties and roll press compact heterogeneity." *Powder Technology* 130 (2003) 257-264

[3] P. Sheskey, G. Sackett, L. Maher, K. Lentz, S. Tolle, J. Polli "Roll Compaction Granulation of a Controlled-Release Matrix Tablet Formulation Containing HPMC" *Pharmaceutical Technology* (1999)

[4] M.J. Rhodes, *Principles of Powder Technology*, John Wiley & Sons Ltd. (1990) p. 220

[5] P. Guigon, O. Simon "Roll press design-influence of force feed systems on compaction." *Powder Technology* 130 (2003) 41-48

[6] J.R. Johnson, "A rolling theory for granular solids," *ASME, Journal of Applied Mechanics* 32 : series E. No. 4, (1965)

[7] K. Sommer, G. Hauser, "Flow and compression properties of feed solids roll-type presses and extrusion press." *Powder Technology* 130 (2003) 272 -276

[8] R.T. Dec, A. Zavaliangos, J. C. Cunningham, "Comparison of various methods of powder compaction in roller press," *Powder Technology* 4642 (2002)

[9] V.P. Katashinskii, "Analytical determination of specific pressure during the rolling of metal powders," *Poroshkovaya Metallurgiya* (1996) 1-10

[10] K. Schonert , U. Sander, "Shear stresses and material slip in high pressure roller mills." *Powder Technology* 122 (2002) 136 -144

[11] Mise en forme des materials , calcul/elasticitè – French textbook.

[12] Abaqus 6.2 Manual

[13] A. Michrafy, D. Ringenbacher, P. Tchoreloff "Modeling the compaction behaviour of powders: application to pharmaceutical powders." *Powder Technology* 127 (2002) 257-266

[14] A. Michrafy, S.Kadiri, J. Dodds "Wall friction and its effects on the density distribution in the compaction of pharmaceutical excipients." To be published in *Chemical Engineering Research and Design.*

[15] M.J. Rhodes, *Principles of Powder Technology*, John Wiley & Sons Ltd. (1990) p. 100

[16] M.J. Rhodes, *Principles of Powder Technology*, John Wiley & Sons Ltd. (1990) p. 195

[17] K. Schonert , U. Sander, "Operational conditions of a screw-feeder-equipped high-pressure roller mill." *Powder Technology* 105 (1999) 282 -287

[18] M.J. Rhodes, *Principles of Powder Technology*, John Wiley & Sons Ltd. (1990) p. 215

[19] J.H. Tundermann, A.R.E. Singer "The flow of iron powder during roll compaction " Powder Metallurgy, (1968) Vol. 11, No. 22

[20] J.H. Tundermann, A.R.E. Singer "Deformation and densification during the rolling of metal powders," *Powder Metallurgy*, (1969) Vol. 12, No. 23

[21] G.Y. Tzou, "*Theoretical study on the cold sandwich sheet rolling considering Coulomb friction*." Yung Ta Institute of Technology (2001)

[22] H. Gao, S.C. Ramalingam; G.C. Barber, G. Chen "Analysis of symmetrical cold rolling with varying coefficients of friction." Journal of Materials Processing Technology 124 (2002) 178-182

[23] Matlab Online Manual version 6.1.0.450 Release 12.1
[24] G. Alderborn and C. Nystrom, *Pharmaceutical Powder Compaction Technology*, Marcel Dekker, Inc.(1996) p. 247
[25] S. Kadiri's ongoing thesis work on classical die compaction.
[26] S.P. Wang, S. Choudhry and T.B. Werheimer "Comparison between the static implicit and dynamic explicit methods for EFM simulation of sheet forming process." MARC Analysis Research Corporation, Palo Alto, CA., USA (1997)
[27] www.alexanderwerk.com - Manufacturer of small industrial machines and apparatus

9 Appendix

9.1 Powder Data

Avicel - Microcrystaline Cellulose
Wall Lubricant - Magnesium Stearate placed on the powder contact surface

	Avicel PH102	*Avicel PH102 + Lubricant*	*Avicel PH101*	**Lactose * Monohydrate**
Particle Size	90 mm	90 mm	50 mm	75 mm
Bulk Density	0.31 g/cm^3	0.31 g/cm^3	0.29 g/cm^3	0.55 g/cm^3
Internal Friction Angle	42°	42°	45°	39°
Wall Friction Coefficient μ (Angle)	0.28 (15.6°)	0.19 (11.4°)	0.29 (16.3°)	0.24 (13.5°)
Compressibility - K	4.75	4.75	3.64	7.7-8.0

* Taken from O. Simon thesis. [1]
All other data was taken from Kadiri's thesis work.[25]
Wall friction coefficient greatly depends on the type of material used in the rollers and is approximated to be the given values from available data.
Values for feed pressure are dependent on a powder.
For example, typical values, for feed pressure for lactose are are 0.04 - 0.06 Mpa [1] for a press with roll diameter of 130 mm

9.2 ABAQUS Source Code

The following figure represents node numbering used in creation of elements.

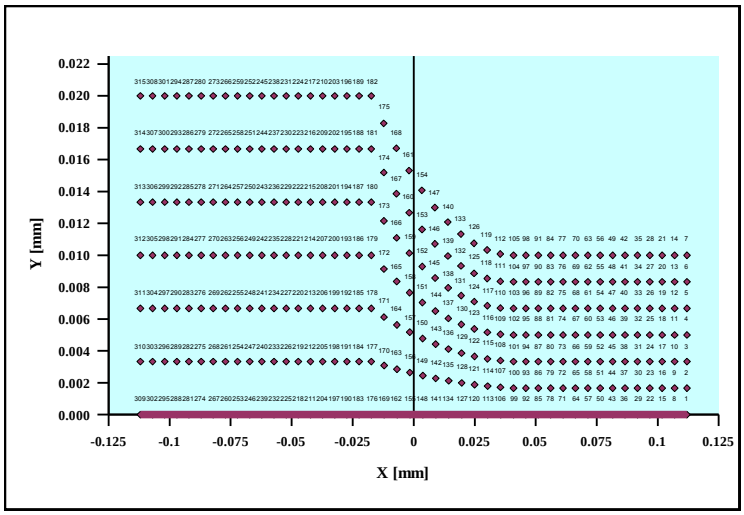


Fig. 9.1 Node Numbers

The next figure is a representation of dimension of the model.

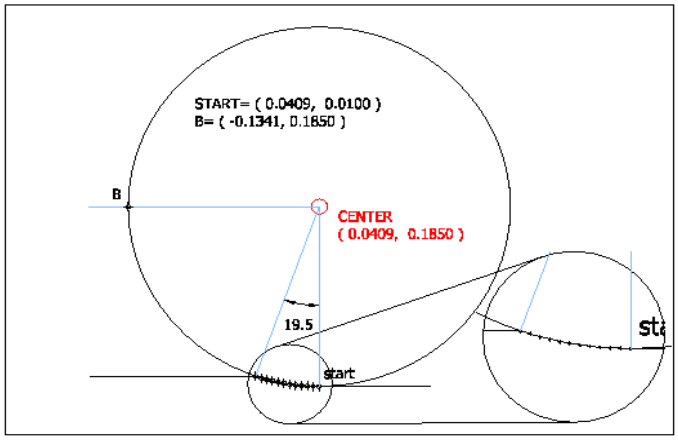


Fig. 9.2 Nip Angle

The source code for the latest ABAQUS/Explicit with ALE method. This example requires a node definition file from the included CD.

rpc2D.inp

```
*** Rolling Compactions of Powders
*** Ecole Des Mines D'Albi
*** May 2003
*** Marcin Balicki - balicki@enstimac.fr
**********************************************************
************************
*** Description:
***     First model using euler method - created by Abder Michrafy
***     Altered the definition of the roll
**********************************************************
************************
*** Notes :
***     Don't use blank lines,especially when the previous line ends
in ','
*HEADING
ADAPTIVE MESHING EXAMPLE
FLAT ROLLING - ADAPTIVE MESH, EULERIAN NODES AT
INLET AND OUTLET
Units - N, m
**********************************************************
************************
**The *RESTART, WRITE option is used to write the model
definition
**and state to the state (job-name.abq) and part (.prt) files.
** These files, which will for convenience be referred to as the
** "restart file," allow an analysis to be completed up to a certain
** point in a particular run and restarted and continued in a
** subsequent run with the *RESTART, READ option.
*RESTART,W,N=20
**W - restart file is to be written during the analysis.
**N - number of intervals during the step at which the *RESTART
output
** states are to be written.
**********************************************************
************************
**call the node definition file
*INCLUDE, INPUT=node2d.inp
**********************************************************
************************
**Master element
*ELEMENT, TYPE=CPE4R
1, 1, 2, 9, 8
** element #, a,b,c,d
** abcd represent the node numbers making up the 4 node element
**TYPE - defines the shape and behavior of the element
**Continuum elements begin with the letter "C."
**"PE," a plane strain element;
**"4R," quadrilateral
**********************************************************
************************
*ELGEN, ELSET=BAR
1,44,7,1,6,1,44
**This option is used to generate elements incrementally.
** the element set name is BAR
**1. Master element #,
**2. Number of elements to be defined in the first row generated.
**3. Increment in node numbers of corresponding nodes from
element to element
**     in the row.
**4. Increment in element numbers in the row. The default is 1.
**If necessary, copy this newly created master row to define a
layer of
**    elements.
**5. Number of rows to be defined, including the master row. The
default is 1.
**6. Increment in node numbers of corresponding nodes from row
to row.
**7. Increment in element numbers of corresponding elements
from row to row.
**If necessary, copy this newly created master layer to define a
block of
**     elements.
**8. Number of layers to be defined, including the master layer.
The
**    default is 1.
**9. Increment in node numbers of corresponding nodes from
layer to layer.
**10. Increment in element numbers of corresponding elements
from layer
**    to layer.
**Repeat this data line as often as necessary. Each line will
generate
**     N1 ´ N2 ´ N3 elements,
**where N1 is the number of elements in a row, N2 is the number
of rows
**    in a layer, and N3 is the number of layers.
**********************************************************
************************
**This option is used to define properties of solid (continuum)
elements,
**infinite elements, and truss elements.
*SOLID SECTION,ELSET=BAR,MATERIAL=POUDRE
5.E-2,
**The argument depends on element type
**ie. if plane strain is used the argument is the element thickness
**
**********************************************************
************************
*** MATERIAL DEFINITION
***
**Definition of material used in the SOLID SECTION
*MATERIAL,NAME=C15
**these can change with respect to field variables (ie temperature)
*ELASTIC
1.5E11,.3
**Young's Modulus, Poisson's ratio
**
**
**Plasticity given must be true stress and true strain
**use ABAQUS manual to convert from nominal stress and strain.
**usually found in material guides.
*PLASTIC
**Yield Stress, plastic strain
168.72E06,0
219.33E06,0.1
272.02E06,0.2
308.53E06,0.3
337.37E06,0.4
361.58E06,0.5
382.65E06,0.6
401.42E06,0.7
418.42E06,0.8
434.01E06,0.9
448.45E06,1.0
**
*DENSITY
7.85E3,
** Example of druker prager cap model for material
** SOIL STRATA D1 (700 ft - 1100 ft)
** ------------------------------
*MATERIAL,NAME=D1
*ELASTIC
 47700.,.17
*CAP PLASTICITY
 200.,36.9,0.33,0.02,0.,1.
```

```
*CAP HARDENING
 400.,.0
 600.,.02
 800.,.05
 900.,.09
**CAP CREEP, LAW=SINGHM,
MECHANISM=CONSOLIDATION
** 2.2e-7, 2.1e-2, 1.0, 1.0, 50.0
** 3.5e-5, 2.1e-2, 1.0, 1.0, 212.0
***PERMEABILITY,SPECIFIC=3.07e-2
** 16.8,,0
** 32.0,,250
*EXPANSION
 0.32E-05,
*DENSITY
7.85E3,
**********************************************************
**********************
***
*MATERIAL, NAME=POUDRE
** Proprietes elastiques
**   E          nu      RhoR
*ELASTIC, TYPE=ISOTROPIC, dependencies=1
0.1134E+9,0.28, ,0.47
0.3028E+9,0.28, ,0.56
0.5453E+9,0.28, ,0.63
1.0146E+9,0.28, ,0.72
1.7871E+9,0.28, ,0.82
*CAP PLASTICITY
** Parametres de la courbe intrinseque
**  d      beta(°)      R    epvol0   alpha   K
0.00014E+6 , 36.82  , 0.558 , 0.  , 0.03  , 1.
*CAP HARDENING
** Comportement en durcissement
**    P        eps(vol)pl
0.00013E+6 , 0.
9.086E+6, 0.915
19.47E+6, 1.122
30.54E+6, 1.239
40.15E+6, 1.330
54.75E+6, 1.379
66.95E+6, 1.426
***USER DEFINED FIELD
***DEPVAR
**1
*DENSITY
7.85,
**********************************************************
**********************
***  VARIABLES - NODES/ NODE SETS
***
**create a node for defining the roll
*NODE,NSET=REF
10000,  0.0409 ,  0.185
**#,x,y
**********************************************************
**********************
***
***This option assigns nodes to a node set ="name"
*NSET,NSET=BOT,GEN
1,309,7
** a,b,c
**a. First node in the set.
**b. Last node in the set.
**c. Increment in node numbers between nodes in the set.
*NSET,NSET=TOP,GEN
7, 315, 7
*NSET,NSET=BACK,GEN
309, 315, 1
*NSET,NSET=BACK2,GEN
310, 315, 1
*NSET,NSET=FRONT,GEN
1, 7, 1
*NSET,NSET=EULER
BACK, FRONT
** 2 node set labels to be assigned to this node set.
**
***   The back fo the slab
*NSET,NSET=EQN1,GEN
310, 315, 1
***    Front of the slab
*NSET,NSET=EQN2,GEN
2, 7, 1
**********************************************************
**********************
***
***                        Degres de liberte contraints
***
*EQUATION
2,
EQN1,1,1.0,309,1,-1.0
*EQUATION
2,
EQN2,1,1.0,1,1,-1.0
***
*ELSET, ELSET=EULER1, GEN
1, 221, 44
*ELSET, ELSET=EULER2, GEN
44, 264, 44
*ELSET,ELSET=TOP,GEN
221, 264, 1
*ELSET,ELSET=BOT,GEN
1, 44, 1
***
**The roller is rotated at a constant angular velocity of 1
revolution per
**second (6.28 rad/sec).With roll radius at .175 that yeilds a roller
surface
**speed of 1.1 m/sec. The plate is given an initial velocity in the
global
**x-direction. The initial velocity is chosen to match the x-
component
** of velocity of the roller at the point of first contact. (19.5°) cos
(x)*W
** This choice of initial velocity results in a net acceleration of
zero
** in the x-direction at the point of contact and minimizes the
initial
** impact between the plate and the roller.
** This minimizes the initial transient disturbance
*INITIAL CONDITIONS,TYPE=VELOCITY
BAR,1,1.0369
*BOUNDARY
BOT, 2, 2
*SURFACE, NAME=SURF1, REGION TYPE=SLIDING
TOP,S2
*SURFACE, NAME=EULER1, REGION TYPE=EULERIAN
EULER1,S1
*SURFACE, NAME=EULER2, REGION TYPE=EULERIAN
EULER2,S3
**********************************************************
**********************
**roller definition
**arc starts at START and finishes at FIRST PAIR
**of circle elements, second pair is the center of the circle
**circle is drawn in a counterclockwise direction.
**relative rigidity of the roller is high so it is completely rigid
*SURFACE,TYPE=SEGMENTS,NAME=RIGID,FILLET
RADIUS=.001
START,   0.040900,   0.010000
CIRCL,   -.134100,   0.185000 ,  0.0409 ,  0.185
*RIGID BODY, REFNODE=10000, ANALYTICAL SURFACE =
RIGID
**********************************************************
**********************
** start of a step arguments are the same as header
```

```
*STEP
Powder Compaction
**
**the type of simulation - Dynamic
**Include EXPLICIT parameter to specify explicit time
integration.
**can be used to perform quasi-static analyses with complicated
**contact conditions; and allows for either automatic or fixed time
**incrementation to be used--by default
*DYNAMIC,EXPLICIT
,0.225
**the argument defines overrides default time of process.(ie .2)
**
**********************************************************
**********************
**Prescribing boundary conditions at nodes
**The roller is rotated at a constant angular velocity of 1
revolution per
**second (6.28 rad/sec).With roll radius at .175.
**Boundary conditions are applied to those parts of the model
where the
** displacements are known. Such parts may be constrained to
remain fixed
** (have zero displacement) during the simulation or may have
specified,
** nonzero displacements. In either situation the constraints are
applied
** directly to the  nodes of the model.
** <node number>, <first dof>, <last dof>, <magnitude of
displacement>
** 1=X; 2=Y; 3=Z;4=rot X;5=rot Y;=rot Z;
** Boundary conditions on a node are cumulative
** The node 10000 = ROLL -> set DOF# 1 through 5 to 0
*BOUNDARY
10000,1,5
**
** Set the rotation of the ROLL about Z (DOF #6) axis to velocity
V
*BOUNDARY,TYPE=VELOCITY
10000,6,6,6.2832
**
**the back is constrained only in the Y (2) direction
**DOF to constrain are 2 to 2 (=only 2)and displacement in Y
direction is 0
**review =EULERIAN
*BOUNDARY,TYPE=VELOCITY,REGION TYPE=EULERIAN
BACK2,2,2,0.0
**
**********************************************************
**********************
** Mass scaling is often used for computational efficiency in
quasi-static
** analyses and in some dynamic analyses that contain a few very
small
** elements that control the stable time increment.
** Mass scaling is an alternative to increasing the loading rate.
** When using rate-dependent materials, mass scaling is
preferable
** because increasing the loading rate artificially changes the
** material properties.
*FIXED MASS SCALING, FACTOR=27000.0
**********************************************************
**********************
*SURFACE INTERACTION,NAME=FRICT
*FRICTION
 0.3,
*CONTACT PAIR,INTERACTION=FRICT, CPSET=CONTACT
SURF1,RIGID
**
**********************************************************
**********************
*FILE OUTPUT, NUMBER INTERVAL=10,
TIMEMARKS=YES
*EL FILE, ELSET=TOP
 PEEQ,
*NODE FILE,NSET=REF
 U,RF
*OUTPUT, FIELD,NUMBER INTERVAL=10
*ELEMENT OUTPUT
PEEQ,S
*NODE OUTPUT
U,V
*CONTACT OUTPUT, CPSET=CONTACT, VARIABLE=ALL
*OUTPUT,HISTORY,TIME INTERVAL=1.E-4
*NODE OUTPUT,NSET=REF
RF2
***--------------------
*NSET, NSET=QA_TEST_REFN
REF,
*ELSET, ELSET=QA_TEST
TOP,
*NSET, NSET=QA_TEST
BOT,
*OUTPUT, FIELD, TIME MARKS=YES, NUMBER
INTERVAL=10
*ELEMENT OUTPUT, ELSET=QA_TEST
PEEQ,
*NODE OUTPUT, NSET=QA_TEST
U,
*NODE OUTPUT, NSET=QA_TEST_REFN
RF,
*OUTPUT, HIST, FREQ=9999
*ENERGY OUTPUT, VAR=PRESELECT
******
*ADAPTIVE MESH,ELSET=BAR,FREQUENCY=5
*ADAPTIVE MESH CONSTRAINT
EULER,1,1,0.
BACK,2,2,0.
*ENDSTEP
```

9.3 Johanson Model - Matlab Program

Source Code Included

CD with all source files (nip.m, nip.fig)

The program was developed on a Solaris machine.
Graphics may be different on other platforms.

Required files:

nip.m	main program
nip.fig	the graphical user interface for this program
SlipStickFun.m	two functions representing the state of the powder
slipODE.m	ODE for the powder while slip occurs
forceFactor.m	force factor equation
torqueFactor.m	torque factor equation
modaldlg.m	help menu
modaldlg.fig	help menu graphical user interface
johansonModel.jpg	image of the system

Run Instructions:

All the files should be located in the same directory.

Once Matlab is up and running there are two methods of stating the program:

a) Click on File Open, choose the directory containing required files, click on nip.m. This will open a Graphical User Interface (GUI) similar to the one in figure 9.3.

b) Type nip on the Matlab command line. If the local path is the directory containing the required files a GUI will appear.

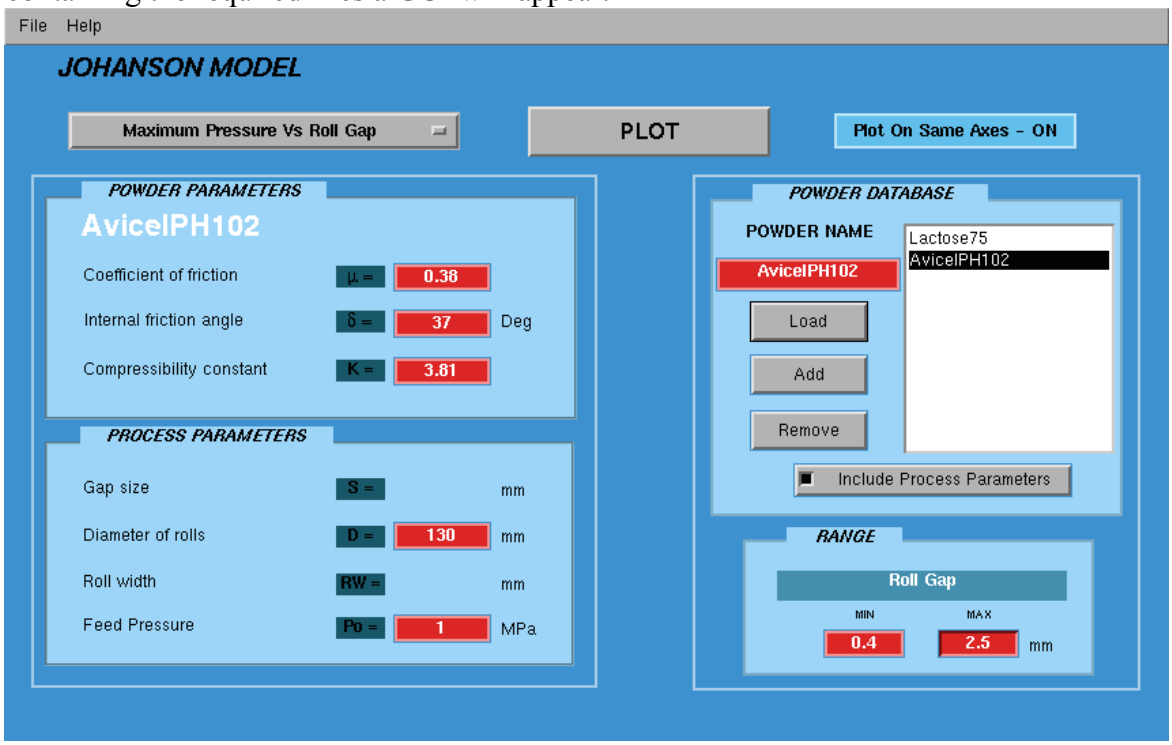


Fig. 9.3Main User Interface

Once the GUI is running

A pull-down menu contain file operations and help.

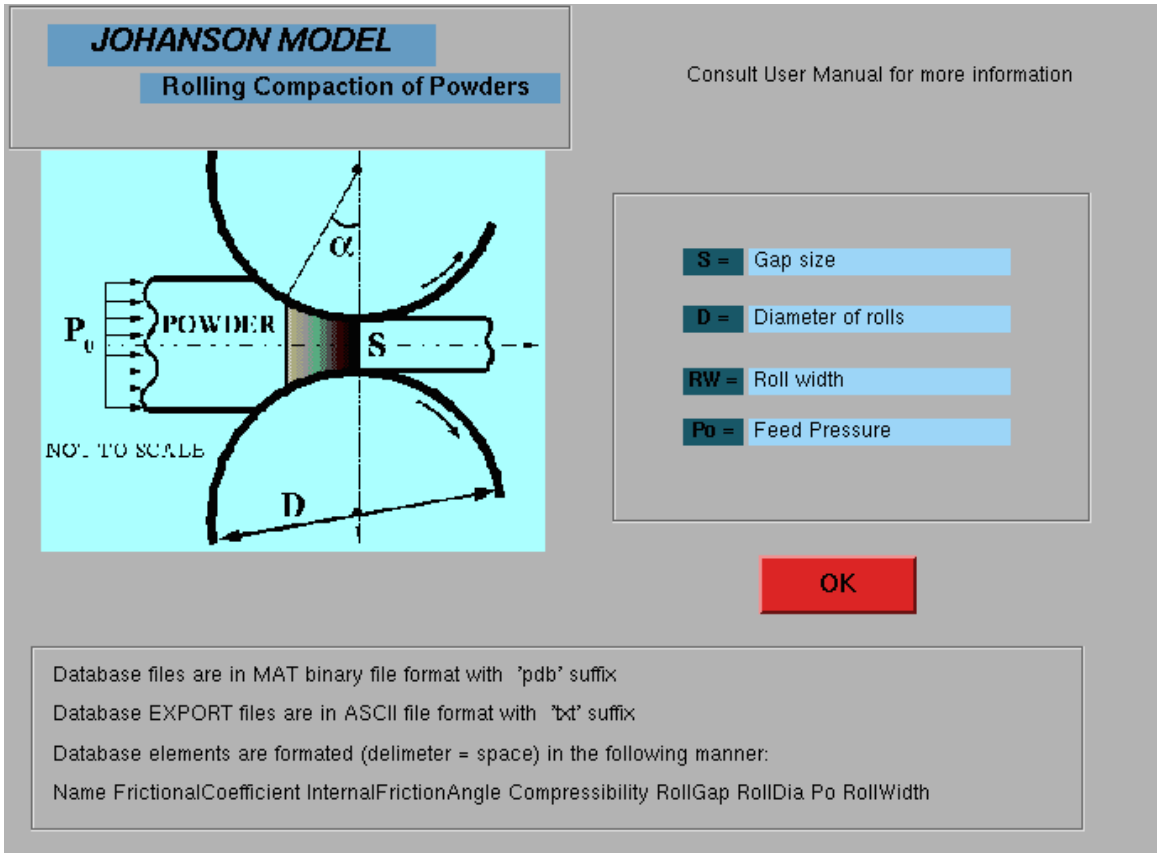


Fig. 9.4 Help Screen

MATLAB 6.1 SOURCE CODE

nip.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%     Johanson Model
%%     Date May 6, 2003
%%     Marcin Balicki
%%     Ecole Des Mines D'Albi
%%     balicki@enstimac.fr
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Description:
%% Application of Johanson Model of rolling Compaction of
powder.
%% Allows for quick analysis effects of parameter variation on
%% powder compaction
%% It provides a means of creating and altering databases of
%% powder properties and process parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Note:
%% The program was developed on a Solaris machine
%% Graphics may be different on other platforms

%%Required files:
%% nip.m           This file - main program
%% nip.fig         the graphical user interface for this program
%% SlipStickFun.m   two functions representing the state of the
powder
%% slipODE.m        ODE for the powder while slip occurs
%% forceFactor.m    force factor equation
%% torqueFactor.m   torque factor equation
%% modaldlg.m       help menu
%% modaldlg.fig     help menu graphical user interface
%% johansonModel.jpg    image of the system
%%%%%%%%%
% To DO:
% Option for plotting in main GUI axes or separate figure
% Warning 'save on exit'
%%%%%%%%
%Menu Options
%The program saves and imports powder database which is saved
as a matlab file
%with '.pdb' suffix
%The program can export the database as a text file as well.
%The print command prints the user interface, some user
adjustments are required.
%%%%%%%%
%All the graphs are plotted in a new figure unless the %plot in
same figure
% button
%is selected in which case consecutive plots of the same type will
be plotted in
%one figure
%%%%%%%%
%%%%%%%%
% user input
% S=Gap size(between rolls)
% D=Diameter of Rolls
% K=Compressibility constant
% d=delta - internal friction angle of powder
% Po=initial pressure at the entrance (perpendicular to major
principal stress
% W=phi-wall (roll-powder) friction angle (user enters friction
coefficient -> tan(phi)=mu)
% v=acute angle between the tangent to the roll surface and the
direction of the major principal stress
%     can be calculated using W and d
% RW=roll width
% rMin = min range value
% rMax = max range value
%%%%%%
function varargout = nip(varargin)
% NIP Application M-file for nip.fig
%    FIG = NIP launch nip GUI.
%    NIP('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.0 07-May-2003 16:37:58

if nargin == 0  % LAUNCH GUI
   disp(' ')
   disp('---------------------------------------------')
   disp('Powder Rolling Compaction  - Johanson method')
   disp('Marcin Balicki')
   disp('Ecole des Mines d Albi - 2003')
   disp('---------------------------------------------')
   disp(' ')

   fig = openfig(mfilename,'reuse');
   set(fig,'Tag','Main');
            % Generate a structure of handles to pass to callbacks,
and store it.
            handles = guihandles(fig);
            guidata(fig, handles);
   %note the current DB file in use (used in saving)
   handles.LastFile='';
   %create default powder databse
   handles.PowderDB(1).Name='Lactose75';
   handles.PowderDB(1).W='0.24';
   handles.PowderDB(1).d='39';
   handles.PowderDB(1).K='7.75';
   handles.PowderDB(1).S='1';
   handles.PowderDB(1).D='130';
   handles.PowderDB(1).Po='0.06';
   handles.PowderDB(1).RW='50';
   %create default powder databse
   handles.PowderDB(2).Name='AvicelPH102';
   handles.PowderDB(2).W='0.28';
   handles.PowderDB(2).d='42';
   handles.PowderDB(2).K='4.75';
   handles.PowderDB(2).S='1';
   handles.PowderDB(2).D='130';
   handles.PowderDB(2).Po='0.06';
   handles.PowderDB(2).RW='50';
   %create default powder databse
   handles.PowderDB(3).Name='AvicelPH102L';
   handles.PowderDB(3).W='0.19';
   handles.PowderDB(3).d='42';
   handles.PowderDB(3).K='4.75';
   handles.PowderDB(3).S='1';
   handles.PowderDB(3).D='130';
   handles.PowderDB(3).Po='0.06';
   handles.PowderDB(3).RW='50';
   %create default powder databse
   handles.PowderDB(4).Name='AvicelPH101';
   handles.PowderDB(4).W='0.29';
   handles.PowderDB(4).d='45';
   handles.PowderDB(4).K='3.64';
   handles.PowderDB(4).S='1';
   handles.PowderDB(4).D='130';
   handles.PowderDB(4).Po='0.06';
   handles.PowderDB(4).RW='50';
   %load Avicel powder from the database
   LoadPowder(handles, 2);
   %Update powder list that is in the database
   UpdateList(handles);

  %Set default properties of the plot
  set(0,'DefaultLineLineWidth',3);
  set(0,'DefaultAxesLineWidth',3);
  set(0,'DefaultAxesFontSize',16)
  set(0,'DefaultTextFontSize',14)
            if nargout > 0
```

```
                varargout{1} = fig;
        end
    guidata(fig,handles); % store the changes into the guihandle
structure
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION
OR CALLBACK

        try
                if (nargout)
                        [varargout{1:nargout}] =
feval(varargin{:}); % FEVAL switchyard
                else
                        feval(varargin{:}); % FEVAL
switchyard
                end
        catch
                disp(lasterr);
        end

end


%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this
file, and
%| sets objects' callback properties to call them through the
FEVAL
%| switchyard above. This comment describes that mechanism.
%|
%| Each callback subfunction declaration has the following form:
%| <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES,
VARARGIN)
%|
%| The subfunction name is composed using the object's Tag and
the
%| callback type separated by '_', e.g. 'slider2_Callback',
%| 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.
%|
%| H is the callback object's handle (obtained using GCBO).
%|
%| EVENTDATA is empty, but reserved for future use.
%|
%| HANDLES is a structure containing handles of components in
GUI using
%| tags as fieldnames, e.g. handles.figure1, handles.slider2. This
%| structure is created at GUI startup using GUIHANDLES and
stored in
%| the figure's application data using GUIDATA. A copy of the
structure
%| is passed to each callback.  You can store additional
information in
%| this structure at GUI startup, and you can change the structure
%| during callbacks.  Call guidata(h, handles) after changing your
%| copy to replace the stored original so that subsequent callbacks
see
%| the updates. Type "help guihandles" and "help guidata" for
more
%| information.
%|
%| VARARGIN contains any extra arguments you have passed to
the
%| callback. Specify the extra arguments by editing the callback
%| property in the inspector. By default, GUIDE sets the property
to:
%| <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [],
guidata(gcbo))
%| Add any extra arguments after the last argument, before the
final
%| closing parenthesis.


% --------------------------------------------------------------------
%most of the following just check if the user input is correct
% --------------------------------------------------------------------
function varargout = rMin_Callback(h, eventdata, handles,
varargin)
user_entry = str2double(get(h,'string'));
if isnan(user_entry)
   errordlg('You must enter a numeric value','Bad Input','modal')
end
% --------------------------------------------------------------------
function varargout = rMax_Callback(h, eventdata, handles,
varargin)

user_entry = str2double(get(h,'string'));
if isnan(user_entry)
   errordlg('You must enter a numeric value','Bad Input','modal')
end
% --------------------------------------------------------------------
function varargout = varS_Callback(h, eventdata, handles,
varargin)
user_entry = str2double(get(h,'string'));
if isnan(user_entry)
   errordlg('You must enter a numeric value','Bad Input','modal')
end
% --------------------------------------------------------------------
function varargout = varD_Callback(h, eventdata, handles,
varargin)
user_entry = str2double(get(h,'string'));
if isnan(user_entry)
   errordlg('You must enter a numeric value','Bad Input','modal')
end
% --------------------------------------------------------------------
function varargout = varK_Callback(h, eventdata, handles,
varargin)
user_entry = str2double(get(h,'string'));
if isnan(user_entry)
   errordlg('You must enter a numeric value','Bad Input','modal')
end
% --------------------------------------------------------------------
function varargout = vard_Callback(h, eventdata, handles,
varargin)
user_entry = str2double(get(h,'string'));
if isnan(user_entry)
   errordlg('You must enter a numeric value','Bad Input','modal')
end
% --------------------------------------------------------------------
function varargout = varPo_Callback(h, eventdata, handles,
varargin)
user_entry = str2double(get(h,'string'));
if isnan(user_entry)
   errordlg('You must enter a numeric value','Bad Input','modal')
end
% --------------------------------------------------------------------
function varargout = varW_Callback(h, eventdata, handles,
varargin)
user_entry = str2double(get(h,'string'));
if isnan(user_entry)
   errordlg('You must enter a numeric value','Bad Input','modal')
end
% --------------------------------------------------------------------
function varargout = varRW_Callback(h, eventdata, handles,
varargin)
user_entry = str2double(get(h,'string'));
if isnan(user_entry)
   errordlg('You must enter a numeric value','Bad Input','modal')
end
% --------------------------------------------------------------------
function varargout = PopMenu_Callback(h, eventdata, handles,
varargin)
%change the required variable visibility
updateVariableVisibility(handles)

% --------------------------------------------------------------------
function varargout = CheckPlot_Callback(h, eventdata, handles,
varargin)
```

```
%Displays the Plot On Same Axes status
if get(h,'Value')==1
    set(h,'String', 'Plot On Same Axes - ON');
    set(h,'BackgroundColor', [0.38,0.75,0.92]);
else
    set(h,'String', 'Plot On Same Axes - OFF');
    set(h,'BackgroundColor', [0.70,0.70,0.70]);
end

% --------------------------------------------------------------------
function fig=newFigure(handles,figName)
%returns the handle to the figure used for plotting
%function opens a new figure for the plot
%can also be made to use the axes of the gui
%by checking the state of a button
%but this would not allow editing of graphs

%if no figures exist create one
if get(0,'Children')== 0;
    fig=figure('name',figName);
%but if they exist and plot in same figure is off
%create a new figure with the name passed in to the function
elseif get(handles.CheckPlot,'Value') == 0;
    fig=figure('name',figName);
    hold off    %shut off same axes plotting
%If figure exist and the plot should be on same axes
else
    %get all the figures with figName
    [flag,figs] = figflag(figName);
    %if none then create one
    if flag==0;
        fig=figure('name',figName);
    %if they exist then make one of them the current one
    else
        fig=figs(size(figs,2)); %get the last figure that matches the
figName
        hold on %and force it to plot on the same axes.
    end
end

% --------------------------------------------------------------------
function updateVariableVisibility(handles)
% Determine the current choice and change the input fields
visibility
% so the user can only enter the required fields
%pop_Names = get(handles.PopMenu,'Value')  %returns all
choices
%returns the number of the choice (first position = 1)
%set all to off
%W is the friciton coefficient
set([handles.varW,
    handles.varS,
    handles.varD,
    handles.varK,
    handles.vard,
    handles.varPo,
    handles.varRW],'Visible', 'off');

    set(handles.rUnit,'String','');
    set(handles.rVar,'String','');
    set(handles.rMin,'String','0');
    set(handles.rMax,'String','100');

pop_Selected = get(handles.PopMenu,'Value');
switch pop_Selected
  case 1
      %nothing selected turn all on
    set([handles.varW,
    handles.varS,
    handles.varD,
    handles.varK,
    handles.vard,
    handles.varPo,
 handles.varRW],'Visible', 'on');
 set(handles.rUnit,'String','');
 set(handles.rVar,'String','');
 set(handles.rMin,'String','0');
 set(handles.rMax,'String','100');

case 2
    %nipAngle
 set([handles.varW,
 handles.varS,
 handles.varD,
 handles.varK,
 handles.vard],'Visible', 'on');
 set(handles.rUnit,'String','Deg');
 set(handles.rVar,'String','Angular Position');
 set(handles.rMin,'String','0');
 set(handles.rMax,'String','60');
case 3
    % Nip Angle Vs Internal  Friction Angle
 set([handles.varW,
 handles.varS,
 handles.varD,
 handles.varK],'Visible', 'on');
 set(handles.rUnit,'String','Deg');
 set(handles.rVar,'String','Internal Friction Angle');
 set(handles.rMin,'String','20');
 set(handles.rMax,'String','70');
case 4
  % Nip Angle Vs Wall Friction
 set([handles.varS,
 handles.varD,
 handles.varK,
 handles.vard],'Visible', 'on');
 set(handles.rUnit,'String','');
 set(handles.rVar,'String','Friction Coefficient');
 set(handles.rMin,'String','0.1');
 set(handles.rMax,'String','0.8');
case 5
    % Nip Angle Vs Compressibility
  set([handles.varW,
 handles.varS,
 handles.varD,
 handles.vard],'Visible', 'on');
 set(handles.rUnit,'String','');
 set(handles.rVar,'String','Compressibility (K)');
 set(handles.rMin,'String','1');
 set(handles.rMax,'String','12');
case 6
    % Nip Angle Vs Roll Gap
 set([handles.varW,
 handles.varD,
 handles.varK,
 handles.vard],'Visible', 'on');
 set(handles.rUnit,'String','mm');
 set(handles.rVar,'String','Roll Gap');
 set(handles.rMin,'String','0.4');
 set(handles.rMax,'String','2.5');
case 7
    % Pressure Distribution in Nip Region
  set([handles.varW,
 handles.varS,
 handles.varD,
 handles.varK,
 handles.vard,
 handles.varPo],'Visible', 'on');
 set(handles.rUnit,'String','Deg');
 set(handles.rVar,'String','Angular Postition');
 set(handles.rMin,'String','0');
 set(handles.rMax,'String','10');
case 8
    % Maximum Pressure Vs Nip Angle
   set([handles.varW,
 handles.varS,
```

```
 handles.varD,
 handles.varK,
 handles.vard,
 handles.varPo],'Visible', 'on');
 set(handles.rUnit,'String','Deg');
 set(handles.rVar,'String','Nip Angle');
 set(handles.rMin,'String','7');
 set(handles.rMax,'String','12');
case 9
   % Maximum Pressure Vs Po
  set([handles.varW,
 handles.varS,
 handles.varD,
 handles.varK,
 handles.vard],'Visible', 'on');
 set(handles.rUnit,'String','MPa');
 set(handles.rVar,'String','Feed Pressure');
 set(handles.rMin,'String','0');
 set(handles.rMax,'String','2');
case 10
   % Maximum Pressure Vs Internal Friction Angle
 set([handles.varW,
 handles.varS,
 handles.varD,
 handles.varK,
 handles.varPo],'Visible', 'on');
 set(handles.rUnit,'String','Deg');
 set(handles.rVar,'String','Internal Friction Angle');
 set(handles.rMin,'String','10');
 set(handles.rMax,'String','60');
case 11
  % Maximum Pressure Vs Wall Friction
 set([handles.varS,
 handles.varD,
 handles.varK,
 handles.vard,
 handles.varPo],'Visible', 'on');
 set(handles.rUnit,'String','');
 set(handles.rVar,'String','Friction Coefficient');
 set(handles.rMin,'String','0.1');
 set(handles.rMax,'String','0.8');
case 12
   % Maximum Pressure Vs Roll Gap
 set([handles.varW,
 handles.varD,
 handles.varK,
 handles.vard,
 handles.varPo],'Visible', 'on');
 set(handles.rUnit,'String','mm');
 set(handles.rVar,'String','Roll Gap');
 set(handles.rMin,'String','0.4');
 set(handles.rMax,'String','2.5');
case 13
   % Maximum Pressure Vs Compressibility
 set([handles.varW,
 handles.varS,
 handles.varD,
 handles.vard,
 handles.varPo],'Visible', 'on');
 set(handles.rUnit,'String','');
 set(handles.rVar,'String','Compressibility (K)');
 set(handles.rMin,'String','1');
 set(handles.rMax,'String','12');
case 14
   % Roll Force Vs Roll Gap
 set([handles.varW,
 handles.varD,
 handles.varK,
 handles.vard,
 handles.varPo,
 handles.varRW],'Visible', 'on');
 set(handles.rUnit,'String','mm');
 set(handles.rVar,'String','Roll Gap');
  set(handles.rMin,'String','0.4');
  set(handles.rMax,'String','2.5');
 case 15
    % Roll Torque Vs RollGap
  set([handles.varW,
  handles.varD,
  handles.varK,
  handles.vard,
  handles.varPo,
  handles.varRW],'Visible', 'on');
  set(handles.rUnit,'String','mm');
  set(handles.rVar,'String','Roll Gap');
  set(handles.rMin,'String','0.4');
  set(handles.rMax,'String','2.5');
 otherwise
   disp('Unknown method.')
end

% --------------------------------------------------------------------
function varargout = CalcButton_Callback(h, eventdata, handles,
varargin)
%runs the chosen ploting function
%returns the number of the choice (first position = 1)
pop_Selected = get(handles.PopMenu,'Value');
%After the calculate button was pressed the pop menu value is
retrieved
%and appropriate function is called
switch pop_Selected
  case 1
   disp('CHOOSE PLOT')
  case 2
   NipAngle(handles)
  case 3
   NipAngleVsInternalFrictionAngle(handles)
  case 4
   NipAngleVsWallFriction(handles)
  case 5
   NipAngleVsCompressibility(handles)
  case 6
   NipAngleVsRollGap(handles)
  case 7
   PressureDistribution(handles)
  case 8
   MaximumPressureVsNipAngle(handles)
  case 9
   MaximumPressureVsPo(handles)
  case 10
   MaximumPressureVsInternalfFrictionAngle(handles)
  case 11
   MaximumPressureVsWallFriction(handles)
  case 12
   MaximumPressureVsRollGap(handles)
  case 13
   MaximumPressureVsCompressibility(handles)
  case 14
   RollForceVsRollGap(handles)
  case 15
   RollTorqueVsRollGap(handles)
  otherwise
   disp('Unknown method.')
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Johanson Model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Nip Angle
% Nip Angle Vs Internal  Friction Angle
% Nip Angle Vs Wall Friction
% Nip Angle Vs Compressibility
% Nip Angle Vs Roll Gap
% Pressure Distribution in Nip Region
% Maximum Pressure Vs Nip Angle
% Maximum Pressure Vs Po
```

```
% Maximum Pressure Vs Internal Friction Angle
% Maximum Pressure Vs Wall Friction
% Maximum Pressure Vs Roll Gap
% Maximum Pressure Vs Compressibility
% Roll Force VsRollGap
% Roll Torque VsRollGap

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Extracts user input from the gui
%getUserVarInput
%all variables are global
function [] = getUserVarInput(handles)
%set global variables used in functions.
global d2r r2d D S W d K v u RW Po rMin rMax
%get all the variables
%angle variables are converted to radiants
d2r=pi/180;     % conversion factor degrees into radiants
r2d=180/pi;     % conversion factor radiants into degrees

%%%%%%%%
% S=Gap size(between rolls)
% D=Diameter of Rolls
% K=Compressibility constant
% d=delta - internal friction angle of powder
% Po=initial pressure at the entrance (perpendicular to major
principal stress
% W=phi - wall (roll-powder) friction angle (attained from friction
coefficient tan(phi)=mu)
% v=cute angle between the tangent to the roll surface and the
direction of the major principal stress
%     can be calculated using W and d
% RW=roll width
% rMin = min range value
% rMax = max range value
%%%%%%%
S = str2double(get(handles.varS,'String'));
D = str2double(get(handles.varD,'String'));
K = str2double(get(handles.varK,'String'));
d = str2double(get(handles.vard,'String'))*d2r;
Po = str2double(get(handles.varPo,'String'));
RW=str2double(get(handles.varRW,'String'));              %Roll
width

u=(pi/4)-(d/2);
W=str2double(get(handles.varW,'String'));
%w at this moment is the fricition coefficient
%take atan to get the wall friction angle in radiants
W=atan(W);
%calculate v which is used all equations
v=(pi-asin(sin(W)/sin(d)) -W)/2;
rMin = str2double(get(handles.rMin,'String'));
rMax = str2double(get(handles.rMax,'String'));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function a=getAplha(handles)
%Solve for alpha
%calls fsolve function which uses two functions(slip, stick)
%contained in SlipStickFun.m and solves for the point of their
intersection
%returns alpha in radiants
%handles is passed to the function for formality.
%all variables are global
%set global variables used in functions.
global d2r r2d D S W d K v u RW Po rMin rMax
%call fsolve mysystem is the function containing the two
equations.
InitialGuess = [.1;.1];
Options = optimset('Display','off');
XY = fsolve(@SlipStickFun, InitialGuess, Options);
%XY returns the answer, the x component is the angle alpha
a=XY(1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function NipAngle(handles)
%NIP ANGLE
%plots slip and stick pressure gradients
%point of intersection is the nip angle
disp('NipAngle')
%set global variables used in functions outside of this file.
global d2r r2d D S d K v W u Po RW rMin rMax
%get all the variables
getUserVarInput(handles);
a=getAplha(handles)
%value of y when x = a
ay=(K.*(2.*cos(a)-1-S/D).*tan(a))./((D/2).*((1+S/D-
cos(a)).*cos(a)));
% x represents theta
x=linspace(rMin,rMax*d2r);
%slip gradient equation
fn1=(4.*((pi/2)-x-v).*tan(d))./(D/2.*(1+S/D-
cos(x)).*(cot(((x+v+(pi/2))/2)-u)-cot(((x+v+pi/2)/2)+u)));
%stick gradient equation
fn2=(K.*(2.*cos(x)-1-S/D).*tan(x))./((D/2).*((1+S/D-
cos(x)).*cos(x)));
%straight line
z=0;
%get a new or correct figure for plotting
figure(newFigure(handles,'NIP ANGLE'));
plot(x*r2d,fn1,x*r2d,fn2,'k--',x*r2d,z,'k--',a*r2d,ay,'ro');
title('Nip Angle (\alpha)');
xlabel('Angular Position [Deg]'), ylabel('Pressure Gradient');
legend('SLIP','NO SLIP');
legend('boxoff')
grid off;

%prints variables used on the graph:
TPosX=(max(x)*r2d*0.6);       % returns the maximum value in x
direction * factor
TPosY=max(fn1)*0.8;          % returns the maximum value in y
direction * factor
TSp=TPosY*0.09;
text(a*r2d+(TPosX*0.05),ay+TSp,0,['\alpha = ',num2str(a*r2d),'
Deg']);

text(TPosX,TPosY,0,        ['S= ',num2str(S),' mm']);
text(TPosX,TPosY-TSp,0,    ['D= ',num2str(D),' mm']);
text(TPosX,TPosY-TSp*2,0,  ['K= ',num2str(K)]);
text(TPosX,TPosY-TSp*3,0,  ['\delta= ',num2str(d*r2d),' Deg']);
text(TPosX,TPosY-TSp*4,0,  ['\mu= ',num2str(tan(W))]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function NipAngleVsInternalFrictionAngle(handles)
%Nip Angle Vs Internal Friction Angle
%
disp('NipAngleVsInternalFrictionAngle')
global d2r r2d D S d K v W u Po RW rMin rMax
%get all the variables
getUserVarInput(handles);
i=1;
for M=rMin:1:rMax
   d=M*d2r;              %get internal friction angle
   u=(pi/4)-(d/2);       %calculate a new u
   %calculate v which is used all equations
   v=(pi-asin(sin(W)/sin(d)) -W)/2;
   N(i,1)=M;
   N(i,2)=getAplha(handles)*r2d; %get alpha and turn into degrees
for plot
   i=i+1;
end

figure(newFigure(handles,'Nip Angle Vs Internal Friction Angle'));
```

```
plotA=plot(N(:,1),N(:,2));

title('Nip Angle Vs Internal Friction Angle');
xlabel('Internal Friction Angle [Deg]');
ylabel('Nip Angle(\alpha) [Deg]');
grid off;
xData = get(plotA,'XData');     % Get the plotted data
yData = get(plotA,'YData');
% returns the maximum value in x direction
TPosX=(min(xData)) + (max(xData)-min(xData))*0.2;
TPosY=(max(yData))*0.9;          % returns the maximum value in
y direction
TSp=TPosY*0.07;
text(TPosX,TPosY,0,         ['S= ',num2str(S),' mm']);
text(TPosX,TPosY-TSp,0,     ['D= ',num2str(D),' mm']);
text(TPosX,TPosY-TSp*2,0,   ['K= ',num2str(K)]);
text(TPosX,TPosY-TSp*3,0,   ['\mu= ',num2str(tan(W))]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function NipAngleVsWallFriction(handles)
%Nip Angle Vs Wall Friction Angle
%
%Phi=W=wall friction angle user inputs the friction coefficient
% ---------------------------------------------------------------------
disp('NipAngleVsWallFriction')
global d2r r2d D S d K v W u Po RW rMin rMax
%get all the variables
getUserVarInput(handles);
%convert ranges into wall friction angle in radiants
rMin=atan(rMin)
rMax=atan(rMax)
%number of iterations
step=(rMax-rMin)/40;
i=1;
for M=rMin:step:rMax
   W=M;
   v=(pi-asin(sin(W)/sin(d)) -W)/2;
   N(i,1)=tan(M);  %convert wall friction angle to friction
coefficient
   N(i,2)=getAplha(handles)*r2d; %get alpha and turn into degrees
for plot
   i=i+1
end
figure(newFigure(handles,'Nip Angle Vs Wall Friction'));
plotA=plot(N(:,1),N(:,2));
title('Nip Angle Vs Wall Friction');
xlabel('Wall Friction Coefficient (\mu)');
ylabel('Nip Angle (\alpha) [Deg]');

xData = get(plotA,'XData'); % Get the plotted data
yData = get(plotA,'YData');
TPosX=(max(xData))*0.2;         % returns the maximum  value in
x direction
TPosY=(max(yData))*0.9;          % returns the maximum  value in
y direction
TSp=TPosY*0.07;
%print the variables unique to this plot
text(TPosX,TPosY,0,         ['S= ',num2str(S), ' mm']);
text(TPosX,TPosY-TSp,0,     ['D= ',num2str(D), ' mm']);
text(TPosX,TPosY-TSp*2,0,   ['K= ',num2str(K)]);
text(TPosX,TPosY-TSp*3,0,   ['\delta= ',num2str(d*r2d), ' Deg']);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function NipAngleVsCompressibility(handles)
%Nip Angle Vs Compressibility
%
% ---------------------------------------------------------------------

disp('NipAngleVsCompressibility')
global d2r r2d D S d K v W u Po RW rMin rMax
%get all the variables
getUserVarInput(handles);

step=(rMax-rMin)/20;
i=1;
for M=rMin:step:rMax;
   K=M;
   N(i,1)=K;
   N(i,2)=getAplha(handles)*r2d; %get alpha and turn into degrees
for plot
   i=i+1
end
figure(newFigure(handles,'Nip Angle Vs Compressibility'));

plotA=plot(N(:,1),N(:,2));
title('Nip Angle Vs Compressibility');
xlabel('K - Copmressibility ');
ylabel('Nip Angle (\alpha) [Deg]');

xData = get(plotA,'XData'); % Get the plotted data
yData = get(plotA,'YData');
TPosX=(max(xData))*0.8;         % returns the maximum  value in
x direction
TPosY=(max(yData))*0.9;          % returns the maximum  value in
y direction
TSp=TPosY*0.07;
text(TPosX,TPosY,0,         ['S= ',num2str(S), ' mm']);
text(TPosX,TPosY-TSp,0,     ['D= ',num2str(D), ' mm']);
text(TPosX,TPosY-TSp*2,0,   ['\delta= ',num2str(d*r2d),' Deg']);
text(TPosX,TPosY-TSp*3,0,   ['\mu= ',num2str(tan(W))]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function NipAngleVsRollGap(handles)
% Nip Angle Vs Roll Gap
%
% ---------------------------------------------------------
disp('NipAngleVsRollGap')
global d2r r2d D S d K v W u Po RW rMin rMax
%get all the variables
getUserVarInput(handles);
step=(rMax-rMin)/20;
i=1
for M=rMin:step:rMax
   S=M;
   N(i,1)=S;
   N(i,2)=getAplha(handles)*r2d; %get alpha and turn into degrees
for plot
   i=i+1
end
figure(newFigure(handles,'Nip Angle Vs Roll Gap'));
plotA=plot(N(:,1),N(:,2));
title('Nip Angle Vs Roll Gap');
xlabel('Roll Gap [mm]');
ylabel('Nip Angle (\alpha) [Deg]');

xData = get(plotA,'XData');     % Get the plotted data
yData = get(plotA,'YData');
TPosX=(max(xData))*0.2;         % returns the maximum  value in
x direction
TPosY=(min(yData)) + (max(yData)-min(yData))*0.9;   %returns
the maximum  value in y direction
TSp=(max(yData)-min(yData))*0.07;
text(TPosX,TPosY,0,         ['D= ',num2str(D),' mm']);
text(TPosX,TPosY-TSp,0,     ['K= ',num2str(K)]);
text(TPosX,TPosY-TSp*2,0,   ['\delta= ',num2str(d*r2d),' Deg']);
text(TPosX,TPosY-TSp*3,0,   ['\mu= ',num2str(tan(W))]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function PressureDistribution(handles)
%PRESSURE DISTRIBUTION
%Stress vs Angular position
%Assumption - normal stress is equal to pressure excerted on the
```

```
powder by roll
% ---------------------------------------------------------------------
disp('PressureVsAngularPosition')
global d2r r2d D S d K v W u Po RW rMin rMax
%get all the variables
getUserVarInput(handles);
%get the nip angle
a=getAplha(handles);

%%%%%
%Pressure distribution
%Solve the differential equation of powder behavior in SLIP to get pressure at alpha

%initial condition - feed stress (pressure) at x0
so=[Po/(1-sin(d))];
%point of application of feed pressure
x0=(pi/2)-v;
%variable used in ODE solver, the limits of analysis
%first item (limit) is the initial condition point, the second is the limit-0
xspan=[x0,0];
%Runga-Kutta ODE solver
sol=ode45(@slipODE,xspan,so);
% extract the stress at nip angle alpha
sa = deval(sol,a);

%Plot the pressure distribution equation
%ranges are in degrees so convert
x=linspace(rMin*d2r,rMax*d2r);
%pressure distirbution (No slip) equation with initial condition at angle a
s=sa.*((1+S/D-cos(a).*cos(a))./(1+S/D-cos(x).*cos(x))).^K;

figure(newFigure(handles,'Pressure Distribution'));
plotA=plot(x*r2d,s);
title('Pressure Distribution in Nip Region');
xlabel('Angular Position [Deg]');
ylabel('Pressure [MPa]');

xData = get(plotA,'XData'); % Get the plotted data
yData = get(plotA,'YData');
TPosX=(max(xData))*0.7;        % returns the maximum  value in x direction
TPosY=(max(yData))*0.9;        % returns the maximum  value in y direction
TSp=TPosY*0.07;

text(TPosX,TPosY,0,        ['S= ',num2str(S), ' mm']);
text(TPosX,TPosY-TSp,0,    ['D= ',num2str(D), ' mm']);
text(TPosX,TPosY-TSp*2,0,  ['Po= ',num2str(Po), ' Mpa']);
text(TPosX,TPosY-TSp*3,0,  ['K= ',num2str(K)]);
text(TPosX,TPosY-TSp*4,0,  ['\delta= ',num2str(d*r2d), ' Deg']);
text(TPosX,TPosY-TSp*5,0,      ['\mu= ',num2str(tan(W))]);
text(TPosX,TPosY-TSp*6,0,  ['\alpha= ',num2str(a*r2d), ' Deg']);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function MaximumPressureVsNipAngle(handles)
%
%EFFECTS OF NIP ANGLE ON PRESSURE AT ANGULAR DISPLACEMENT 0
%Pressure vs Nip Angle
% -------------------------------------------------------
disp('PressureVsNipAngle')
global d2r r2d D S d K v W u Po RW rMin rMax
%get all the variables
getUserVarInput(handles);

%%%%%
%Solve the differential equation in slip region
%For each alpha
%From the ODE solutions get pressure at each alpha
%Using the Initial condition find what is the value
%of sigma at angular position 0, then plot

%initial condition - feed stress (pressure) at x0
so=[Po/(1-sin(d))];
%point of application of feed pressure
x0=(pi/2)-v;
%variable used in ODE solver, the limits of analysis
%first item (limit) is the initial condition point, the second is the limit-0
xspan=[x0,0];
%Runga-Kutta ODE solver
sol=ode45(@slipODE,xspan,so);
% extract the stress at nip angle alpha
sa = deval(sol,a);

%create an array that will hold alpha
%rmin and max are in deg at this moment
Alpha=linspace(rMin,rMax);
%for all the elements in Alpha get s
%at position 0
x=0;
for M=1:1:(size(Alpha,2))
   N(M,1)=Alpha(M);
   a=Alpha(M)*d2r;
   sa = deval(sol,a); %pressure at alpha from the slip equation
   %the pressure at position 0 = maximum pressure.
   N(M,2)=sa.*((1+S/D-cos(a).*cos(a))./(1+S/D-cos(x).*cos(x))).^K;
end
figure(newFigure(handles,'Maximum Pressure Vs Nip Angle'));
plotA=plot(N(:,1),N(:,2));
title('Maximum Pressure Vs Nip Angle');
xlabel('Nip Angle(\alpha) [Deg]');
ylabel('Maximum Pressure [MPa]');
%grid;
xData = get(plotA,'XData');     % Get the plotted data
yData = get(plotA,'YData');
TPosX=rMin+(rMax-rMin)*0.2;         % returns the maximum value in x direction
TPosY=(max(yData))*0.8;         % returns the maximum  value in y direction
TSp=TPosY*.07;

text(TPosX,TPosY,0,        ['S= ',num2str(S), ' mm']);
text(TPosX,TPosY-TSp,0,     ['D= ',num2str(D), ' mm']);
text(TPosX,TPosY-TSp*2,0,   ['Po= ',num2str(Po), ' Mpa']);
text(TPosX,TPosY-TSp*3,0,   ['K= ',num2str(K)]);
text(TPosX,TPosY-TSp*5,0,   ['\mu= ',num2str(tan(W))]);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function MaximumPressureVsPo(handles)
%EFFECTS OF ENTRY PRESSURE ON PRESSURE AT DISPLACEMENT 0
%Pressure0vsPo
%opens a new plot window
% ---------------------------------------------------------------------

disp('PressureVsPo')
global d2r r2d D S d K v W u Po RW rMin rMax
%get all the variables
getUserVarInput(handles);

%get the nip angle
a=getAplha(handles);

%%%%%
%for each Po calculate a new s!
%calculate so
```

```
%Solve the differential equation in slip region
%For each alpha
%From the ODE solutions get pressure at each alpha
%Using the Initial condition find what is the value
%of sigma at angular position 0, then plot


%point of application of feed pressure
x0=(pi/2)-v;
%first item (limit) is the initial condition point, the second is the
limit-0
xspan=[x0,0];

NumOfEl=100;
%create an array that will hold the different Po values
PoValues=linspace(rMin,rMax,NumOfEl);
%for all the elements in Alpha get s at position 0
x=0;
for M=1:1:(size(PoValues,2));
   %initial condition - feed stress (pressure) at x0
   so=[PoValues(M)/(1-sin(d))];
   %Runga-Kutta ODE solver
   sol=ode45(@slipODE,xspan,so);
   N(M,1)=PoValues(M);
   sa = deval(sol,a);
   N(M,2)=sa.*((1+S/D-cos(a).*cos(a))./(1+S/D-
cos(x).*cos(x))).^K;
end
figure(newFigure(handles,'Maximum Pressure Vs Po'));
plotA=plot(N(:,1),N(:,2));
title('Maximum Pressure Vs Feed Pressure (Po)');
xlabel('Feed Pressure (Po) [MPa] ');
ylabel('Maximum Pressure [MPa]');

xData = get(plotA,'XData');    % Get the plotted data
yData = get(plotA,'YData');
TPosX=(max(xData))*0.2;         % returns the maximum  value in
x direction
TPosY=(max(yData))*0.8;         % returns the maximum  value in
y direction
TSp=TPosY*.07;

text(TPosX,TPosY,0,         ['S= ',num2str(S), ' mm']);
text(TPosX,TPosY-TSp,0,     ['D= ',num2str(D), ' mm']);
text(TPosX,TPosY-TSp*2,0,   ['K= ',num2str(K)]);
text(TPosX,TPosY-TSp*3,0,   ['\delta= ',num2str(d*r2d), ' Deg']);
text(TPosX,TPosY-TSp*4,0,   ['\mu= ',num2str(tan(W))]);
text(TPosX,TPosY-TSp*5,0,   ['\alpha= ',num2str(a*r2d), ' Deg']);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function MaximumPressureVsInternalfFrictionAngle(handles)
%Maximum Pressure (at Position 0) vs Internal Friction Angle
%
% ---------------------------------------------------------------------

disp('Pressure0VsInternalfFrictionAngle')
global d2r r2d D S d K v W u Po RW rMin rMax
%get all the variables
getUserVarInput(handles);

%%%%%
%for each d calculate new alpha!
%calculate so
%Solve the differential equation in slip region
%For each alpha
%From the ODE solutions get pressure at each alpha
%Using the Initial condition find what is the value
%of sigma at angular position 0, then plot

step=(rMax-rMin)/40;
i=1;
for M=rMin:step:rMax
   d=M*d2r;              %get internal friction angle in redients
   u=(pi/4)-(d/2);
   %calculate v which is used all equations
   v=(pi-asin(sin(W)/sin(d)) -W)/2;
   a=getAplha(handles);   %get the nip angle
   %point of application of feed pressure
   x0=(pi/2)-v;
   %initial condition - feed stress (pressure) at x0
   so=[Po/(1-sin(d))];
   %variable used in ODE solver, the limits of analysis
   %first item (limit) is the initial condition point, the second is the
limit-0
   xspan=[x0,0];
   sol=ode45(@slipODE,xspan,so);
   % extract the stress at nip angle alpha
   sa = deval(sol,a);
   N(i,1)=M;             %the internal friciton angle in degrees
   N(i,2)=sa.*((1+S/D-cos(a).*cos(a))./(1+S/D-
cos(x).*cos(x))).^K;
   i=i+1;

end
figure(newFigure(handles,'Maximum Pressure Vs Internal Fricition
Angle'));
plotA=plot(N(:,1),N(:,2));
grid off;
title('Maximum Pressure Vs Internal Fricition Angle');
xlabel('Internal Fricition Angle [Deg]');
ylabel('Maximum Pressure [MPa]');
xData = get(plotA,'XData');    % Get the plotted data
yData = get(plotA,'YData');
TPosX=(max(xData))*0.2;         % returns the maximum  value in
x direction
TPosY=(max(yData))*0.8 ;        % returns the maximum  value in
y direction
TSp=TPosY*.07;

text(TPosX,TPosY,0,         ['S= ',num2str(S), ' mm']);
text(TPosX,TPosY-TSp,0,     ['D= ',num2str(D), ' mm']);
text(TPosX,TPosY-TSp*2,0,   ['Po= ',num2str(Po), ' MPa']);
text(TPosX,TPosY-TSp*3,0,   ['K= ',num2str(K)]);
text(TPosX,TPosY-TSp*4,0,   ['\mu= ',num2str(tan(W))]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function MaximumPressureVsWallFriction(handles)
%Maximum Pressure (at Position 0) vs Wall Friction Angle
% ---------------------------------------------------------------------
disp('MaximumPressureVsWallfFriction')
global d2r r2d D S d K v W u Po RW rMin rMax
%get all the variables
getUserVarInput(handles);

%%%%%
%for each phi(W) calculate new alpha!
%calculate so
%Solve the differential equation in slip region
%For each alpha
%From the ODE solutions get pressure at each alpha
%Using the Initial condition find what is the value
%of sigma at angular position 0, then plot

%angular position
%maximum pressure is at 0
x=0;
%convert coefficient ranges into wall friction angle in radiants
rMin=atan(rMin);
rMax=atan(rMax);
step=(rMax-rMin)/40;
i=1;
so=[Po/(1-sin(d))];
```

```
for M=rMin:step:rMax
   W=M;
   v=(pi-asin(sin(W)/sin(d)) -W)/2;
   a=getAplha(handles);
   x0=(pi/2)-v;
   %variable used in ODE solver, the limits of analysis
   %first item (limit) is the initial condition point, the second is the
limit-0
   xspan=[x0,0];
   sol=ode45(@slipODE,xspan,so);
   % extract the stress at nip angle alpha
   sa = deval(sol,a);
   N(i,1)=tan(M);  %convert wall friction angle to friction
coeffcient
   N(i,2)=sa.*((1+S/D-cos(a).*cos(a))./(1+S/D-
cos(x).*cos(x))).^K;
   i=i+1;

end
figure(newFigure(handles,'Maxiumum Pressure Vs Wall
Fricition'));
plotA=plot(N(:,1),N(:,2));
title('Maxiumum Pressure Vs Wall Fricition');
xlabel('Wall Fricition Coefficient (\mu)');
ylabel('Maxiumum Pressure [MPa]');

xData = get(plotA,'XData');    % Get the plotted data
yData = get(plotA,'YData');
TPosX=(max(xData))*0.2;         % returns the maximum  value in
x direction
TPosY=(max(yData))*0.8;         % returns the maximum  value in
y direction
TSp=TPosY*.07;

text(TPosX,TPosY,0,        ['S= ',num2str(S), ' mm']);
text(TPosX,TPosY-TSp,0,    ['D= ',num2str(D), ' mm']);
text(TPosX,TPosY-TSp*2,0,  ['Po= ',num2str(Po), ' MPa']);
text(TPosX,TPosY-TSp*3,0,  ['K= ',num2str(K)]);
text(TPosX,TPosY-TSp*4,0,  ['\delta= ',num2str(d*r2d), ' Deg']);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function MaximumPressureVsRollGap(handles)
%Pressure at Position 0 vs Roll Gap
%
% ---------------------------------------------------------------------

disp('MaxmimumPressureVsRollGap')
global d2r r2d D S d K v W u Po RW rMin rMax
%get all the variables
getUserVarInput(handles);
%initial condition
so=[Po/(1-sin(d))];

%%%%
%for each S calculate new alpha!
%calculate so
%Solve the differential equation in slip region
%For each alpha
%From the ODE solutions get pressure at each alpha
%Using the Initial condition find what is the value
%of sigma at angular position 0, then plot
%initial condition - feed stress (pressure) at x0
so=[Po/(1-sin(d))];
%point of application of feed pressure
x0=(pi/2)-v;
%variable used in ODE solver, the limits of analysis
%first item (limit) is the initial condition point, the second is the
limit-0
xspan=[x0,0];
i=1;
%maximum pressure is at 0
x=0;
step=(rMax-rMin)/40;
for M=rMin:step:rMax
   S=M;
   a=getAplha(handles);
   %Runga-Kutta ODE solver
   sol=ode45(@slipODE,xspan,so);
   % extract the pressure at nip angle alpha
   sa = deval(sol,a);
   N(i,1)=M;
   N(i,2)=sa.*((1+S/D-cos(a).*cos(a))./(1+S/D-
cos(x).*cos(x))).^K;
   i=i+1;

end
figure(newFigure(handles,'Maximum Pressure Vs Roll Gap'));
%Fig=figure;
%set(Fig,'Name','Pressure at Position 0 Vs Roll Gap')
plotA=plot(N(:,1),N(:,2));
title('Maximum Pressure Vs Roll Gap');
xlabel('Roll Gap [mm]');
ylabel('Maximum Pressure [MPa]');
%grid;

xData = get(plotA,'XData');    % Get the plotted data
yData = get(plotA,'YData');
TPosX=(max(xData))*0.6;         % returns the maximum  value in
x direction
TPosY=(max(yData))*0.8;         % returns the maximum  value in
y direction
TSp=TPosY*.07;

text(TPosX,TPosY-TSp,0,    ['D= ',num2str(D), ' mm']);
text(TPosX,TPosY-TSp*2,0,  ['Po= ',num2str(Po), ' MPa']);
text(TPosX,TPosY-TSp*3,0,  ['K= ',num2str(K)]);
text(TPosX,TPosY-TSp*4,0,  ['\delta= ',num2str(d*r2d), ' Deg']);
text(TPosX,TPosY-TSp*5,0,  ['\mu= ',num2str(tan(W))]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function MaximumPressureVsCompressibility(handles)
%Maximum Pressure at Position 0 vs Roll Diameter
%
% ---------------------------------------------------------------------

global d2r r2d D S d K v W u Po RW rMin rMax
%get all the variables
getUserVarInput(handles);
%intial condition
so=[Po/(1-sin(d))];
%%%%%
%for each S calculate new alpha!
%calculate so
%Solve the differential equation in slip region
%For each alpha
%From the ODE solutions get pressure at each alpha
%Using the Initial condition find what is the value
%of sigma at angular position 0, then plot

%initial condition - feed stress (pressure) at x0
so=[Po/(1-sin(d))];
%point of application of feed pressure
x0=(pi/2)-v;
%variable used in ODE solver, the limits of analysis
%first item (limit) is the initial condition point, the second is the
limit-0
xspan=[x0,0];

i=1;
%maximum pressure is at 0
x=0;
step=(rMax-rMin)/20;
for M=rMin:step:rMax
   K=M;
```

```
    a=getAplha(handles);
    %Runga-Kutta ODE solver
    sol=ode45(@slipODE,xspan,so);
    % extract the pressure at nip angle alpha
    sa = deval(sol,a);
    N(i,1)=K;
    N(i,2)=sa.*((1+S/D-cos(a).*cos(a))./(1+S/D-
cos(x).*cos(x))).^K;
    i=i+1
end
figure(newFigure(handles,'Maximum Pressure Vs
Compressibility'));

plotA=plot(N(:,1),N(:,2));
title('Maximum Pressure Vs Compressibility');
xlabel('Compressibility (K)');
ylabel('Maximum Pressure [MPa]');

xData = get(plotA,'XData');    % Get the plotted data
yData = get(plotA,'YData');
TPosX=(max(xData))*0.2;        % returns the maximum  value in
x direction
TPosY=(max(yData))*0.8;        % returns the maximum  value in
y direction
TSp=TPosY*.07;

text(TPosX,TPosY-TSp,0,    ['S= ',num2str(S), ' mm']);
text(TPosX,TPosY-TSp*2,0,  ['D= ',num2str(D), ' mm']);
text(TPosX,TPosY-TSp*3,0,  ['Po= ',num2str(Po), ' MPa']);
text(TPosX,TPosY-TSp*4,0,  ['\delta= ',num2str(d*r2d),' Deg']);
text(TPosX,TPosY-TSp*5,0,  ['\mu= ',num2str(tan(W))]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function RollForceVsRollGap(handles)
%ROLL FORCE
%
% ------------------------------------------------------------------
disp('RollForce')
global d2r r2d D S d K v W u Po RW rMin rMax
%get all the variables
getUserVarInput(handles);

%Solve for Maximum pressure-Pm which occurs at position 0
%-same as other maximum pressures
%integrate the force factor equation from 0 to alpha
%evaluate the Roll force equation
%plot it versus roll gap

%initial condition - feed stress (pressure) at x0
so=[Po/(1-sin(d))];
%point of application of feed pressure
x0=(pi/2)-v;
%variable used in ODE solver, the limits of analysis
%first item (limit) is the initial condition point, the second is the
limit-0
xspan=[x0,0];


%force factor function
%f=inline('(((S/D)/((1+S/D-cos(x))*cos(x)))^K)*cos(x)')

i=1;
x=0;    %location of maximum stress.
%for each gap size
%calc a
%solve ODE
%get stress at angle a
%calculate stress at position 0
step=(rMax-rMin)/20;
for M=rMin:step:rMax
    N(i,1)=M;
    S=M;  %Roll Gap
    a=getAplha(handles);
    %Runga-Kutta ODE solver
    sol=ode45(@slipODE,xspan,so);
    % extract the stress at nip angle alpha
    sa = deval(sol,a);
    Pm=sa.*((1+S/D-cos(a).*cos(a))./(1+S/D-cos(x).*cos(x))).^K;
    %calculate Force factor F by integrating from 0 to a
    F=quad(@forceFactor,0,a);
    %Roll Force in newtons
    %this needs to be investigated not sure of conversion) /100
    N(i,2)=(Pm*RW*D*0.5*F)/100;
    i=i+1
end

figure(newFigure(handles,'Roll Force Vs Roll Gap'));
plotA=plot(N(:,1),N(:,2));
title('Roll Force Vs Roll Gap');
xlabel('Roll Gap [mm]');
ylabel('Roll Force [N]');

xData = get(plotA,'XData');    % Get the plotted data
yData = get(plotA,'YData');
TPosX=(max(xData))*0.6;        % returns the maximum  value in
x direction
TPosY=(max(yData))*0.8;        % returns the maximum  value in
y direction
TSp=TPosY*.07;

text(TPosX,TPosY,0,        ['D= ',num2str(D),' mm']);
text(TPosX,TPosY-TSp,0,    ['K= ',num2str(K)]);
text(TPosX,TPosY-TSp*2,0,  ['RW= ',num2str(RW),' mm']);
text(TPosX,TPosY-TSp*3,0,  ['Po= ',num2str(Po),' MPa']);
text(TPosX,TPosY-TSp*4,0,  ['\delta= ',num2str(d*r2d),' Deg']);
text(TPosX,TPosY-TSp*5,0,  ['\mu= ',num2str(tan(W))]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function RollTorqueVsRollGap(handles)
%ROLL TORQUE
%
% ------------------------------------------------------------------
disp('RollTorque')
global d2r r2d D S d K v W u Po RW rMin rMax
%get all the variables
getUserVarInput(handles);

%Solve for Maximum pressure (Pm) which occurs at position 0
%-same as other maximum pressures
%integrate the torque factor equation from 0 to alpha
%evaluate the Roll Torque equation
%plot it versus roll gap

%initial condition - feed stress (pressure) at x0
so=[Po/(1-sin(d))];
%point of application of feed pressure
x0=(pi/2)-v;
%variable used in ODE solver, the limits of analysis
%first item (limit) is the initial condition point, the second is the
limit-0
xspan=[x0,0];

x=0;    %location of maximum stress.
%for each gap size
%calc a
%solve ODE
%get stress at angle a
%calculate stress at position 0
%
i=1;
step=(rMax-rMin)/20;
for M=rMin:step:rMax
    S=M;  %Roll Gap
    N(i,1)=S;
```

```
    a=getAplha(handles);
    %Runga-Kutta ODE solverlha(handles);
    sol=ode45(@slipODE,xspan,so);
    % extract the stress at nip angle alpha
    sa = deval(sol,a);
    Pm=sa*((1+S/D-cos(a)*cos(a))/(1+S/D-cos(x)*cos(x)))^K;
    %calculate Force factor T by integrating from 0 to a
    %Torque factor T
    T=quad(@torqueFactor,0,a);
    %Roll Torque (divide by 10 e 6 to get N-Meters
    %(this needs to be investigated not sure of conversion)
    N(i,2)=Pm*RW^2*D*0.125*T/1000000;
    i=i+1
end
figure(newFigure(handles,'Roll Torque Vs Roll Gap'));

plotA=plot(N(:,1),N(:,2));
title('Roll Torque Vs Roll Gap ');
xlabel('Roll Gap  [mm]');
ylabel('Roll Torque [N-M]');

xData = get(plotA,'XData');     % Get the plotted data
yData = get(plotA,'YData');
TPosX=(max(xData))*0.7;         % returns the maximum  value in
x direction
TPosY=(max(yData))*0.9;         % returns the maximum  value in
y direction
TSp=TPosY*.07;

text(TPosX,TPosY-TSp,0,     ['D= ',num2str(D),' mm']);
text(TPosX,TPosY-TSp*2,0,   ['Po= ',num2str(Po),' MPa']);
text(TPosX,TPosY-TSp*3,0,   ['RW= ',num2str(RW),' mm']);
text(TPosX,TPosY-TSp*4,0,   ['K= ',num2str(K)]);
text(TPosX,TPosY-TSp*5,0,   ['\delta= ',num2str(d*r2d),' Deg']);
text(TPosX,TPosY-TSp*6,0,   ['\mu= ',num2str(tan(W))]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%The Load and save system for powder and process database
%
% --------------------------------------------------------------------
function varargout = PowderName_Callback(h, eventdata,
handles, varargin)
set(handles.PowderLabel,'String',get(h,'string'));
% --------------------------------------------------------------------
function varargout = PowderList_Callback(h, eventdata, handles,
varargin)
% --------------------------------------------------------------------
function varargout = removeButton_Callback(h, eventdata,
handles, varargin)
%remove the powder currently selected
%shift the powder list to eliminate gaps
%possibility of adding warning that there are no more powders in
menu.
handles.PowderDB(get(handles.PowderList,'Value'))=[];
guidata(h,handles); % store the changes in the handles structure
%select the first item in the menu
set(handles.PowderList,'Value',1);
UpdateList(handles);
% --------------------------------------------------------------------
function varargout = addButton_Callback(h, eventdata, handles,
varargin)
%get all data from user input
%get the number of powders add one
PowderNum=size(handles.PowderDB,2)+1;
handles.PowderDB(PowderNum).Name=get(handles.PowderName
,'String');
handles.PowderDB(PowderNum).W=get(handles.varW,'String');
handles.PowderDB(PowderNum).d=get(handles.vard,'String');
handles.PowderDB(PowderNum).K=get(handles.varK,'String');
handles.PowderDB(PowderNum).S=get(handles.varS,'String');
handles.PowderDB(PowderNum).D=get(handles.varD,'String');
handles.PowderDB(PowderNum).Po=get(handles.varPo,'String');
handles.PowderDB(PowderNum).RW=get(handles.varRW,'String')
;
guidata(h,handles); % store the changes
UpdateList(handles);

% --------------------------------------------------------------------
function varargout = loadButton_Callback(h, eventdata, handles,
varargin)
LoadPowder(handles, get(handles.PowderList,'Value'));
% --------------------------------------------------------------------
function varargout = File_Callback(h, eventdata, handles,
varargin)
% --------------------------------------------------------------------
function varargout = Load_Callback(h, eventdata, handles,
varargin)
%loads/opens a database file
[filename, pathname] = uigetfile(...
        {'*.pdb', 'All PDB-Files (*.pdb)'; ...
        '*.*','All Files (*.*)'}, ...
        'Select Powder Database');
% If "Cancel" is selected then return
if isequal([filename,pathname],[0,0])
    return
% Otherwise construct the fullfilename and Check and load the
file
else
    File = fullfile(pathname,filename);
    data = load(File, '-Mat');
    flds = fieldnames(data);
    % The file is valid if the variable is called "P" and it has
    % fields called "Name" and "W"  two are enough
    % Validate the MAT-file
    pass=0
    if (length(flds) == 1) & (strcmp(flds{1},'P'))
                fields = fieldnames(data.P);
                if (length(fields) == 8) &(strcmp(fields{1},'Name'))
& (strcmp(fields{2},'W'))
                        pass = 1;
        handles.PowderDB=data.P;
        handles.LastFIle = File;
        guidata(h,handles) %store in global structure
                end
    end
    if pass==0
            errordlg('Not a valid Powder Database','Powder
Database Error')
    end% if the PDB-file is not valid, do not save the name

end

disp(['Loaded : ',File]);
UpdateList(handles);
LoadPowder(handles, 1);
% --------------------------------------------------------------------
function varargout = Save_Callback(h, eventdata, handles,
varargin)
% Get the Tag of the menu selected either save or saveas
Tag = get(h,'Tag');
% Get the powder array
P=handles.PowderDB;
% Based on the item selected, take the appropriate action
File = handles.LastFile;
%if never saved before or save as
if (File=='') | (Tag=='SaveAs')
% Allow the user to select the file name to save to
            [filename, pathname] = uiputfile( ...
                    {'*.pdb';'*.*'}, ...
                    'Save as');
            % If 'Cancel' was selected then return
            if isequal([filename,pathname],[0,0])
                    return
            else
```

```
                    % Construct the full path and save
                    File = fullfile(pathname,filename);
                    save(File,'P')
                    handles.LastFile = File;
                    guidata(h,handles)
        end
else
   % Save P database to the default file
   save(File,'P')
end
disp(['Saved :',File]);

% --------------------------------------------------------------------
function varargout = PrintDB_Callback(h, eventdata, handles, varargin)
%this function allows users to preview the GUI before printing
printpreview(handles.Main)
% --------------------------------------------------------------------
function varargout = Exit_Callback(h, eventdata, handles, varargin)
  button = questdlg('Are you sure you want to Exit?',...
   'Closing Program?','Yes','No','Help','No');
   if strcmp(button,'Yes')
      disp('Exiting')
      delete(handles.Main)
   elseif strcmp(button,'No')
      disp('Canceled Exit operation')
      return
   elseif strcmp(button,'Help')
      disp('Sorry, no help available :(')
   end
% --------------------------------------------------------------------
function varargout = Help_Callback(h, eventdata, handles, varargin)
% --------------------------------------------------------------------
function varargout = About_Callback(h, eventdata, handles, varargin)
msgbox('Johanson Method -- Marcin Balicki -- EMAC 04/2003','About')
% --------------------------------------------------------------------
function varargout = HelpGuide_Callback(h, eventdata, handles, varargin)
  pos_size = get(handles.Main,'Position');
  dlg_pos = [pos_size(1)+pos_size(3)/5 pos_size(2)+pos_size(4)/5];
  user_response = modaldlg(dlg_pos);

% --------------------------------------------------------------------
function varargout = checkPParam_Callback(h, eventdata, handles, varargin)
% --------------------------------------------------------------------
function LoadPowder(handles, PowderNum)
%when a selection is made load that particular powder
%check if the the user wants to load the process parameters also
P=handles.PowderDB(PowderNum);
set(handles.PowderName,'String',P.Name);
set(handles.PowderLabel,'String',P.Name);
set(handles.varW,'String', P.W);
set(handles.vard,'String', P.d);
set(handles.varK,'String', P.K);

if get(handles.checkPParam, 'Value')==1;
   set(handles.varS,'String',P.S);
   set(handles.varD,'String', P.D);
   set(handles.varRW,'String',P.RW);
   set(handles.varPo,'String',P.Po);
end
% --------------------------------------------------------------------
function UpdateList(handles)
%check the size of the database
%take each powder name and place it in the list
P=handles.PowderDB;
%reset list
s='';
%number of powders in a list
for M=1:1:size(P,2);
   P(M).Name;
   s=strvcat(s,P(M).Name);
end
%select the first powder in the lits
set(handles.PowderList,'Value',1);
%update the display
set(handles.PowderList,'String',s);

% --------------------------------------------------------------------
function varargout = Export_Callback(h, eventdata, handles, varargin)
P=handles.PowderDB;
% export file format is ( - is a space)
%Name-FrictionalCoefficient-InternalFrictionAngle-Compressibility-RollGap-RollDia-Po-RollWidth
% Allow the user to select the file name to save to
[filename, pathname] = uiputfile( ...
            {'*.txt';'*.*'}, ...
            'Save as');
% If 'Cancel' was selected then return
if isequal([filename,pathname],[0,0])
            return
else
            % Construct the full path and save
            File = fullfile(pathname,filename);
            fid = fopen(File,'w');
   for N=1:size(P,2)
      txt=[P(N).Name,' ',P(N).W,' ',P(N).d,' ',P(N).K,' ',P(N).S,' ',P(N).D,' ',P(N).Po,' ',P(N).RW];
      fprintf(fid,'%s\n',txt);
   end
   fclose(fid);
end
disp(['Exported :  ',File])
```

SlipStickFun.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%     Johanson Model
%%     Date May 6, 2003
%%     Marcin Balicki
%%     Ecole Des Mines D'Albi
%%     balicki@enstimac.fr
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Description:
%% A function representing the two gradient equations
%% The first function represents the pressure gradient when the
%% powder is slipping and the second when the powder sticks
%% to the wall surface
%% This format is required for the fsolve command which accepts
%% a refrence to a function and solves for x
%% the variables are defined in the main program

function F=SlipStickFun(V)
global d2r r2dD S D W d K v u
      x=V(1); y=V(2);

F=[y-(4.*(pi/2-x-v).*tan(d))./((D/2.*(1+S/D-cos(x))).*(cot(((x+v+pi/2)/2)-u)-cot(((x+v+pi/2)/2)+u)));
y-(K.*(2.*cos(x)-1-S/D).*tan(x))./((D/2).*((1+S/D-cos(x)).*cos(x)))];
```

slipODE.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%     Johanson Model
```

```
%%      Date May 6, 2003
%%      Marcin Balicki
%%      Ecole Des Mines D'Albi
%%      balicki@enstimac.fr
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Description:
%% A function representing the differential equation of the
%% pressure gradient when the powder is slipping against
%% the wall surface. This function representation is used
%% by the numerical ODE solver
%%
%% the variables are defined in the main program

function dsdx=slipODE(s,x)
global d2r r2d S D B d K v u

dsdx=[(4.*s.*(pi/2-x-v).*tan(d))./((D/2.*(1+S/D-
cos(x))).*(cot(((x+v+pi/2)/2)-u)-cot(((x+v+pi/2)/2)+u)))];
```

forceFactor.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%      Johanson Model
%%      Date May 6, 2003
%%      Marcin Balicki
%%      Ecole Des Mines D'Albi
%%      balicki@enstimac.fr
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Description:
%% a function representing the FORCE factor
%% used in calculating the roll force
%% the variables are defined in the main program

function f=forceFactor(x)
global d2r r2d S D B d K v u

f=(((S./D)./((1+S./D-cos(x)).*cos(x))).^K).*cos(x);
```

torqueFactor.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%      Johanson Model
%%      Date May 6, 2003
%%      Marcin Balicki
%%      Ecole Des Mines D'Albi
%%      balicki@enstimac.fr
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Description:
%% a function representing the TORQUE factor
%% used in calculating the roll TORQUE
%% the variables are defined in the main program

function t=torqueFactor(x)
global d2r r2d S D B d K v u

t=(((S./D)./((1+S./D-cos(x)).*cos(x))).^K).*cos(2.*x);
```

modaldlg.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%      Johanson Model
%%      Date May 6, 2003
%%      Marcin Balicki
%%      Ecole Des Mines D'Albi
%%      balicki@enstimac.fr
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Description:
%% The help figure code
%% It displays helpful information and is called from the help
%% pulldown menu
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Note:
%% The program was developed on a Solaris machine
%% Graphics may be different on other platforms

function answer = modaldlg(varargin)
% modaldlg Application M-file for modaldlg.fig
%    answer = modaldlg return the answer.
%    modaldlg('callback_name') invoke the named callback.
%    modaldlg([left bottom]) locates the dialog.
% Last Modified by GUIDE v2.0 20-Jul-2000 13:59:31

error(nargchk(0,4,nargin)) % function takes only 0 or 4 argument
if nargin == 0 | isnumeric(varargin{1}) % LAUNCH GUI

  fig = openfig(mfilename,'reuse');

  % Use system color scheme for figure:
  set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

  % Generate a structure of handles to pass to callbacks, and store
it.
  handles = guihandles(fig);
  guidata(fig, handles);
  axes(handles.figImage);
  iptsetpref('ImshowBorder','tight');
  iptsetpref('ImshowAxesVisible','off');
  iptsetpref('ImshowBorder','tight');
  imshow('johansonModel.jpg');
  %axes(handles.figImage);
  %x=linspace(0,5);
  %g=plot(sin(x),x);
  %Position figure
   if nargin == 1
          pos_size = get(fig,'Position');
          pos = varargin{1};
          if length(pos) ~= 2
                 error('Input argument must be a 2-element
vector')
          end
      %take this part out to prevent the new figure from opening off
screen
          %new_pos = [pos(1) pos(2) pos_size(3) pos_size(4)];
          %set(fig,'Position',new_pos,'Visible','on')
          figure(fig)
   end



  % Wait for callbacks to run and window to be dismissed:
  uiwait(fig);

  % UIWAIT might have returned because the window was deleted
using
  % the close box - in that case, return 'cancel' as the answer, and
  % don't bother deleting the window!
  if ~ishandle(fig)
          answer = 'cancel';
  else
          % otherwise, we got here because the user pushed one
of the two buttons.
          % retrieve the latest copy of the 'handles' struct, and
return the answer.
          % Also, we need to delete the window.
          handles = guidata(fig);
      answer = handles.answer;
          delete(fig);
```

```
 end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION
OR CALLBACK

        try
                if (nargout)
                        [varargout{1:nargout}] =
feval(varargin{:}); % FEVAL switchyard
                else
                        feval(varargin{:}); % FEVAL
switchyard
                end
        catch
                disp(lasterr);
        end
end

% --------------------------------------------------------------
% No button callback stores 'no' in the handles struct, and
% stores the modified handles struct
% (so the main function can see the change).
% --------------------------------------------------------------
function varargout = noButton_Callback(h, eventdata, handles,
varargin)
handles.answer = 'OK';
guidata(h, handles);
uiresume(handles.figure1);
```

README.TXT

ROLLING COMPACTION OF POWDERS - JOHANSON METHOD
AUTHOR - MARCIN BALICKI
Ecole des Mines d'Albi
2003

*
The program was developed on a Solaris machine;
Graphics may be different on other platforms.

Required files:

nip.m	This file - main program
nip.fig	the graphical user interface for this program
SlipStickFun.m	two functions representing the state of the powder
slipODE.m	ODE for the powder while slip occurs
forceFactor.m	force factor equation
torqueFactor.m	torque factor equation
modaldlg.m	help menu
modaldlg.fig	help menu graphical user interface
johansonModel.jpg	image of the system

Run Instructions:

All the files should be located in the same directory.
Once Matlab is up and running there are two methods of stating the program:

a) Click on File Open, choose the directory containing required files,
click on nip.m.
This will open a Graphical User Interface (GUI).

b) Type nip on the Matlab command line.
If the local path is the directory containing the required files
a GUI will appear.

ENJOY!

balick@cooper.edu

Author :

Marcin BALICKI

Home Address :

87 Dobbin Street
Apartment 210
Brooklyn, NY 11222
USA

School Address:

The Cooper Union
51 Astor Place
New York, NY 10003
USA

E-Mail :

conceptcatcher@hotmail.com