THE UNIVERSITY OF CALGARY

MODELLING THE MOVEMENT OF SOLIDS THROUGH FLIGHTED ROTATING DRUMS

by

Richard G. Sherritt

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

MASTER OF ENGINEERING

DEPARTMENT OF

CHEMICAL AND PETROLEUM ENGINEERING

CALGARY, ALBERTA

November, 1991

ISBN 0-315-75183-5

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled, "Modelling the Movement of Solids Through Flighted Rotating Drums" submitted by Richard G. Sherritt in partial fulfillment of the requirements for the degree of Master of Engineering.

Dr. L.A. Behie, Supervisor
Department of Chemical and Petroleum Engineering

Dr. A.K. Mehrotra
Department of Chemical and Petroleum Engineering

Dr. R.D. Rowe
Department of Mechanical Engineering

November 12, 1991

ABSTRACT

A rotating drum with lifting flights is an important piece of industrial equipment used to contact granular solids with a gas stream. Rotary dryers, the most common application of drums with flights, are used for drying materials such as wood chips, coal, grain, metallurgical ores and fertilizer pellets.

A mathematical model which determines the residence time of particles in a rotary drum with lifting flights is derived. In the model, particles are considered in two phases; the airborne phase and the dense phase. The holdup and the flow rate of particles in both phases are determined. The model incorporates several new approaches and techniques not found in previous rotary drum models.

By defining the shape of the lifting flights by the rotation angle and the length of the dense phase surface, the flight discharge rate is calculated directly and the holdup is found by integrating the discharge rate. This allows the model to be conveniently applied to a drum with any type of flight. The flow of the dense phase is determined by considering the length of the dense phase surfaces in the discharging and the non-discharging flights. Although the approximations for the surface length in non-discharging flights needs to be improved, the method of determining the dense phase flow is an improvement from previous empirical methods.

This is the first drum residence time model that considers a feed with a distribution of particle sizes. It is also the first model to solve for holdup which may

vary with axial location allowing it to simulate horizontal drums.

Experiments in a large wind tunnel were conducted to study the shielding effect on the horizontal displacement of particles falling in sheets. The results support the concept that a lower than average gas velocity exists inside the sheets. An empirical equation is used to relate the velocity inside the sheets to the flight discharge rate, the fall distance and the axial length of the flight. When the empirical equation was incorporated into the residence time model, the effect of the gas rate on the particles residence time more closely fit experimental results.

Overall, the model is more general than previous models. The improved capabilities of the model are demonstrated by simulating inclined and horizontal drums with cocurrent and countercurrent gas flow. Not only may the drum have any type of lifting flights and even a centrefill with flights, but also the feed may have a single or a mixture of particle sizes. Moreover, the drum may be underloaded or overloaded.

ACKNOWLEDGMENTS

More than a few kind persons have contributed to the creation of this document and I am glad to have this opportunity to extent my appreciation to them.

Many of my colleagues at UMATAC Industrial Processes, A Division of UMA Engineering Ltd. have assisted beyond the call of duty. Daune Muir helped acquire the research literature. Al Eroshinsky, Marvin Bowron, Jerry Yacie, Brian Baillee, Rick Lavallie, Rick Vermeerch, Lee Nathe, and Marat Rubinshteyn worked many hard hours constructing the wind tunnel apparatus and executing experiments. Allan Kurceba and Mark Erratt provided quality drafting. Bill Taciuk, Sean Goodwin and Don Blood were always willing to help when problems arose or to listen when I needed to discuss a complex idea or two. Special thanks to Rod Caple, who was always supportive and contributed several novel perspectives which are included in this study.

The author would also like to thank Bob Turner of Alberta Oil Sands Technology and Research Authority and Les Baker of Southern Pacific Petroleum N.L. for arranging permission to use experimental data from the wind tunnel experiments.

A word of thanks is also deserved by Dr. Leo Behie, my academic supervisor, for his much appreciated recommendations and technical critique.

Finally, special thanks to my wife, Carole, and children, Stephanie and Craig, for their patience and understanding.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

Page

TABLE OF CONTENTS (continued)

Page

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

Page

LIST OF TABLES

LIST OF FIGURES

Page

LIST OF FIGURES (continued)

Page

LIST OF FIGURES (continued)

Page

LIST OF FIGURES (continued)

NOMENCLATURE

$a = \sqrt{\frac{g\sin|\alpha|}{K}}$

a_m = Sheikh's dense phase velocity number

A = frontal area of sheets of particles, m^2

B = fraction of cycle time that particle is stationary relative to the wall

c = length of surface cord on non-discharging flight, m

C_D = drag coefficient

D_{ex} = lotus diameter of exterior flight tips, m

D_i = centrefill wall diameter, m

D_{in} = lotus diameter of interior flight tips, m

D_o = drum diameter, m

D_p = particle diameter, m

f = axial flow rate of particles of one flight, kg/s

F_0 = particle feed rate, kg/s

F_1 = axial flow rate of airborne particles, kg/s

F_2 = axial flow rate of dense phase particles, kg/s

F_D = drag force, N

F_g = gravitational force, N

Fr = Froude's number, dimensionless

g = gravitational acceleration, m/s^2

h = holdup of particles in a flight, m^3/m

H = total drum holdup, m^3/m

H_1 = holdup of particles in airborne phase, m^3/m

H_2 = holdup of particles in dense phase, m^3/m

I = integral of $\ell^2\delta$, m^3 radians

J = integral of ℓ^3, m^3 radians

k = empirical constant

$$K = \frac{3C_D\rho_g}{4D_p\rho_p}$$

ℓ = length of surface cord on a discharging flight, m

L = length of drum, m

m = empirical constant

m = mass holdup of a flight, kg/m

m = ratio of actual holdup to design holdup

m_p = mass of particle, kg

M_1 = total mass holdup of airborne particles, kg

n = rotational speed, rps

N_{ex} = number of exterior flights

N_{in} = number of interior flights

Re_p = Reynold's number for a particle, dimensionless

t = time, s

t_1 = time of fall, s

T = total residence time, s

V_c = axial gas velocity within curtains, m/s

V_g = average axial gas velocity, m/s

V_s = axial velocity in space between curtains, m/s

V_x = particle velocity in axial direction, m/s

$x_{d,i}$ = mass fraction of particles of size i in drum

$x_{f,i}$ = mass fraction of particles of size i in feed

y = vertical distance of fall, m

Greek symbols

α = incline of drum to horizontal, radians

β = central angle of fill of rolling bed, radians

δ = axial displacement, m

θ = angle of rotation of exterior flight, radians

μ = coefficient of friction

μ_g = gas viscosity, kg/m s

ρ_b = bulk density of particles, kg/m^3

ρ_g = gas density, kg/m^3

ρ_p = particle density, kg/m^3

ϕ = kinetic angle of repose, radians

ψ = angle of rotation of interior flight, radians

ω = angular rotational speed, radians/s

Subscripts

0 feed

1 airborne phase

2 dense phase

ex exterior flight

g gas

i particle size grouping or location of initial discharge

in interior flight

p particle

CHAPTER 1

INTRODUCTION

Rotating drums with lifting flights are used to contact granular solids with a gas stream in operations such as the drying or cooling of particles where process temperatures are not too extreme. Drums are popular whenever a large throughput is required. They are noted for their flexibility for handling difficult feeds and their ease of operation. Rotary dryers, the most common application of rotating drums with lifting flights, can accommodate many different feedstocks, such as wood particles, coal, grain, metallurgical ores, and fertilizer pellets.

An example of a rotating drum with lifting flights is shown in Figure 1.1. Features of a drum can vary greatly. Drums, which may be horizontal or inclined up to 5 degrees, rotate at relatively low speeds, usually between 3 and 8 revolutions per minute. Industrial drums vary in shell size from 0.3 metres diameter by 2 metres length to 7 metres diameter by 90 metres length.

Drums may be single pass as shown in Figure 1.1 or multipass. In a single pass drum, solids enter one end and move down the length of the drum to the exit at the opposite end. In a triple pass drum, consisting of three concentric shells, the solids travel through the centre shell and are discharged into the middle shell. The solids change direction and make a second pass. A third pass is made in the outer most shell.

The gas flow may be cocurrent or countercurrent to the solids flow. Drums with cocurrent flow may be either horizontal or inclined, while drums with countercurrent

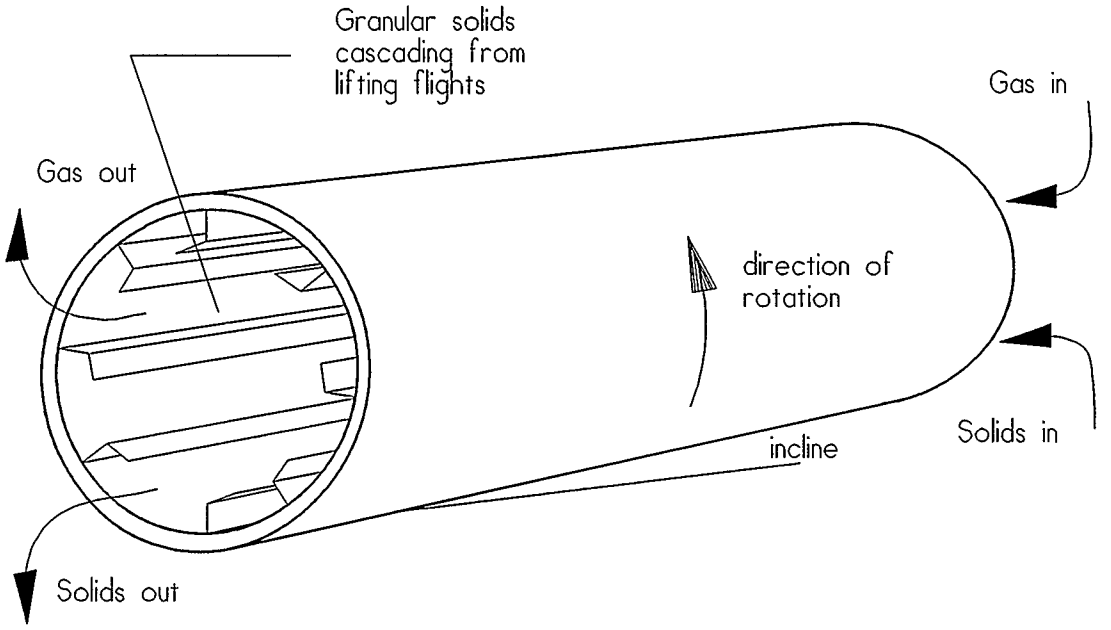


Figure 1.1 An example of a rotating drum with lifting flights to contact granular solids with a gas stream in cocurrent flow

flow of gas and solids are invariably inclined. In an inclined drum, the solids are fed to the high end and are discharged at the low end.

Inside the drum are a number of flights, spaced about the circumference and extending along the length of the drum. As the drum rotates, the flights lift and cascade the particles into the gas stream. The cascading particles form sheets or curtains which traverse the drum below the flights.

Flight shape, number and arrangement can vary greatly as illustrated in typical manufacturer literature [ABB Raymond (1990), Kennedy Van Saun (1982)]. Figure 1.2 illustrates the cross sections of some flighted rotating drums. Drums may have an open centre (a), or may have some type of centrefill (b). The centrefill itself may have flights (c) which catch the falling material and redisperse it below the centrefill. The centrefill could be the inner shell of a multipass drum.

The modelling of rotating drum operations is typically separated into two areas. The first area is concerned with the residence time of the particles including their location and movement in the drum. Once the particle dynamics are known, the second area concerning the heat and mass transfer between the particles and the gas stream can be addressed. This study considers only the first area, that is, factors affecting the residence time of particles in a rotating drum.

In Chapter 2, the published literature on the residence time of particles in flighted rotating drums is reviewed. Prior to the 1960's, only empirical equations were available. Since then, most publications report approaches based on first principles and empirical relationships derived outside of rotating drums. Chapter 2 also introduces definitions and

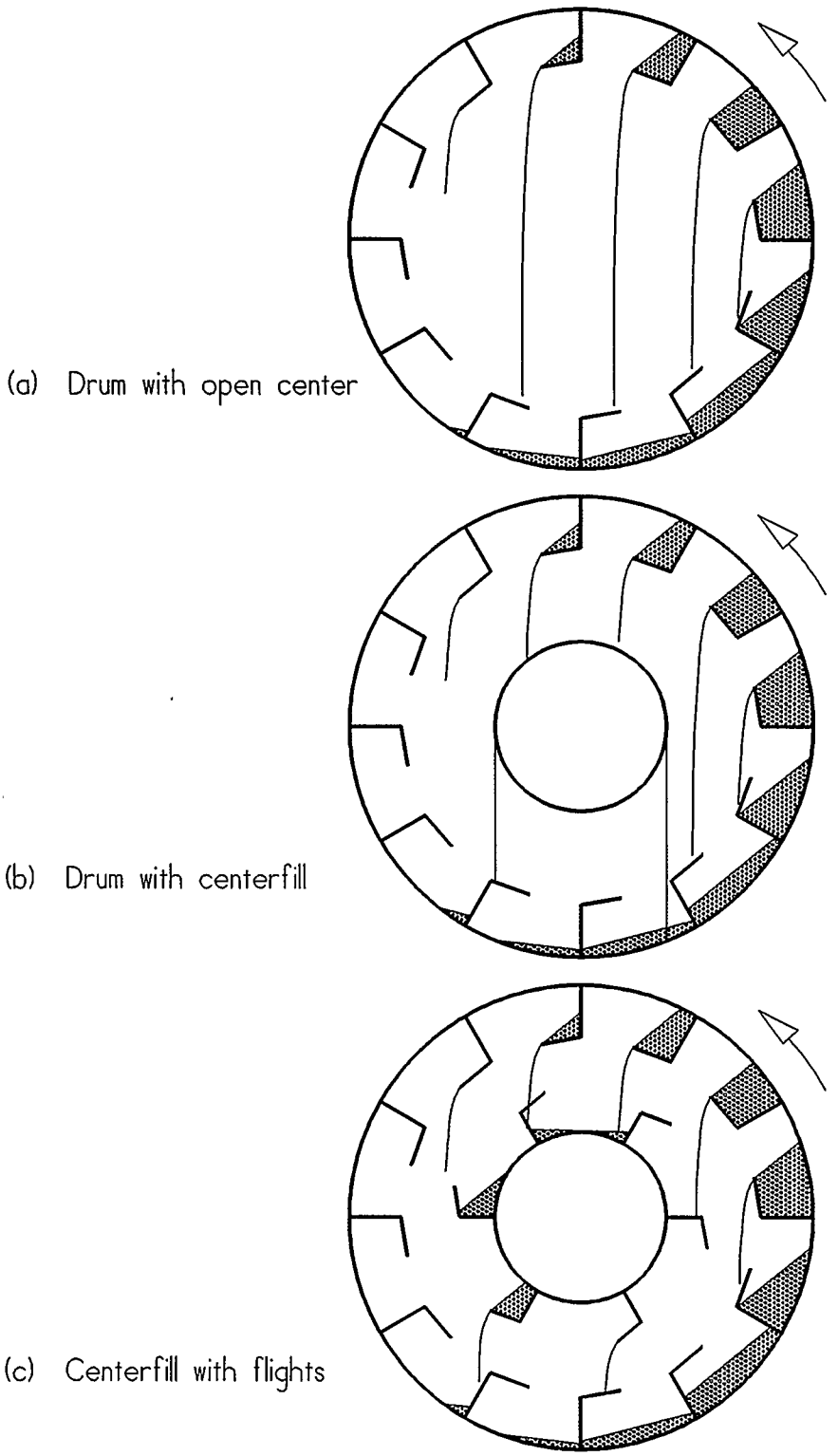


Figure 1.2 Cross sections of a drum with an open centre and drums with a centrefill

concepts developed by previous authors.

Indications are that the more scientific approach available in the published literature has been slow in its acceptance in actual industrial designs [Reay (1979), van Brackel (1979), Purcell (1979)]. One reason may be the complexity of describing the movement of the particles through the drum, but another reason is that the existing residence time models are only applicable to specific types of drums or to a certain operation mode. For example, most existing models can only be applied to inclined drums with open centres with one or two types of flights. None of the models consider a feedstock which consists of a distribution of particle sizes nor a drum in which the holdup may vary along its length.

In Chapter 3, equations for particle holdup and flow rate are derived. The equations are applicable to most drums no matter the shape of the flights, whether the drum is inclined or horizontal, or whether there is a centrefill. The derivations consider cocurrent and countercurrent flow and a material with a mixture of particle sizes.

Another shortcoming of the existing models is their inability to accurately describe the interaction between the airborne or falling particles and the gas stream. In Chapter 4, some experiments investigating the horizontal displacement of a sheet of particles falling through a horizontal gas stream are described. The experimental results are used to derive an empirical correction to the average gas velocity to determine the apparent lower-than-average velocity encountered by the falling particles.

In Chapter 5, equations from Chapters 2, 3, and 4 are combined to solve for the residence time of particles in a flighted rotating drum. Because the equations can be

solved differently depending on whether constant holdup along the drum length can be assumed, solution algorithms are developed for both cases. Predictions made by the resulting computer programs are compared to the published experimental data.

In Chapter 6, a summary of the results and recommendations for future study are given.

CHAPTER 2

LITERATURE REVIEW

2.1 Empirical Models

Prutton, Miller and Schuette (1942) were the first to report an empirical equation for estimating the solids residence time of a flighted drum. Experimental results from a small 0.2 metre diameter drum were used to develop the following relationship.

$$T = \frac{kL}{D\,n\tan\alpha} + m\,V_g \qquad (1)$$

where T is the solids residence time, L is the length of the drum, D is the diameter of the drum, n is the rotational speed, α is the incline, and V_g is the gas velocity. The dimensionless coefficient k was determined to be dependent on the number and type of flights. The coefficient m was dependent on properties of the particles and the direction of the gas flow.

Based on his experience with full scale equipment, Smith (1942) reported equations of the following form.

$$T = \frac{6kL}{D\,n\tan\alpha} \qquad (2)$$

The coefficient k was a function of the gas velocity. The constant 6 became a 9 for a drum with a centrefill with flights.

Using data from tests on a small 0.3 metre diameter drum, Friedman and Marshall (1949) developed a similar empirical residence equation.

$$T = \frac{13.8\,L}{D\,n^{0.9}\tan\alpha} \pm 118.1\frac{0.005\,L\,G}{D_p^{0.5}\,F} \qquad (3)$$

where G is the gas mass rate, D_p is the particle diameter and F is the solids mass rate.

Seaman and Mitchell (1954) also developed an empirical relationship for the residence time of particles in a flighted rotating drum.

$$T = \frac{L}{f(H)\,D\,n(\tan\alpha + m\,V_g)} \qquad (4)$$

Coefficients, $f(H)$, a function of the drum holdup, and m were determined from data obtained using a small 0.3 metre diameter dryer.

Beginning in the 1960's, most researchers have taken a more theoretical approach to determining the residence time of flighted rotating drums. However, some empirical equations continue to be reported, such as that by Hallström (1985) who correlated data from a large 3.3 metre diameter industrial dryer.

$$T = \frac{L}{5.27\,D\,n\,(\tan\alpha + 0.002\,V_g^{1.4})} \qquad (5)$$

A more theoretical approach considers the geometry of the drum and the dynamics

of the particles in much more detail.

2.2 Kinetic Angle of Repose

Figure 2.1 shows a discharging flight in a rotating drum, where the tip of the flight is located at angle θ to the horizontal centreline. The angle is measured in the direction of rotation. In this study, the convention is that θ is equal to zero where the flight tip crosses the centreline as it rotates from the upper half to the lower half of the drum. Angle θ is equal to π where the tip crosses the horizontal centreline as it rotates from the lower half to the upper half of the drum. Although this is opposite to the convention used by previous authors, the reason for the change in the zero reference for angle θ is convenient for the model to be developed in the next chapter.

The kinetic angle of repose, ϕ, is the angle of the surface of the free flowing particles to horizontal. The kinetic angle of repose varies with the position of the flight due to the centrifugal forces caused by rotation. Schofield and Glikin (1962) considered the forces acting on a particle about to fall from the flight. The following expression for the kinetic angle of repose was obtained.

$$\phi = \tan^{-1}\left[\frac{\mu - Fr(\cos\theta - \mu\sin\theta)}{1 + Fr(\mu\cos\theta + \sin\theta)}\right] \qquad (6)$$

$$\text{where} \qquad Fr = \frac{D\omega^2}{2g} \qquad (7)$$

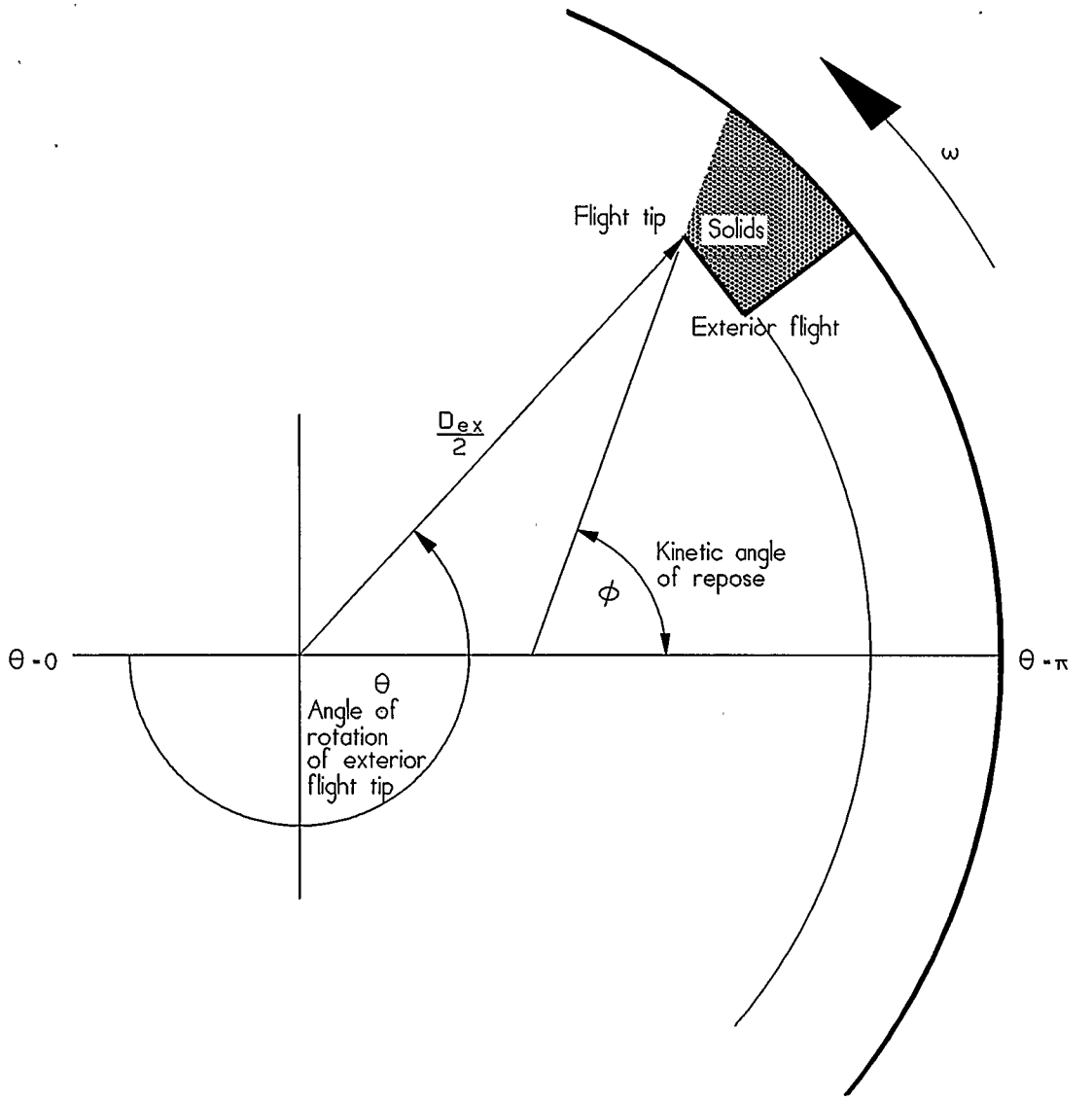


Figure 2.1 A discharging exterior flight in a rotating drum with its tip at angle of rotation θ

The kinetic coefficient of friction, μ, is a characteristic of the particle type. Kelly (1969) built an apparatus to determine the coefficient of friction and found the above equation to be valid for a large range of the Froude number, *Fr*, which is the ratio of the centrifugal to gravitational forces at the drum periphery.

2.3 Distribution of Particles by the Flight

The flight shape or profile for a particular drum can vary greatly. Several different flight profiles are illustrated in Figure 2.2. Each type of flight has its own pattern for distributing the falling particles across the drum. Glikin (1970) examined the distribution pattern of the simple two-sided right angle flight (a). Most industrial drums utilize some type of two-sided flight.

Kelly noted that most flights used in industry are far from optimum. Not only is the distribution of the particles across the drum poor, but often the flights are emptied near the top of the drum, leaving a large volume of the drum vacant of falling particles.

Kelly proposed several, more efficient flights such as the equal angular distribution (EAD) flight (b) and the centrally biased distribution (CBD) flight (c). Even though these flights furnished a more symmetric distribution of the particles across the drum, they were not claimed to be optimum flights. Kelly could not state a criterion for an optimum flight profile.

Kelly did concede that if the particles are not free flowing, then simpler industrial flights must be used. The cost of construction and maintenance may also influence the flight profile chosen for a particular design. Therefore, numerous flight profiles and,

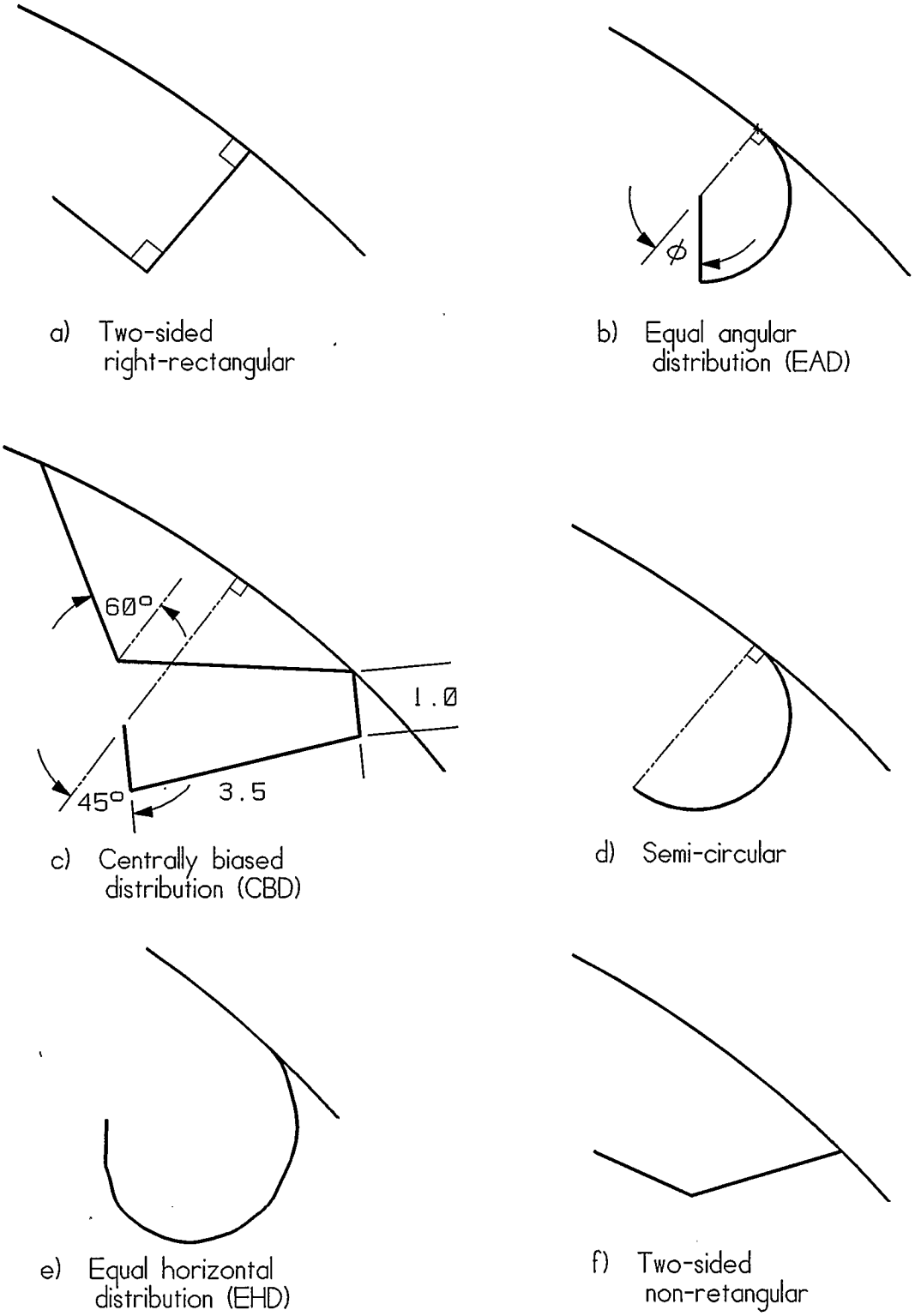


Figure 2.2 Some possible flight profiles

consequently, various distribution patterns are possible.

The flight profile is not the only factor which influences the distribution of the particles across the drum. Distribution also depends on the loading of the drum. *Design loading* of a drum is defined as the loading which is necessary such that the flights start to discharge just as their tips cross the horizontal centreline i.e. the angle of rotation θ is equal to π.

In an *overloaded drum*, some particles slip past each flight tip as it approaches the horizontal centreline. In an extremely overloaded drum, a rolling bed of particles, which can cover several flights, exists at the bottom of the drum. In any case, the degree to which a drum is overloaded does not effect the distribution pattern of the particles by the flights.

In an *underloaded drum*, however, the flights do not start discharging until after the horizontal centreline is crossed. This causes a portion of the drum to be void of any falling particles. Kelly commented that underloading should be avoided because of the inefficiency caused by the uneven distribution of particles. Kamke (1984), however, argued that drums should be operated in an underloaded condition. He claimed, due to overloading, some particles passed through the drum without ever being caught in a flight and cascaded, causing the product uniformity to be sacrificed. Overloading a drum may also unnecessarily increase the power required to rotate the drum.

In any event, it is possible for drums to be operated either in an underloaded or overloaded condition. The distribution pattern of the flights will be the same for design loaded and overloaded conditions. Underloading, however, will alter the distribution

pattern.

2.4 Advance of Particles

The cascade cycle of a particle consists of a number of stages. The particle spends a period of time retained in a flight while it is lifted from the lower half of the drum to where it is discharged from the flight. The next stage is the time the particle spends falling. After falling, the particle may bounce several times before coming to rest in a flight in the lower portion of the drum. In an underloaded or design loaded drum, the particle will remain in the flight to be lifted to the upper half of the drum to begin the cycle again and again. In an overloaded drum, the particle may spill from the flight before being carried to the upper half of the drum. The particle may slip past several flights before finally being lifted to the upper half of the drum to repeat the cycle.

The particle may be advanced along the length of the drum during each stage of its cascade cycle. The advance during each stage contributes to the overall rate that the particle passes through the drum. The possible modes of advancing are:

a) advance during the fall due to the drag forces of the gas stream and the incline of the drum,

b) advance during bouncing due to the forward momentum of the falling particle and the incline of the drum, and

c) advance while retained in the flights or in the bed of an extremely overloaded drum, due to the rolling motion of the particles, known as *kiln action.*

The total advance of a particle during a cascade cycle is the sum of the advance

during each stage of the cycle.

2.4.1 Advance during the Fall

Schofield and Glikin (1962) considered the behaviour of a particle as it falls from the flight. The particle was assumed to fall without interference from the other particles. The forces acting on a falling particle are shown in Figure 2.3.

Previous authors have used various conventions for selecting the positive direction for the axial movement. In this study, the sign convention is that the direction of the gas flow determines the positive axial direction. A positive incline of the drum is shown in the figure.

Ignoring vertical drag, a force balance in the axial direction (angle α from horizontal) for a particle falling through an axially flowing gas steam gave:

rate of momentum change in x-direction	=	force on particle due to gravity in x-direction	+	drag force on particle in x-direction

$$\frac{\pi}{6}D_p^3\rho_p\frac{dV_x}{dt} = \frac{\pi}{6}D_p^3\rho_p g\sin\alpha + \frac{\pi}{4}D_p^2C_D\rho_g\frac{(V_g - V_x)^2}{2} \qquad (8)$$

The drag coefficient for a sphere in motion in a gas stream was determined from a relationship developed by Schiller and Naumann (1933).

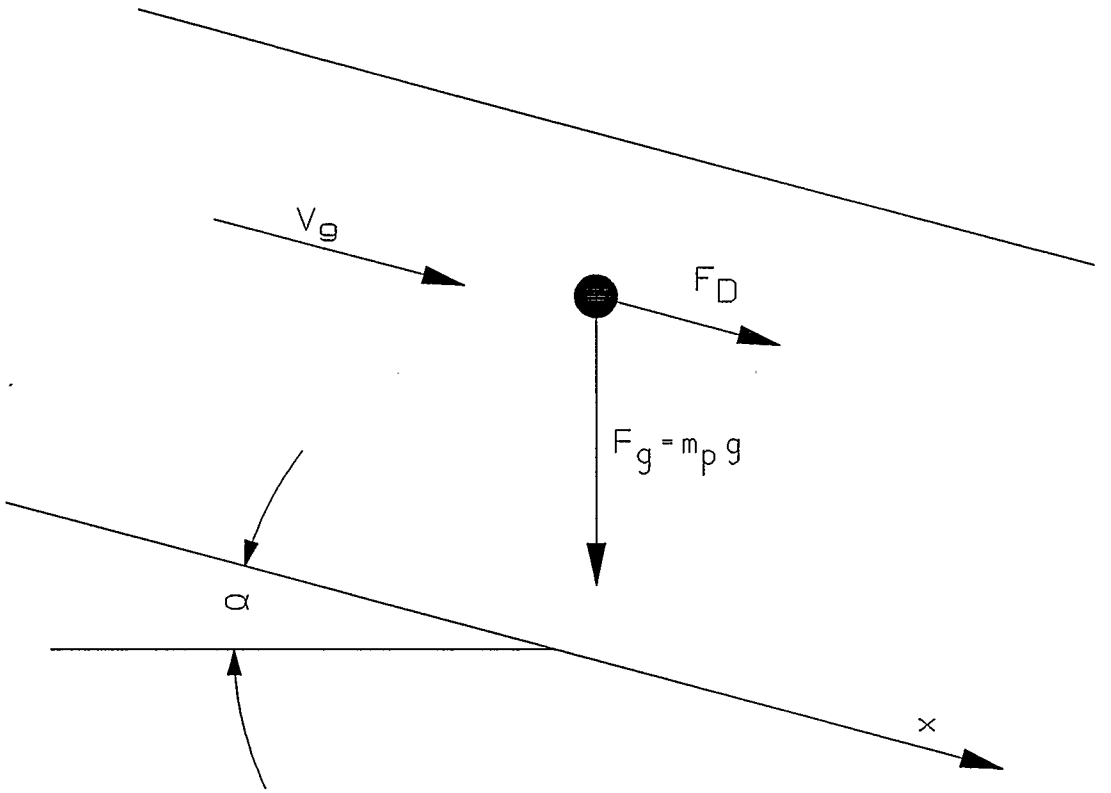


Figure 2.3 Forces on a falling particle

$$\text{if} \quad Re_p < 0.2 \quad \text{then} \quad C_D = \frac{24}{Re_p} \tag{9}$$

$$\text{if} \quad 0.2 < Re_p < 1000 \quad \text{then} \quad C_D = \frac{24}{Re_p}\left(1.0 + 0.15Re_p^{0.687}\right) \tag{10}$$

$$\text{where} \quad Re_p = \frac{\rho_g D_p (V_g - V_x)}{\mu_g} \tag{11}$$

By simplifying Equation 8, the acceleration of the falling particle in the axial direction was obtained.

$$\frac{dV_x}{dt} = g\sin\alpha + \frac{3C_D\rho_g}{4D_p\rho_p}(V_g - V_x)^2 \tag{12}$$

Assuming the initial velocity of the particle is zero, Schofield and Glikin integrated Equation 12 twice to get the displacement of the fallen particle in the axial direction for a drum inclined angle α to horizontal when angle α is negative.

$$\delta_1 = (V_g - a)t_1 - \frac{1}{K}\ln\left[\frac{1}{2a}\left(a + V_g + (a - V_g)\exp(-2aKt_1)\right)\right] \tag{13}$$

$$\text{where} \quad K = \frac{3C_D\rho_g}{4D_p\rho_p} \tag{14}$$

$$\text{and} \qquad a = \sqrt{\frac{g \sin|\alpha|}{K}} \tag{15}$$

When drum incline α is negative, the contribution to the net displacement by the drag force and the incline are in opposite directions. Drum incline α is negative in countercurrent drums where the net displacement of the solids is normally negative.

If the drum incline α is positive, which is the case for inclined cocurrent drums, the individual contributions to the net displacement are both positive or in the same direction as the gas flow. Kamke (1984) integrated Equation 12 twice to get the following equation for the distance advanced by a fallen particle in a drum with positive incline.

$$\delta_1 = V_g t_1 + \frac{1}{K} \ln \left[\frac{\cos\left[\tan^{-1}\left[\frac{V_g}{a} \right] \right]}{\cos\left[-aKt_1 + \tan^{-1}\left[\frac{V_g}{a} \right] \right]} \right] \tag{16}$$

Kamke also derived an expression for the distance advanced by a fallen particle for a drum with no incline, $\alpha = 0$. The equation is applicable to horizontal cocurrent drums.

$$\delta_1 = V_g t_1 + \frac{1}{K} \ln \left[\frac{1}{KV_g t_1 + 1} \right] \tag{17}$$

Parameters K and a for Equations 16 and 17 are given by Equations 14 and 15

above.

As a first approximation, Schofield and Glikin simplified Equation 12 by assuming the axial velocity of the falling particle, V_x, was much smaller than the gas velocity, hence, it could be neglected in the determination of the drag force on the particle. Integration of the simplified Equation 12 gave the following equation for the displacement.

$$\delta_1 = \frac{1}{2} g t_1^2 \sin\alpha + \frac{1}{2} K V_g t_1^2 \qquad \textbf{(18)}$$

The first term of Equation 18 is the displacement due to the incline of the drum. The second term is the displacement due to the drag forces of the gas stream. Equation 18 can be applied to both positive and negative inclines.

Schofield and Glikin, Kelly, and Kamke, all assumed the upward resistance of the air on the falling particle to be negligible. Therefore, the time of fall was:

$$t_1 = \sqrt{\frac{2y}{g}} \qquad \textbf{(19)}$$

O'Donnell (1975) included the drag forces in the vertical direction for his determination of the fall time. In most cases, however, the fall time is short and Equation 19 is adequate.

2.4.2 Advance due to Bouncing and Kiln Action

The advance of the particles when they are not falling is avoided in most studies of flighted rotating drums. Most authors limited their models to design loaded or underloaded drums, where most of the axial movement of the solids occurs in the airborne phase. From experiments with a small dryer, Kelly noticed that modes of movement other than by falling can be significant for some conditions. Kelly's investigation showed that the axial rate due to other modes was related to the rotational speed, the drum incline and to the degree of drum loading.

O'Donnell (1975) introduced the concept of the particle bouncing. After falling, the particle may bounce several times. On each bounce the particle will continue to advance. O'Donnell developed computer routines to describe the distance advanced by an individual particle from both bouncing and kiln action. O'Donnell's results indicated that bouncing could account for up to half the total axial movement in an underloaded drum. Kiln action could account for most of the axial movement in an overloaded drum.

De and Mukherhee (1975) studied the flow rate of the bed in flighted drums, specifically the effect of circular dams at the drum discharge. They developed a theoretical method to relate the bed holdup to the flow rate. The model gave poor agreement to their experimental results. They believed that, because of some of their assumptions, that their method was more applicable to drums without flights.

Sheikh (1987) and Matchett and Baker (1987) both used the following empirical equation for the mass flow rate of the particles advancing while not falling.

$$F_2 = a_m n D \tan\alpha \; H_2 \rho_b \qquad \textbf{(20)}$$

The coefficient, a_m, which was called the *dense phase velocity number*, was determined experimentally in small flighted drums with zero gas flow. For underloaded and design loaded drums, the coefficient was found to vary with particle type and number of flights, but was independent of the rotational speed, drum diameter and inclination angle. Equation 20 was unable to predict the flow in overloaded drums. Notice that, according to Equation 20, the flow is zero for a horizontal drum. All of Sheikh's experiments were for inclined drums.

2.4.3 Advance due to Kiln Action in Drums without Flights

Although treatment of the dense phase in a flighted drum has met minimal success, axial transport has been more extensively analyzed for a bed in a rotating drum without flights. Figure 2.4 illustrates the bed in a flightless rotating cylinder. The behaviour of the particles in the transverse plane is a rolling action. Particles come to rest near the wall (A). They rotate with the wall until they emerge at the top of the bed (B). At this point, they slide down the surface of the bed. The particles come to rest near the wall (A′) and begin the cycle again.

While the particles are at rest relative to the rotating wall, there is no displacement in the axial direction. Axial displacement occurs as the particles slide down the upper surface of the bed. The distance advanced δ_2 depends on both the drum incline α and the slope of the bed surface with respect to the drum axis γ.

Saeman (1951) published an analytical expression for the axial advance of the bed in an inclined kiln. At about the same time, Vahl and Kingma (1952) published the same

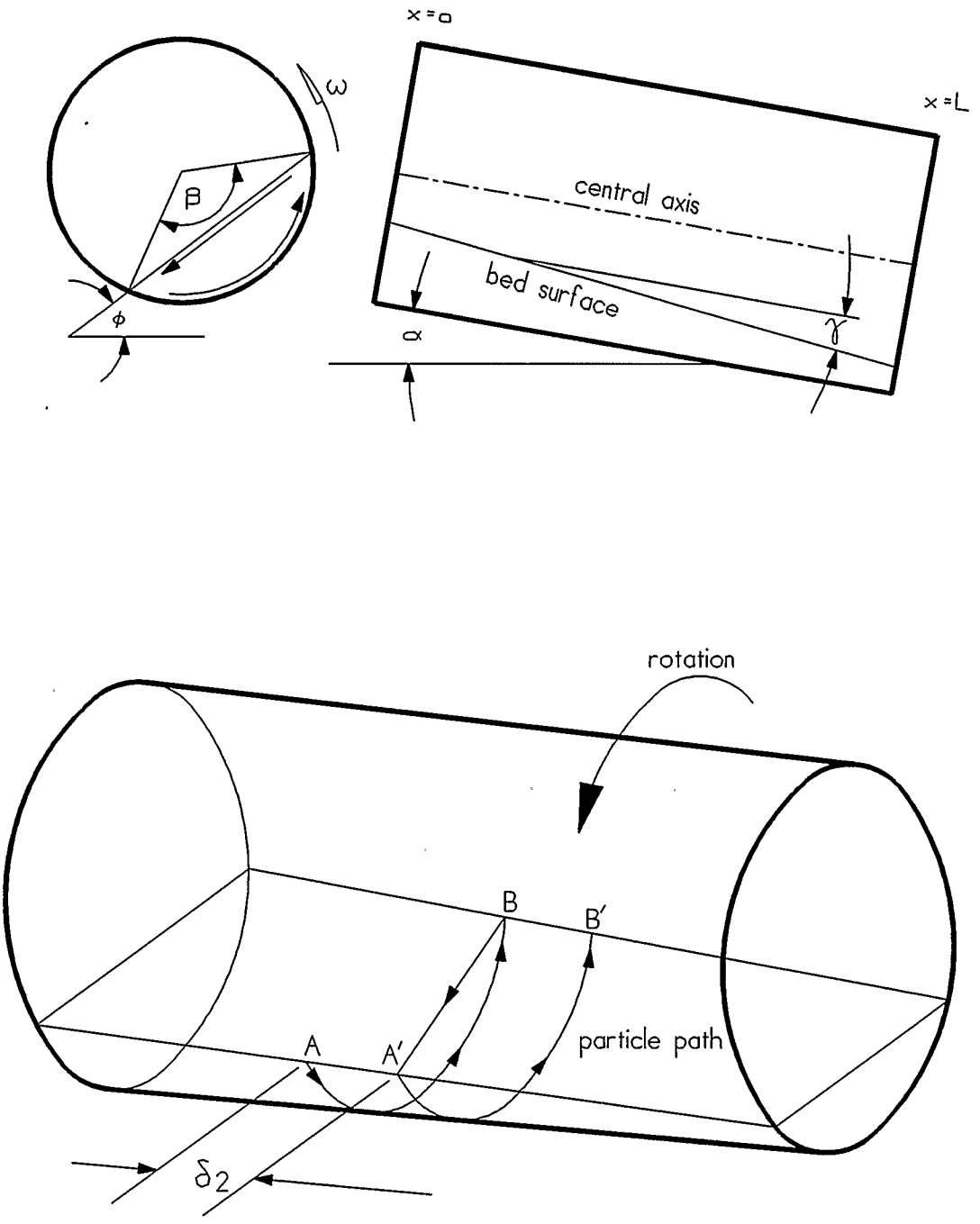


Figure 2.4 Advance in the bed of a rotating drum without flights

expression as Saeman but for horizontal drums only. Kramer and Croockewit (1952) augmented the work by Vahl and Kingma to get the same result as Saeman for an inclined kiln. The axial distance advanced by a single sliding particle is:

$$\delta_2 = \frac{\overline{BA'}(\alpha + \gamma\cos\phi)}{\sin\phi} \tag{21}$$

where $\overline{BA'}$ is the distance the particle slides before re-entering the bed. The time the particle spends at rest relative to the wall is assumed to be much larger than the time the particle spends sliding on the surface. The average axial advance and the rate of emerging particles were combined to obtain the axial flow rate of the bed.

$$F_2 = \frac{\pi}{6} n D^3 \rho_b \left[\frac{\alpha + \gamma\cos\phi}{\sin\phi}\right] \sin^3\left[\frac{\beta}{2}\right] \tag{22}$$

where the angle of fill, β, is related to the surface cord length, $\overline{BA'}$, by the following equation.

$$\overline{BA'} = D \sin\left[\frac{\beta}{2}\right] \tag{23}$$

Kramer and Croockewit found that Equation 22 gave good agreement with experimental results on a small 0.2 metre diameter drum.

Hogg, Shoji and Austin (1974) derived an equation for the special case of a horizontal drum with a variable fill level but also allowed for the time the particle spent

in the sliding layer. A coefficient, B, was introduced as the fraction of the cycle time, which is the rest time plus the sliding time. From experiments by Tscheng (1978), the time fraction, B, is related to the Froude number, in the range for most drums, by the following equation.

$$B = 1.0 - 0.807Fr^{0.196} \quad \text{when} \quad 0.0025 < Fr < 0.04 \tag{24}$$

For most drums, the Froude number is sufficiently small that the time the particles spend in the sliding layer is negligible and Equation 22 is adequate for predicting the advance of the solids in a drum without flights.

2.5 Residence Time Models

Schofield and Glikin (1962) were the first to introduce the concept that the mean residence time of a particle in a rotating drum is the product of the number of cascade cycles and the average time per cycle. The number of cycles is equal to the total length divided by the average advance for a cycle. Therefore, the residence time is:

$$\text{Residence time} = \frac{\text{(Drum length) (Average cycle time)}}{\text{(Average cycle advance)}}$$

$$\text{or} \quad T = \frac{L_e \, t_{ave}}{\delta_{ave}} \tag{25}$$

Most of the recently published models for the residence time of particles in a rotating drum are based on Equation 25. Assumptions vary and the methods used to determine the averages also vary.

2.5.1 Models without Bouncing or Kiln Action

Several of the existing models assume that the only time the particles advance axially is while they are falling. From Equation 25, Schofield and Glikin derived the following expression for the residence time in a flighted rotating drum:

$$T = \frac{L}{y_{ave}\left[\sin\alpha + \frac{KV_g^2}{g}\right]}\left[\sqrt{\frac{2y_{ave}}{g}} + \frac{\pi}{\theta_{ave}n}\right] \qquad \textbf{(26)}$$

The equation can be applied to inclined or horizontal drums. Flow may be either cocurrent or countercurrent; however, in the countercurrent case the sign is negative. The drum must be design loaded and have an open centre. The average fall distance, y_{ave}, and the average discharge angle, θ_{ave}, are weighted averages determined from a drawing of the drum cross section. The volume of the particles on a discharging flight at equal intervals of rotation is determined from the drawing. The fall distance and the angle at each interval are weighted by the volume discharged during the interval. The method could be used for any flight profile. Application of the method to an industrial-sized dryer was demonstrated, but predictions were not compared to any experimental results.

Kamke (1984) developed a computer program for a residence time model for underloaded or design loaded drums. The drum could have an open centre or a centrefill with flights. Kamke has been the only author to consider a centrefill. The drum could be inclined or horizontal but flow had to be cocurrent. Right-angled, two-sided flights were used.

The average cycle advance and the average cycle time were determined by first computing the volumetric flight holdup at one degree increments of the rotation angle. The change in flight holdup volume was used to weight the advance and the fall time in the calculation of the averages.

Kamke used his model to predict residence times for a medium-sized horizontal cocurrent rotary dryer. Kamke was also the only author to consider a feed with a distribution of particle sizes. From his experiments with an industrial wood chip dryer, he reported that the actual residence times of the finer particles were longer than predicted and the residence times of the coarser particles were shorter than predicted. The particles seemed to behave as a group which could be represented by a mean particle diameter. The diameter was a weight-mean determined by:

$$\overline{D_p} = \sum_{i=1}^{N_p} x_i D_{pi} \qquad (27)$$

Kamke's residence time model is summarized by Kamke and Wilson (1986).

2.5.2 Models with Bouncing or Kiln Action

Kelly (1969) reported a residence time model similar to Schofield and Glikin's for an open drum with EAD flights. For EAD flights, the average fall distance, y_{ave}, and the average discharge angle, θ_{ave}, can be determined analytically. Kelly's model allowed for underloaded and overloaded conditions.

Kelly compared his model to actual residence times for a small drum. The theoretical model gave results for the displacement due to the gas stream drag that were 20 to 100 per cent too high. He attributed the failure of the theoretical model to predict experimental results to the particles falling from the flights as sheets or curtains. A shielding effect of the other particles in the falling sheet from the main gas stream prevented a particle from being displaced as predicted. Kelly derived an empirical equation which related the average displacement of the falling particles to the gas velocity and the average fall distance.

$$\delta_1(\alpha=0) = 0.0396 V_g^{0.77} y_{ave}^{1.36} \tag{28}$$

A summary of Kelly's work is provided by Kelly and O'Donnell (1968).

O'Donnell (1975) extended Kelly's model. To allow for the screening effect of the falling particles, O'Donnell assumed that an above-average gas velocity, V_s, prevailed in the free space between sheets while the gas velocity within the falling sheet, V_c, was below the average value. The pressure drop in the sheet due to momentum transferred to the falling particles was assumed to equal to the pressure drop in the free space due to friction with the drum walls. Momentum transfer between the gas in the free space and the sheets was neglected. Equating the pressure drop in the free space to the pressure drop in the sheets yielded the following expression:

$$\frac{fLV_s^2}{D} = \frac{fLV_c^2}{D} + \frac{m_p}{m_a}\sum_{i=1}^{N_p}\left[\frac{x_f^2}{t_f}\right]_i \qquad (29)$$

A mass balance related the average gas velocity, the free space gas velocity and the curtain gas velocity.

$$V_g\left[\frac{\pi D^2}{4}\right] = V_c A + V_s\left[\frac{\pi D^2}{4} - A\right] \qquad (30)$$

The frontal area occupied by the sheets, A, was either estimated from photographs, approximated or assumed to be zero. Numerical integration and an iterative solution were required to determine the gas velocities and particle displacement.

O'Donnell found that the results still over-estimated the experimental data but were closer than previous models. He felt that a better theoretical model which accounted for the screening effect could be obtained. Kelly and O'Donnell (1977) summarize O'Donnell's model.

Thorne (1979) refined O'Donnell's model even further, reducing some of the complexity. Thorne did not allow for the screening effect of the sheets. Residence times predicted for Kelly's small countercurrent drum were too large. The difference increased with increasing gas velocity. A summary of Thorne's model was reported by Thorne and Kelly (1978). Models by Kelly, O'Donnell and Thorne were all developed for inclined drums with EAD flights and open centres.

Matchett and Baker (1987) proposed a slightly different approach to determining the residence time than previous authors. Simply, the residence time is equal to the holdup divided by the flow rate.

$$T = \frac{H}{F_0} \tag{31}$$

Although previous authors consider the same modes of movement, Matchett et al., explicitly state that the particles flow in two phases. The *airborne phase* consists of the falling particles which are advancing due to the drag force of the gas stream and the incline of the drum. The other phase is the remaining material, known as the *dense phase*, which rests in the flights or in the overload bed. The advance of the particles in the dense phase is by bouncing, rolling and sliding due to the incline of the drum or overload bed. Particle interchange freely between the two phases. The total holdup is the sum of the holdup of each phase.

$$H = H_1 + H_2 \tag{32}$$

where subscripts 1 and 2 refer to the airborne and the dense phase, respectively.

Similarly, the total flow rate is the sum of the flow rate in each phase.

$$F_0 = F_1 + F_2 \tag{33}$$

In subsequent development of the model of Matchett et al., Sheikh (1987)

concentrated on correlating an empirical equation for the flow rate of the dense phase F_2 with experiments using a small drum. He was only able to satisfactorily fit the data from underloaded, inclined drums. Matchett and Sheikh (1990) summarized Sheikh's original work.

Langrish (1989) developed the model of Matchett et al. even further by deriving sub-models to describe the thickness and density of a cascading sheet of particles and to determine the lower than average gas velocity within the sheets. Experiments with a small 0.237 metre diameter drum, showed that his attempts to account for the screening effect of the sheet over-compensated for this phenomena.

Models by Matchett et al., Sheikh and Langrish are only applicable to inclined drums with two-sided flights and open centres.

CHAPTER 3

MODEL DEVELOPMENT

All the residence time models for flighted rotating drums, developed in the past, consider a single or mean particle diameter. Most feedstocks for rotary drums, however, have a wide distribution of particle sizes. In a cocurrent drum, the fine particles are moved through the drum more swiftly than the coarse particles, causing the fine particles to have shorter residence times. In a countercurrent drum, residence times for the fine particles are longer than for the coarse particles.

In either case, the residence times vary with particle size. Because residence times vary, the concentration of each particle size in the drum is different than the concentration in the feed. This is significant when the rate of the mass and heat transfer processes occurring in the drum are also a function of particle diameter. For example, in a cocurrent dryer, the concentration of coarse particles in the drum is greater than in the feed. If the mass transfer of moisture from a particle is reduced for larger particles, then using the particle size distribution in the feed would cause a designer to over-estimate the rate of drying for the particles in the drum.

A residence time model that allows for a distribution of particle sizes in the feed would be useful for modelling processes that are strongly dependent on particle size and have feeds with a wide distribution of particle sizes.

Most of the previous authors commented on the reduced effect of the gas stream on the advance of the falling particles caused by the screening by other particles. This

screening effect is so significant that it should be accounted for in the model. To date, an acceptable method of allowing for the screening effect of the other particles in the falling sheets has not been reported.

As mentioned in the previous chapter, most authors ignored the axial movement of the dense phase. If the movement in the dense phase was considered, then it was treated unsatisfactorily using empirical equations. These empirical equations could only be used for the small drums to which they were fitted because they did not consider important parameters such as the distance to the discharge or the slope of the bed surface to the drum axis.

The remainder of this chapter concentrates on the two phase concept reported by Matchett and Baker (1987), Equations 31 through 33, to derive equations for the solids axial flow rate and the solids holdup in flighted rotating drums. The derivations attempt to be general enough that the resulting solids residence time model may be applied to a majority of drum designs. For example, the drum may be inclined or horizontal. Flow may be cocurrent or countercurrent. The drum may be operated in an underloaded, overloaded or design loaded condition. The drum may have any type of flight profile or configuration including a centrefill with flights.

The model will also be able to solve for the variation in residence times for a distribution of particle sizes. The residence time and the concentration in the drum of each particle size will be predicted by the model. The model will also allow for the screening effect of the sheets of falling particles.

3.1 Model Assumptions

In addition to the drum operating at a steady-state, a number of other assumptions are given below.

3.1.1 Flights

Although the flights may have any profile, a few assumptions have been made about the flight behaviour. The discharging flight must be empty on or before the flight tip reaches the opposite side of the drum. In other words, the exterior flight is emptied on or before angle $\theta = 0$. Similarly, an interior flight must be emptied on or before angle $\psi = \pi$, where ψ is the angle of rotation of an interior flight tip. The angle of rotation of an interior flight ψ is defined more completely in Section 3.8.

Also, the interior and exterior flights are sized such that the exterior flights cannot overfill a receiving interior flight. That is, an interior flight will not spill material before the tip reaches angle $\psi = 2\pi$.

3.1.2 Advance of Particles

The axial advance of material is the result of the flow of each of the two phases: flow in the airborne phase F_1 and flow in the dense phase F_2.

For the model to be developed, each flow is independent of the other, except that the flows must sum to the total flow rate F_0.

$$F_0 = F_1 + F_2 \tag{34}$$

In the airborne phase, fine particles and coarse particles may advance axially at different velocities. The axial velocity of the airborne particles is a function of the particle size. In the dense phase, though, the velocity is the same for all particles regardless of size.

3.1.3 Fall Time

The vertical drag force opposing the fall of the particle is assumed to be negligible compared to the force due to gravity. The fall time can be calculated using Equation 19 in the previous chapter. Although this assumption is certainly valid for short falls, it is also valid for longer falls when the shielding effect of the sheets also reduces the drag force on the particles in the vertical direction.

3.1.4 Constant Kinetic Angle of Repose

The usual range of the Froude number at the periphery of a rotary dryer is between 0.0025 and 0.04. The variation in the kinetic angle of repose for this range of Froude number can be determined from Equation 6.

$$\text{for } Fr = 0.04, \qquad \phi_{max} - \phi_{min} = 4.6 \text{ degrees}$$

$$\text{for } Fr = 0.0025, \qquad \phi_{max} - \phi_{min} = 0.3 \text{ degrees.}$$

Even for drums with the highest practical Froude number, the variation of the kinetic angle of repose is small. Therefore, for the model to be developed here, the

kinetic angle of repose is assumed to be constant.

3.1.5 Drum Loading

All the possible degrees of loading are considered. The possible degrees of loading of an open drum and a drum with a flighted centrefill are illustrated in Figures 3.1 and 3.2, respectively. Each loading condition can be characterized by the angle at which the exterior flights begin to discharge referred to as the *initial discharge angle* θ_i.

The definitions of underloaded and design loaded are basically the same as that used by other authors. In an underloaded drum, (a) in Figure 3.1, the initial discharge angle of the exterior flight is in the upper half of the drum, $\pi < \theta_i \leq 2\pi$. In a design loaded drum, the exterior flights begin to discharge just as the tips cross the horizontal centreline, $\theta_i = \pi$. In an overloaded drum (b and c), the exterior flights start to discharge material while in the lower half of the drum. A special case (c) exists if the drum is so overloaded that the material in the first discharging flight buries other flights closer to the horizontal centreline. Flights begin to be buried when the initial discharge angle is perpendicular to the angle of repose of the solids. Material in the discharging flight covers the tips of the proceeding flights. The covered flights cannot discharge material. Therefore, in an overloaded drum, the condition when the exterior flights have an initial angle of discharge which is less than $\phi+\pi/2$ must be considered separately from a drum with an initial angle of discharge of the exterior flight between $\phi+\pi/2$ and π.

Some of the loading conditions considered are pertinent to a drum with a flighted

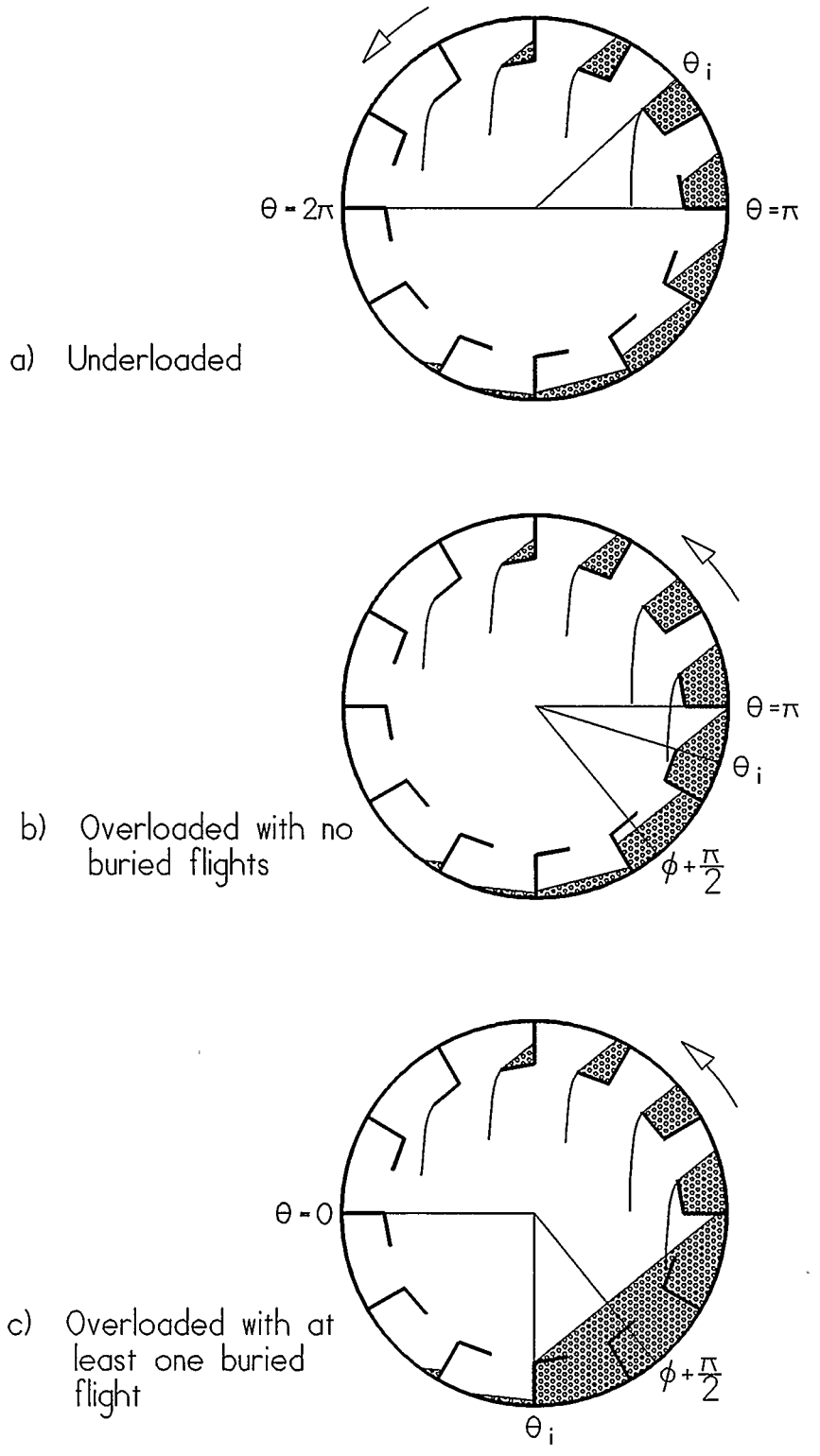


Figure 3.1 Possible degrees of loading in an open drum

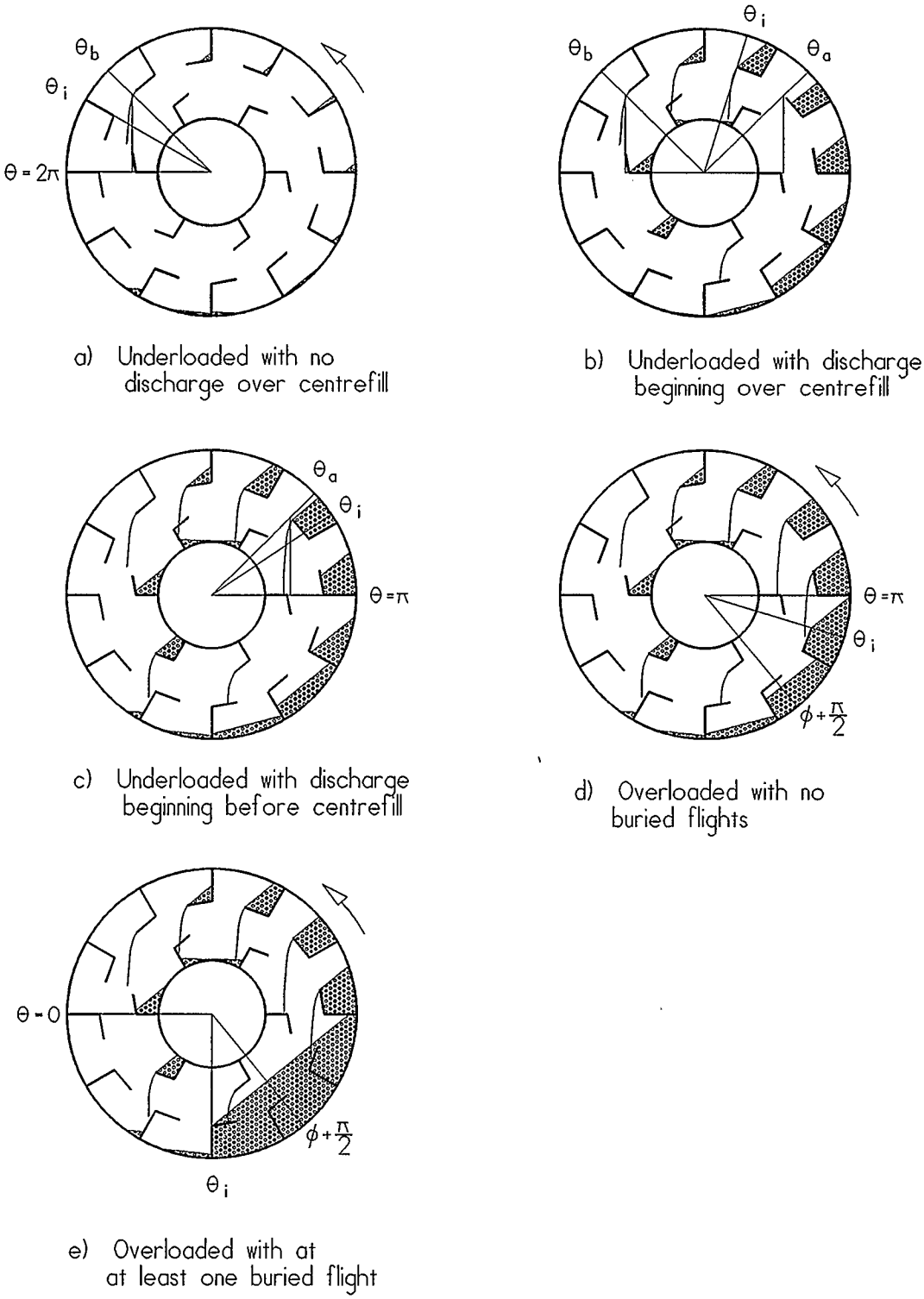


Figure 3.2 Possible degrees of loading in a drum with a flighted centrefill

centrefill only. In an underloaded drum with a centrefill, the angle that the exterior flights start to discharge is still between π and 2π, but the interior flights also have an initial discharge angle ψ_i which is between 0 and π.

The exterior flights can begin to discharge particles over the centrefill or on either side of the centrefill. The exterior flight is over the centrefill if it is between angles θ_a and θ_b as shown in the Figure 3.2. If the initial discharge angle θ_i is between θ_b and 2π (a), then no material is received by the interior flights. If θ_i is between θ_a and θ_b (b), then the exterior flights begin to discharge while over the centrefill. The amount of material received by interior flights depends on the location of θ_i between θ_a and θ_b. If θ_i is less than θ_a (c), then the amount of material received by interior flights does not depend on θ_i.

The angles θ_a and θ_b are determined by the following equations.

$$\theta_a = \pi + \cos^{-1}\left[\frac{D_{in}}{D_{ex}}\right] \tag{35}$$

and

$$\theta_b = \pi + \cos^{-1}\left[-\frac{D_{in}}{D_{ex}}\right] \tag{36}$$

where D_{ex} and D_{in} are the diameters of the exterior and interior flight tips respectively.

3.2 Flight Discharge Rate

Figure 3.3 presents cross sections of an exterior and an interior flight. The exterior flight is at some angle of rotation θ, which is between θ_i and 2π. The interior flight tip is at angle ψ between ψ_i and π. The rate of change in volume of the material in the discharging exterior flight at angle θ in the rotating drum is:

$$\frac{dh_{ex}}{dt} = \frac{\ell_{ex}^2}{2}\omega \qquad \theta_i \leq \theta \leq 2\pi \tag{37}$$

where ℓ_{ex} is the length of the upper surface of the material in the discharging exterior flight at angle θ.

Similarly, the volumetric rate of discharge of an interior flight at angle ψ is:

$$\frac{dh_{in}}{dt} = \frac{\ell_{in}^2}{2}\omega \qquad \psi_i \leq \psi \leq \pi \tag{38}$$

The angular rate of rotation is given by:

$$\omega = 2\pi n$$

By combining Equations 37 and 39 and the bulk density, the mass discharge rate from the exterior flight at angle θ is:

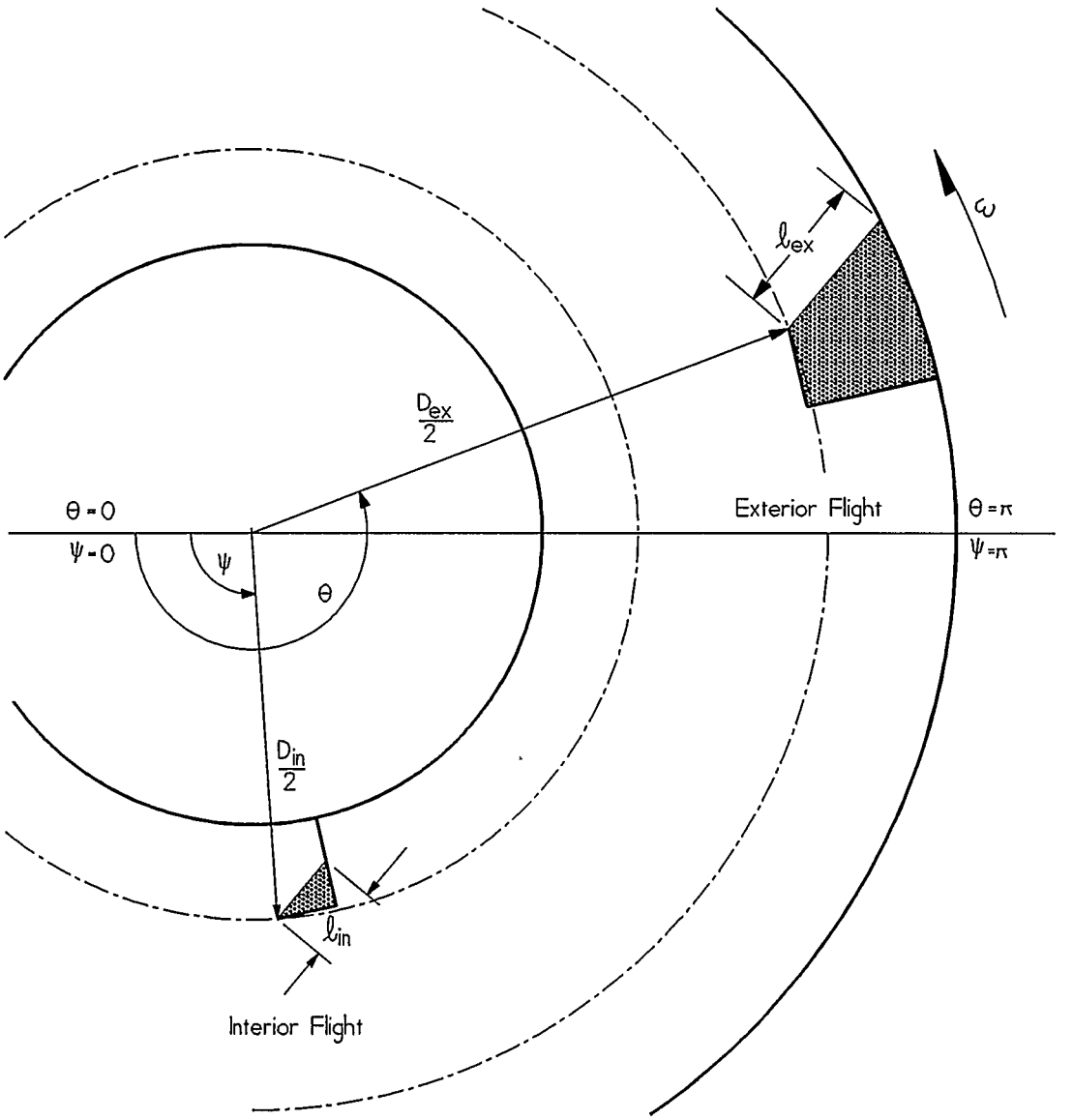


Figure 3.3 Rate of discharge for exterior and interior flights

$$\frac{dm_{ex}}{dt} = \pi n \rho_b \ell_{ex}^2 \qquad \theta_i \leq \theta \leq 2\pi \tag{40}$$

The corresponding expression for the mass discharge rate from an interior flight at angle ψ is:

$$\frac{dm_{in}}{dt} = \pi n \rho_b \ell_{in}^2 \qquad \psi_i \leq \psi \leq \pi \tag{41}$$

Equations 40 and 41 are key to the formulation of the model and are referred to several times in the rest of the chapter. Dimensions for the discharge rate are mass per unit time per unit of flight length.

3.3 Initial Discharge Angle for an Interior Flight

For a drum with a flighted centrefill, the quantity of material received by the interior flights between $\psi = \pi$ and $\psi = 2\pi$ is equal to the quantity of material discharged from the exterior flights between $\theta = \theta_a$ and $\theta = \theta_b$ The volume of material received by an interior flight can be determined by integrating the discharge rate of the exterior flight, Equation 37, from θ_a to θ_b and multiplying by the ratio of the number of exterior flights to the number of interior flights.

$$h_{in} = \frac{N_{ex}}{N_{in}} \int_{\theta_a}^{\theta_b} \frac{\ell_{ex}^2}{2} d\theta \tag{42}$$

The quantity of material discharged by the interior flight between angles $\psi = 0$

and $\psi = \pi$ is equal to the material discharged from the initial angle of discharge ψ_i to $\psi = \pi$. The quantity discharged can be found by integrating the discharge rate of the interior flight, Equation 38, from $\psi = \psi_i$ to $\psi = \pi$.

$$h_{in} = \int_{\psi_i}^{\pi} \frac{\ell_{in}^2}{2} d\psi \tag{43}$$

As long as the flight is not over filled, the quantity received by the interior flight will be equal to the quantity discharged by the same flight. Equations 42 and 43 can be equated to obtain the following equation.

$$\int_{\psi_i}^{\pi} \ell_{in}^2 d\psi = \frac{N_{ex}}{N_{in}} \int_{\theta_a}^{\theta_b} \ell_{ex}^2 d\theta \tag{44}$$

The above equation can be used to solve for the initial angle of discharge for an interior flight ψ_i.

3.4 Axial Movement of Falling Particles

The mass rate of movement in the axial direction for the material of particle size i discharging from an exterior flight at angle θ is the product of the flight discharge rate given by Equation 40, the axial advance during the fall of particle size i, $\delta_{1,ex,i}$, and the mass fraction of particle size i, $x_{d,i}$.

$$f_{1,ex,i} = x_{d,i}\pi n\rho_b \ell_{ex}^2 \delta_{1,ex,i} \tag{45}$$

where $\theta_i \leq \theta \leq 2\pi$ for the exterior flight.

The mass fraction of particles of size i, $x_{d,i}$, is not necessarily the same as the mass fraction of size i in the feed, $x_{f,i}$.

The average axial rate for material of particle size i discharged from an exterior flight rotated from 0 to 2π is:

$$\overline{f_{1,ex,i}} = \frac{\int_{\theta_i}^{2\pi} x_{d,i}\pi n\rho_b \ell_{ex}^2 \delta_{1,ex,i} d\theta}{\int_0^{2\pi} d\theta} \tag{46}$$

The above expression is simplified to:

$$\overline{f_{1,ex,i}} = \frac{1}{2} x_{d,i} n\rho_b I_{ex,i} \tag{47}$$

where

$$I_{ex,i} = \int_{\theta_i}^{2\pi} \ell_{ex}^2 \delta_{1,ex,i} d\theta \tag{48}$$

A similar expression which can be found for the average axial rate of falling particles of size i discharging from an interior flight rotating from $\psi = 0$ to $\psi = 2\pi$ is:

$$\overline{f_{1,in,i}} = \frac{1}{2} x_{d,i} n \rho_b I_{in,i} \tag{49}$$

where

$$I_{in,i} = \int_{\psi_i}^{\pi} \ell_{in}^2 \delta_{1,in,i} d\theta \tag{50}$$

The total axial movement of falling material of particle size i originating from both the exterior and interior flights is the sum of the products of the average transport rate for an exterior flight and the number of exterior flights and the product of the average transport rate for an interior flight and the number of interior flights. The total axial transport rate of falling material of particle size i is:

$$F_{1,i} = \frac{x_{d,i} n \rho_b}{2} \left[N_{ex} I_{ex,i} + N_{in} I_{in,i} \right] \tag{51}$$

A mass balance for particle size i gives the following:

$$x_{f,i} F_0 = x_{d,i} \left[\frac{n \rho_b}{2} \left[N_{ex} I_{ex,i} + N_{in} I_{in,i} \right] + F_2 \right] \tag{52}$$

Solving for $x_{d,i}$ gives:

$$x_{d,i} = \frac{F_0 x_{f,i}}{\frac{n\rho_b}{2}\left[N_{ex} I_{ex,i} + N_{in} I_{in,i}\right] + F_2} \tag{53}$$

But the mass fractions of all particle sizes in the drum must sum to 1.

$$\sum_{i=1}^{N_p} x_{d,i} = 1 \tag{54}$$

Combining Equations 53 and 54, and solving for the total flow rate produces the following equation:

$$\frac{1}{F_0} = \sum_{i=1}^{N_p} \frac{x_{f,i}}{\frac{n\rho_b}{2}\left[N_{ex} I_{ex,i} + N_{in} I_{in,i}\right] + F_2} \tag{55}$$

3.5 Axial Movement of Dense Phase

The axial movement of the dense phase can be separated into two situations; the advance of particles held in a discharging flight and the advance of particles held in a non-discharging flight.

3.5.1 Axial Movement on a Discharging Flight

Consider the discharging flight in Figure 3.4. As the drum rotates particles emerge at the bed surface, slide down the surface, and discharge over the flight tip. If

the drum is horizontal, the sliding particles will move on a plane perpendicular to the drum axis and will not be advanced axially. However, if the drum is inclined, the sliding will have an axial component which will contribute to the overall axial movement.

The mass rate of particles emerging to the surface and discharge from an exterior flight is given by Equation 40.

$$\frac{dm_{ex}}{dt} = \pi n \rho_b \ell_{ex}^2 \quad \text{when} \quad \theta_i \le \theta \le 2\pi \tag{56}$$

The mean distance that the emerging particles slide to reach the flight tip can be determined by considering the centroid of the wedge that is discharged for a small rotation $d\theta$. As $d\theta$ approaches zero, the mean distance particles travel to the flight tip is the distance from the centroid to the flight tip.

$$\overline{AB} = \frac{2}{3}\ell_{ex} \quad \text{when} \quad \theta_i \le \theta \le 2\pi \tag{57}$$

The mean distance advanced in the axial direction is the axial component of the above mean distance. It can be approximated by the following equation if the drum incline α is small.

$$\overline{\delta_{2,ex}} = \frac{2\alpha}{3\sin\phi}\ell_{ex} \quad \text{when} \quad \theta_i \le \theta \le 2\pi \tag{58}$$

The mass rate in the axial direction for a discharging exterior flight is the product

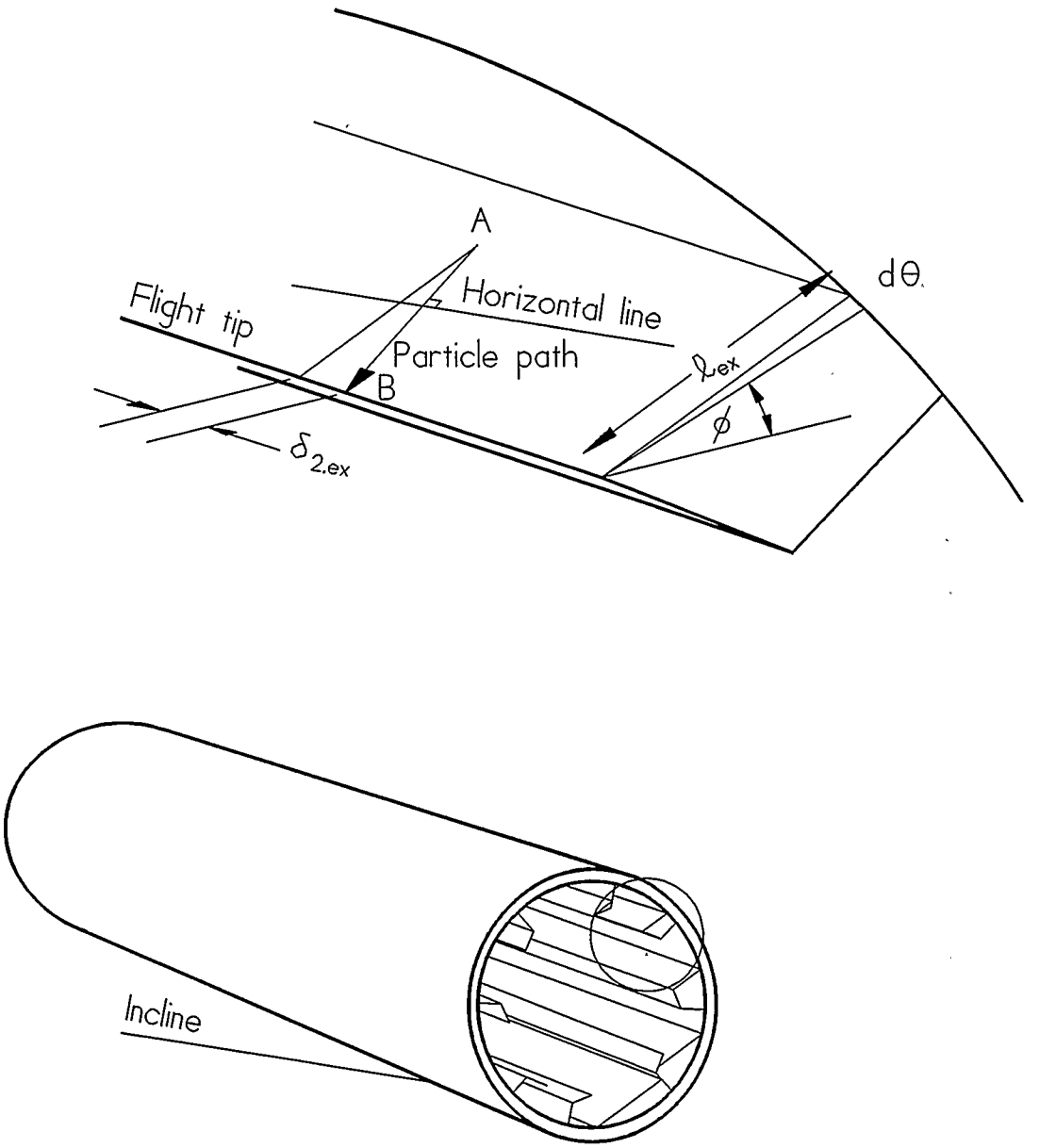


Figure 3.4 Axial movement of particles on a discharging flight

of the discharge rate, Equation 56, and the mean axial displacement, Equation 58.

$$f_{2,ex} = \frac{2\pi n \rho_b \alpha}{3\sin\phi} \ell_{ex}^3 \quad \text{when} \quad \theta_i \le \theta \le 2\pi \tag{59}$$

A similar equation can be derived for the axial flow rate of material discharging from an interior flight at angle ψ.

$$f_{2,in} = \frac{2\pi n \rho_b \alpha}{3\sin\phi} \ell_{in}^3 \quad \text{when} \quad \psi_i \le \psi \le \pi \tag{60}$$

3.5.2 Axial Movement on a Non-Discharging Flight

Figure 3.5 illustrates the dense phase in a non-discharging flight. Similar to the discharging flight, as the drum rotates, particles emerge at the bed surface and slide down the surface. In the non-discharging flight, particles re-enter the bed at the lower portion of the surface. This material in the flight has a rolling action similar to the rolling action in the bed of a drum without flights. In a flighted drum, however, there is a rolling bed in each of the non-discharging flights.

Because particles are not spilling over the flight tip, the surface in a non-discharging flight may not be parallel to the drum axis. Therefore, the particles in a non-discharging flight may move axially due to both the drum incline and to the slope of the bed surface to the drum axis.

The rate that particles emerge at the bed surface is

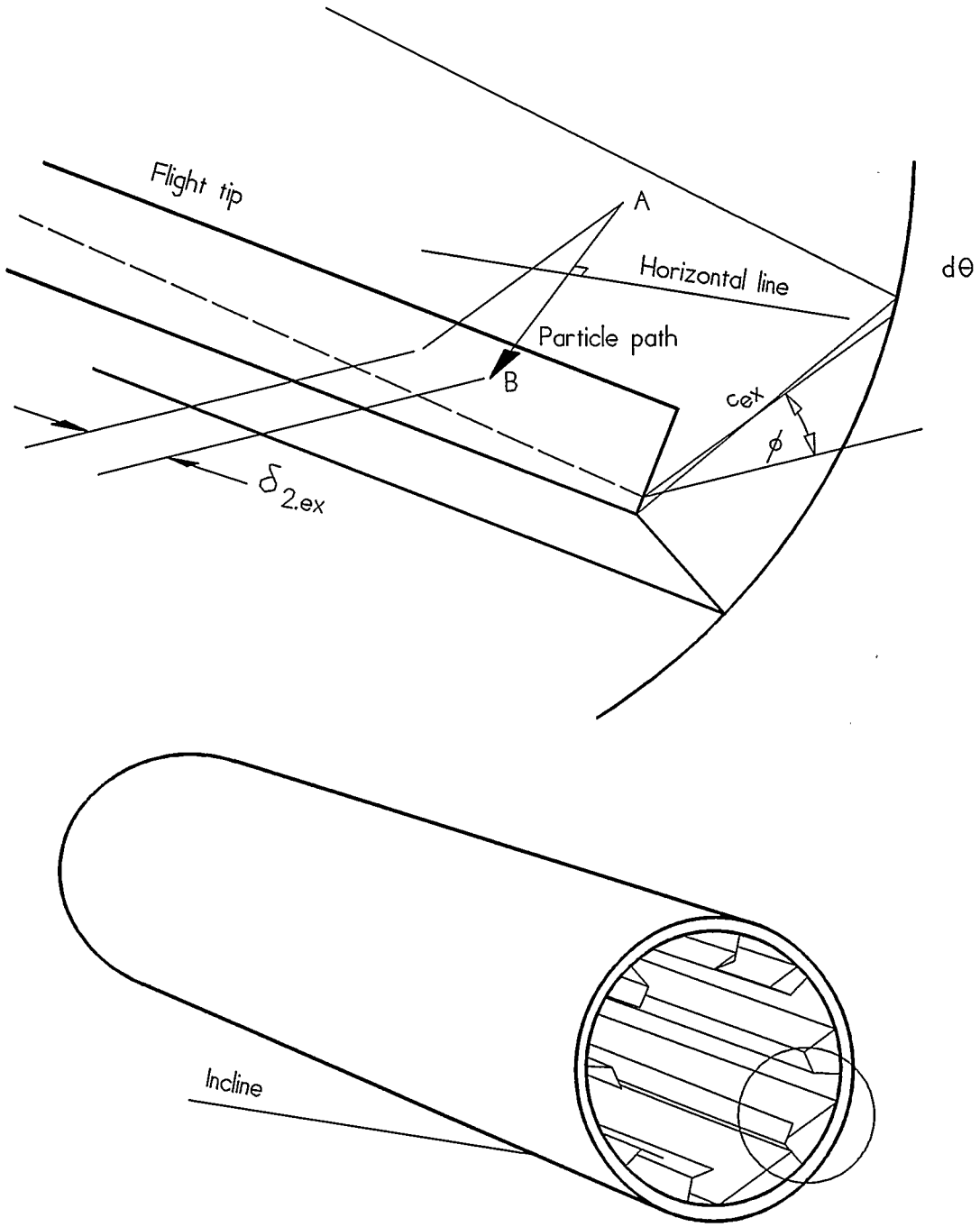


Figure 3.5 Axial movement on a non-discharging flight

$$\frac{dm_{ex}}{dt} = \pi n \rho_b \left[\frac{c_{ex}}{2}\right]^2 \qquad \text{when} \qquad 0 \le \theta \le \theta_i \tag{61}$$

where c_{ex} is the length of the surface on a non-discharging flight.

The mean distance the emerging particles slide before re-entering the bed is

$$\overline{AB} = \frac{2}{3} c_{ex} \qquad \text{when} \qquad 0 \le \theta \le \theta_i \tag{62}$$

If the drum incline α is small, then the mean distance advanced in the axial direction can be approximated by:

$$\delta_{2,ex} = \frac{2}{3} c_{ex} \left[\frac{\alpha + \gamma \cos\phi}{\sin\phi}\right] \qquad \text{when} \qquad 0 \le \theta \le \theta_i \tag{63}$$

where γ is the angle of the bed surface to the drum axis.

The mass rate in the axial direction for material on a single non-discharging exterior flight at an angle θ is the product of the emerging rate, Equation 61, and the axial displacement, Equation 63.

$$f_{2,ex} = \frac{1}{6} \pi n \rho_b \left[\frac{\alpha + \gamma \cos\phi}{\sin\phi}\right] c_{ex}^3 \qquad \text{when} \qquad 0 \le \theta \le \theta_i \tag{64}$$

If the bed surface is not sloped to the drum axis, $\gamma = 0$, then Equation 64 becomes:

$$f_{2,ex} = \frac{1}{6}\pi n \rho_b \frac{\alpha}{\sin\phi} c_{ex}^3 \quad \text{when} \quad 0 \le \theta \le \theta_i \quad \text{and} \quad \gamma = 0 \tag{65}$$

Similarly, the mass rate in the axial direction for material on a single non-discharging interior flight at angle ψ is

$$f_{2,in} = \frac{1}{6}\pi n \rho_b \left[\frac{\alpha + \gamma\cos\phi}{\sin\phi}\right] c_{in}^3 \quad \text{when} \quad 0 \le \psi \le \psi_i \quad \text{and} \quad \pi \le \psi \le 2\pi \tag{66}$$

3.5.3 Total Axial Movement of Dense Phase

The total axial rate of the dense phase in a flighted drum is the average dense phase axial rate in an exterior flight multiplied by the number of exterior flights plus the average dense phase axial rate in an interior flight multiplied by the number of interior flights.

$$F_2 = N_{ex}\overline{f_{2,ex}} + N_{in}\overline{f_{2,in}} \tag{67}$$

The average axial flow rate of the dense phase of an exterior flight is

$$\overline{f_{2,ex}} = \frac{\int_0^{2\pi} f_{2,ex} d\theta}{\int_0^{2\pi} d\theta} \tag{68}$$

The average axial flow rate of the dense phase of an interior flight is

$$\overline{f_{2,in}} = \frac{\int_0^{2\pi} f_{2,in} d\theta}{\int_0^{2\pi} d\theta} \tag{69}$$

In order to derive an equation for the total axial movement of the dense phase, each possible drum loading condition is considered separately. Figure 3.6 illustrates the definition of three more angles. Angle θ_d is the exterior flight angle which is directly below the initial discharge angle of the interior flights ψ_i. Angle θ_f is the exterior flight angle which is directly below the initial discharge angle of the exterior flights in an underloaded drum. Angle ψ_a is the interior flight angle directly below the initial discharge angle of the exterior flights in an underloaded drum. Equations for determining these angles are:

$$\theta_d = \cos^{-1}\left[\frac{D_{in}}{D_{ex}}\cos\psi_i\right] \tag{70}$$

$$\theta_f = 2\pi - \theta_i \tag{71}$$

$$\psi_a = \pi + \cos^{-1}\left[\frac{D_{ex}}{D_{in}}\cos(\theta_i - \pi)\right] \tag{72}$$

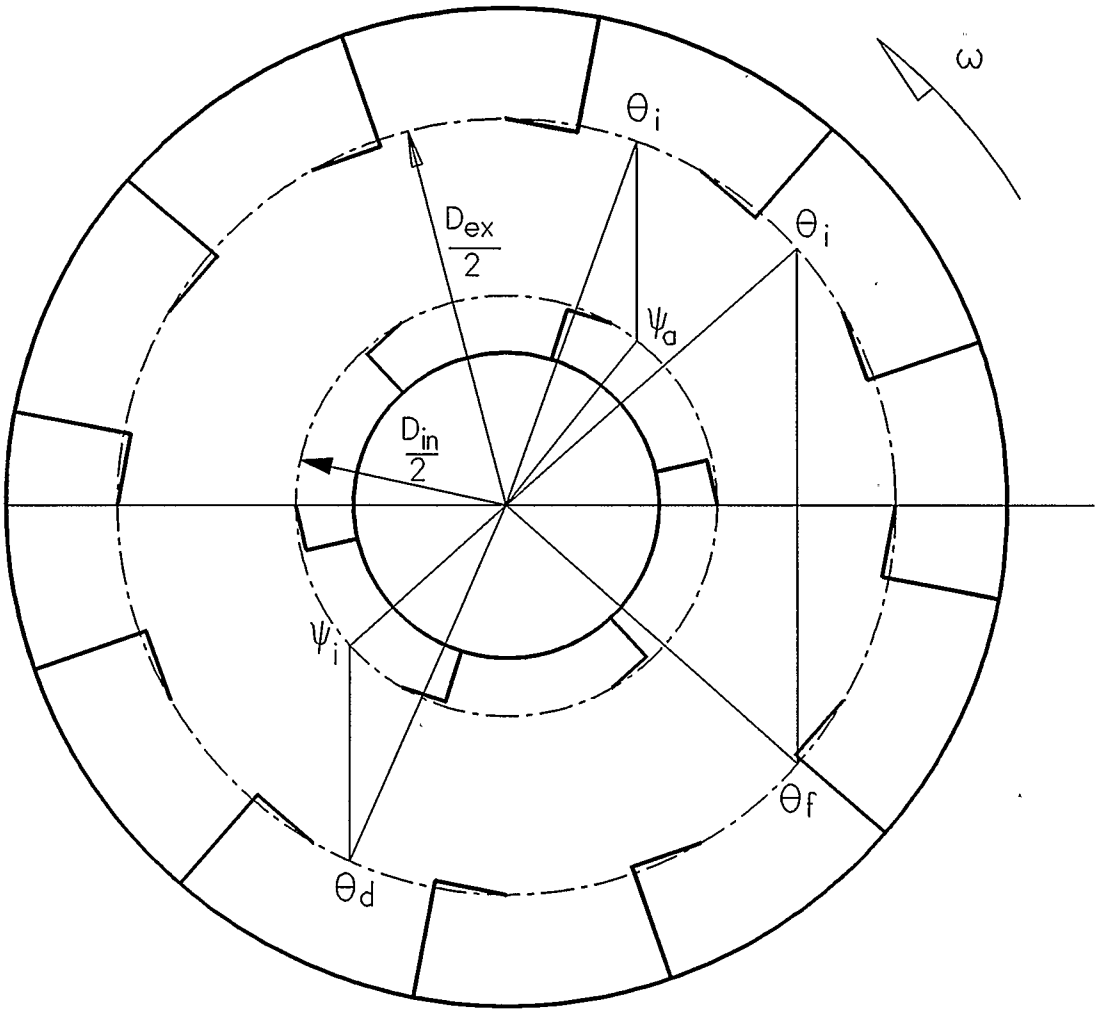


Figure 3.6 Definition of angles θ_d, θ_f and ψ_a

For an underloaded drum with an initial discharge angle of the exterior flights over a centrefill, the following behaviour is observed for an exterior flight during one revolution. Starting at angle $\theta = 0$, the flight is empty. As the drum rotates, the empty flight receives particles which have fallen from the discharging flights above. From angle $\theta = 0$ to the angle which is one flight spacing less than the angle directly below the interior initial discharge angle $\theta = \theta_d - 2\pi/N_{ex}$, the exterior flight is not discharging. Also between these angles, the bed slope relative to the drum axis γ is zero. Equation 65 describes the axial movement for this interval.

In the same drum, while the exterior flight revolves between $\theta_d - 2\pi/N_{ex}$ and the initial discharge angle θ_i, it is still not discharging, but the bed surface may be sloped relative to the drum axis. Therefore, Equation 64 describes the axial movement during this interval. During the rest of the revolution from θ_i to 2π, the flight is discharging and Equation 59 is applicable.

The average axial movement in the exterior flight in the underloaded drum with an initial discharge angle over the centrefill is obtained using Equation 68.

$$\overline{f_{2,ex}} = \frac{n\rho_b}{12}\left[\frac{\alpha}{\sin\phi}\int_0^{\theta_d - 2\pi/N_{ex}} c_{ex}^3 d\theta + \int_{\theta_d - 2\pi/N_{ex}}^{\theta_i} \frac{\alpha+\gamma\cos\phi}{\sin\phi} c_{ex}^3 d\theta + \frac{4\alpha}{\sin\phi}\int_{\theta_i}^{2\pi} \ell_{ex}^3 d\theta\right] \quad (73)$$

Next, considering an interior flight in the same underloaded drum during a revolution reveals that from $\psi = 0$ to $\psi = \psi_i$, the flight is non-discharging and the bed level could be sloped relative to the drum axis. From $\psi = \psi_i$ to $\psi = \pi$, the flight is discharging. From $\psi = \pi$ to the angle ψ_a, where particles are received from the exterior

flights which begin to discharge at θ_i, the interior flight is empty. From $\psi = \psi_a$ to $\psi = 2\pi$, the flight is non-discharging and the bed surface could be sloped with respect to the drum axis. The average axial movement in an interior flight in an extremely underloaded drum is obtained using Equation 69.

$$\overline{f_{2,in}} = \frac{n\rho_b}{12}\left[\int_0^{\psi_i} \frac{\alpha+\gamma\cos\phi}{\sin\phi} c_{in}^3 d\psi + \frac{4\alpha}{\sin\phi}\int_{\psi_i}^{\pi} \ell_{in}^3 d\psi + \int_{\psi_a}^{2\pi} \frac{\alpha+\gamma\cos\phi}{\sin\phi} c_{in}^3 d\psi\right] \tag{74}$$

Combining Equations 67, 73, and 74 gives the total axial movement of the dense phase in an underloaded drum in which the discharge of the exterior flights begins over the centrefill.

$$F_2 = \frac{n\rho_b}{12}\left[N_{ex}\left[\frac{\alpha}{\sin\phi}\int_0^{\theta_d-2\pi/N_{ex}} c_{ex}^3 d\theta + \int_{\theta_d-2\pi/N_{ex}}^{\theta_i} \frac{\alpha+\gamma\cos\phi}{\sin\phi} c_{ex}^3 d\theta + \frac{4\alpha}{\sin\phi}\int_{\theta_i}^{2\pi} \ell_{ex}^3 d\theta\right]\right. \tag{75}$$
$$\left. + N_{in}\left[\int_0^{\psi_i} \frac{\alpha+\gamma\cos\phi}{\sin\phi} c_{in}^3 d\psi + \frac{4\alpha}{\sin\phi}\int_{\psi_i}^{\pi} \ell_{in}^3 d\psi + \int_{\psi_a}^{2\pi} \pi\frac{\alpha+\gamma\cos\phi}{\sin\phi} c_{in}^3 d\psi\right]\right]$$

The following approximations are made for the surface length c on the non-discharging flights:

$$c_{ex} \simeq 0 \quad \text{when} \quad 0 \le \theta \le \theta_d - 2\pi/N_{ex} \tag{76}$$

$$c_{ex} \simeq \ell_{ex,\theta_i} \quad \text{when} \quad \theta_d - 2\pi/N_{ex} \leq \theta \leq \theta_i \tag{77}$$

$$c_{in} \simeq 0 \quad \text{when} \quad 0 \leq \psi \leq \psi_i \quad \text{or when} \quad \psi_a \leq \psi \leq 2\pi \tag{78}$$

Also, the slope of the bed γ to the drum axis is assumed to be constant for all angles where the bed is sloped. The total axial movement of the dense phase in an underloaded drum with the initial discharge of the exterior flights over the centrefill becomes:

$$F_2 = \frac{n\rho_b}{6\sin\phi}\left[\pi N_{ex,s}\left[\alpha+\gamma\cos\phi\right]\ell^3_{ex,\theta_i} + 2\alpha\left[N_{ex}J_{ex}+N_{in}J_{in}\right]\right] \tag{79}$$

where $N_{ex,s}$ is the number of non-discharging flights containing a bed of particles which are sloped with respect to the drum axis and J_{ex} and J_{in} represent integrals for the exterior and interior flights respectively. Equations for determining $N_{ex,s}$, J_{ex}, and J_{in} for the underloaded drum with an initial discharge angle for the exterior flights over the centrefill are:

$$N_{ex,s} = N_{ex}\left[\frac{\theta_i - \theta_d}{2\pi}\right] + 1 \quad \text{for} \quad \theta_a \leq \theta_i \leq \theta_b \tag{80}$$

$$J_{ex} = \int_{\theta_i}^{2\pi} \ell_{ex}^3 d\theta \tag{81}$$

$$J_{in} = \int_{\psi_i}^{\pi} \ell_{in}^3 d\psi \tag{82}$$

The same analysis for other loading conditions also yield Equation 79 for the total axial movement of the dense phase. Equations for $N_{ex,s}$, J_{ex}, and J_{in}, however, differ for the various loading conditions.

The number of sloped surfaces in an underloaded drum in which the exterior flights do not begin to discharge over the centrefill is given by Equation 83.

$$N_{ex,s} = N_{ex}\left[\frac{\theta_i - \theta_f}{2\pi}\right] + 1 \qquad \text{for} \qquad \pi \le \theta_i \le \theta_a \tag{83}$$

Only one sloped surface exists in overloaded drums.

$$N_{ex,s} = 1 \qquad \text{for} \qquad 0 \le \theta_i \le \pi \tag{84}$$

Integrals J_{ex} and J_{in} given by Equations 81 and 83 are also applicable to underloaded drums and overloaded drums that have no buried flights. For an overloaded drum with buried flights, flights between $\theta_i + 2\pi/N_{ex}$ and $2\phi + \pi - \theta_i$ are covered and

particles discharging from these flights do not move axially. The integral for the exterior flights in Equation 79 becomes:

$$J_{ex} = \int_{\theta_i}^{\theta_i+2\pi/N_{ex}} \ell_{ex}^3 d\theta + \int_{2\phi+\pi-\theta_i}^{2\pi} \ell_{ex}^3 d\theta \quad \text{for} \quad 0 \le \theta_i \le \phi+\pi/2 \tag{85}$$

For the special case when the slope of the bed γ to the drum axis is zero, then Equation 79 becomes

$$F_2 = \frac{n\rho_b\alpha}{6\sin\phi}\left[\pi N_{ex,s}\ell_{ex,\theta_i}^3 + 2\left[N_{ex}J_{ex}+N_{in}J_{in}\right]\right] \quad \text{for} \quad \gamma=0 \tag{86}$$

3.6 Drum Holdup

The total holdup of solids in a drum is the sum of the falling material, the material retained in the discharging flights, and the material in the non-discharging flights.

3.6.1 Falling Particle Holdup

The mass of particles in a state of fall originating from an exterior flight at angle θ is the product of the flight discharge rate, given by Equation 40, and the time of fall t_1. The mass in a state of fall originating from an exterior flight at angle θ is:

$$m_{1,ex} = \pi n \rho_b \ell_{ex}^2 t_{1,ex} \quad \text{when} \quad \theta_i \le \theta \le 2\pi \tag{87}$$

The average mass in a state of fall for an exterior flight between 0 and 2π radians is:

$$\overline{m_{1,ex}} = \frac{\int_{\theta_i}^{2\pi} m_{1,ex} d\theta}{\int_0^{2\pi} d\theta} \tag{88}$$

Substituting Equation 87 into Equation 88 and simplifying gives the following expression for the average mass in a state of fall originating from an exterior flight.

$$\overline{m_{1,ex}} = \frac{n\rho_b}{2} \int_{\theta_i}^{2\pi} \ell_{ex}^2 t_{1,ex} d\theta \tag{89}$$

A similar expression can be obtained for the average mass in a state of fall originating from an interior flight between 0 and 2π.

$$\overline{m_{1,in}} = \frac{n\rho_b}{2} \int_{\psi_i}^{2\pi} \ell_{in}^2 t_{1,in} d\psi \tag{90}$$

The total mass in a state of fall originating from both the exterior and interior flights is the sum of the product of the average falling mass for an exterior flight and the

number of exterior flights and the product of the average falling mass for an interior flight and the number of interior flights. The following equation expresses the total mass of particles in the drum in a state of fall.

$$M_1 = \frac{n\rho_b}{2}\left[N_{ex}\int_{\theta_i}^{2\pi} \ell_{ex}^2 t_{1,ex} d\theta + N_{in}\int_{\psi_i}^{\pi} \ell_{in}^2 t_{1,in} d\psi\right] \quad \textbf{(91)}$$

If vertical drag is assumed to be negligible, then the time of fall from any discharging flight is obtained using Equation 19.

The fall distance for particles discharged from an exterior flight in the upper half of the drum but <u>not</u> over a centrefill is the vertical distance from the tip of the discharging flight to the outer wall. For $\pi \le \theta \le \theta_a$ or $\theta_b \le \theta \le 2\pi$,

$$y_{ex} = \frac{D_o}{2}\left[\sin\left[\cos^{-1}\frac{D_{ex}}{D_o}\cos\theta\right] - \frac{D_{ex}}{D_o}\sin\theta\right] \quad \textbf{(92)}$$

The fall distance for particles discharged from an exterior flight over the centrefill wall is the vertical distance from the tip of the discharging flight to the wall of the centrefill. For $\theta_a < \theta < \pi + \cos^{-1}(D_i/D_{ex})$ or $\pi + \cos^{-1}(-D_i/D_{ex}) < \theta < \theta_b$,

$$y_{ex} = \frac{D_{ex}}{2}\left[\frac{D_i}{D_{ex}}\sin\left[\cos^{-1}\frac{D_{ex}}{D_i}\cos\theta\right] - \sin\theta\right] \quad \textbf{(93)}$$

The fall distance for particles discharged from an exterior flight over the flights

of the centrefill, but not over the wall of centrefill, is the vertical distance from the tip of the discharging flight to the horizontal centreline of the drum.

For $\pi+\cos^{-1}(D_i/D_{ex})<\theta<\pi+\cos^{-1}(-D_i/D_{ex})$,

$$y_{ex}=-\frac{D_{ex}}{2}\sin\theta \tag{94}$$

Particles discharged from exterior flights in the lower half of the drum are considered to have a fall distance of zero. For $0\le\theta\le\pi$,

$$y_{ex}=0 \tag{95}$$

The fall distance for particles discharged from an interior flight is the vertical distance from the tip of the discharging flight to the outer wall. For $0\le\psi\le\pi$,

$$y_{in}=\frac{D_o}{2}\left[\sin\left[\cos^{-1}\left[\frac{D_{in}\cos\psi}{D_o}\right]\right]-\frac{D_{in}}{D_o}\sin\psi\right] \tag{96}$$

3.6.2 Discharging Flight Holdup

The holdup of a discharging exterior flight at an angle of rotation θ is the material to be discharged between the current position θ and the position where the flight is empty at 2π. The discharge rate is given by Equation 40. Integration of the discharge rate yields the holdup of a discharging exterior flight at angle θ.

$$\text{for} \quad \theta_i \le \theta \le 2\pi, \quad h_{2,ex} = \int_{\theta}^{2\pi} \frac{\ell_{ex}^2}{2} d\theta \tag{97}$$

Similarly, the holdup of a discharging interior flight at angle ψ is found by integrating the discharge rate between the current position ψ and the position where the flight is empty at π.

$$\text{for} \quad \psi_i \le \psi \le \pi, \quad h_{2,in} = \int_{\psi}^{\pi} \frac{\ell_{in}^2}{2} d\psi \tag{98}$$

3.6.3 Non-discharging Flight Holdup

The holdup of a single exterior non-discharging flight is the material received between the angle $\theta = 0$ when the flight is empty and the current position θ. In most cases, the material received is found by considering the material discharged by flights vertically above the receiving flight. Depending on the position of the receiving flight, material received may be from an interior flight or another exterior flight. At some angles and under some conditions, a non-discharging flight may not be receiving any material.

In order to derive equations for the holdup of the non-discharging flights, each possible drum loading condition is considered separately. Figure 3.7 illustrates the definition of two more angles in a centrefilled drum. Angle θ_c is the exterior flight angle

which is directly below the interior flight at angle $\psi=0$. Angle θ_e is the exterior flight angle which is directly below the interior flight at angle $\psi=\pi$. Equations for determining these angles are:

$$\theta_c = \cos^{-1}\left[\frac{D_{in}}{D_{ex}}\right] \tag{99}$$

$$\theta_e = \cos^{-1}\left[-\frac{D_{in}}{D_{ex}}\right] \tag{100}$$

First consider an underloaded drum with an initial discharge angle for the exterior flights over the centrefill. An exterior flight on the lower periphery receives material from exterior flights between angles of $\theta = 0$ and $\theta = \theta_c$. No material is received between $\theta = \theta_c$ and $\theta = \theta_d$. Material is received from interior flights between $\theta = \theta_d$ and $\theta = \theta_e$, and finally, no material is received by an exterior flight between $\theta = \theta_e$ and $\theta = \theta_i$.

By integrating the amount discharged from the appropriate flights, the holdup of a non-discharging exterior flight in the underloaded drum with the initial discharge of the exterior flights over the centrefill is:

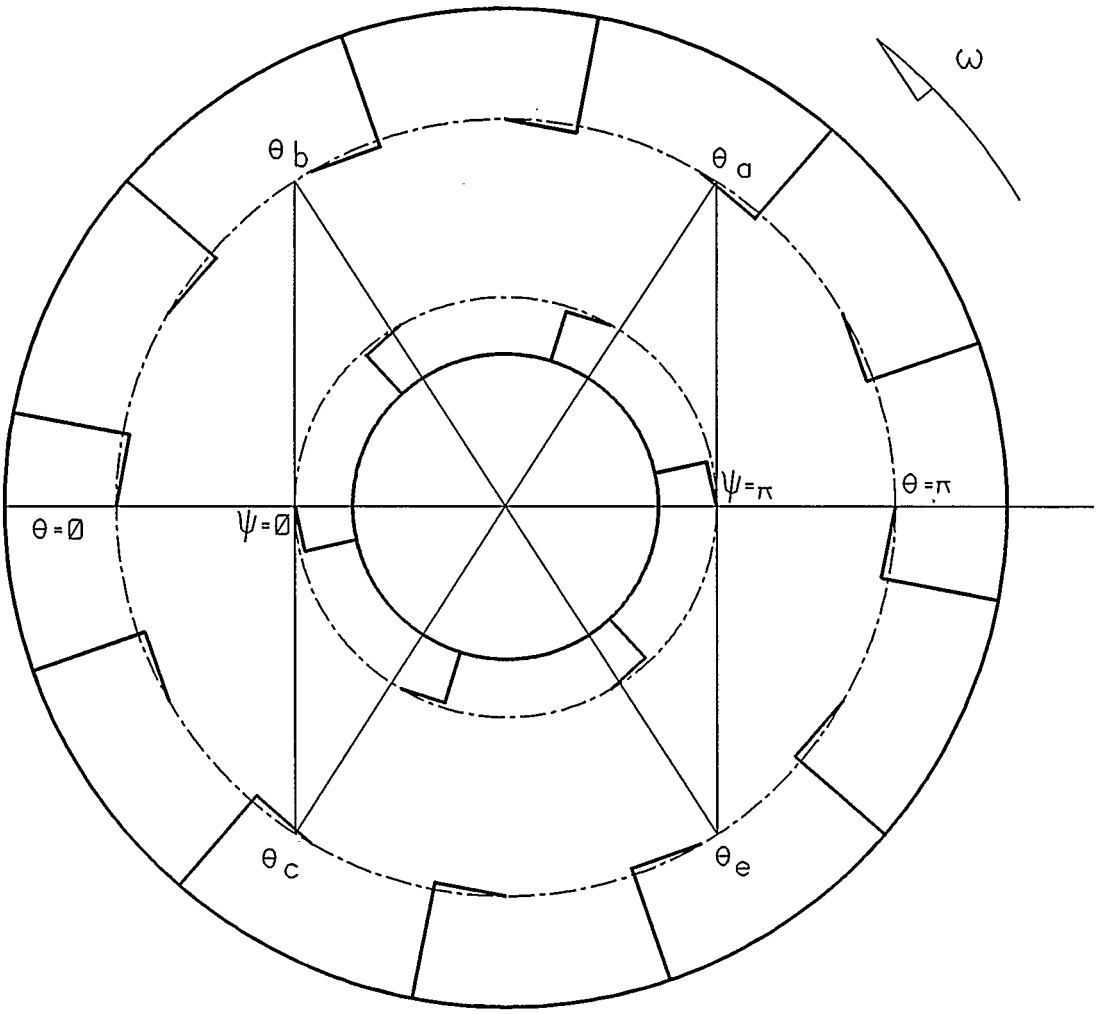


Figure 3.7 Definition of angles θ_c and θ_e

for $0 \le \theta \le \theta_c$,

$$h_{2,ex} = \int_{2\pi}^{2\pi-\theta} \frac{\ell_{ex}^2}{2} d\theta \tag{101}$$

for $\theta_c \le \theta \le \theta_d$,

$$h_{2,ex} = \int_{2\pi}^{2\pi-\theta_c} \frac{\ell_{ex}^2}{2} d\theta \tag{102}$$

for $\theta_d \le \theta \le \theta_e$,

$$h_{2,ex} = \int_{2\pi}^{2\pi-\theta_c} \frac{\ell_{ex}^2}{2} d\theta + \frac{N_{in}}{N_{ex}} \int_{\psi_i}^{\psi} \frac{\ell_{in}^2}{2} d\psi \tag{103}$$

for $\theta_e \le \theta \le \theta_i$,

$$h_{2,ex} = \int_{2\pi}^{2\pi-\theta_c} \frac{\ell_{ex}^2}{2} d\theta + \frac{N_{in}}{N_{ex}} \int_{\psi_i}^{\pi} \frac{\ell_{in}^2}{2} d\psi \tag{104}$$

where $$\psi = \cos^{-1}\left[\frac{D_{ex}}{D_{in}}\cos\theta\right] \tag{105}$$

and $$d\psi = \frac{D_{ex}}{D_{in}} \frac{\sin\theta}{\sin\psi} d\theta \tag{106}$$

Equations 101 through to 106 also apply to an underloaded drum in which the initial discharge angle of the exterior flight is not over the centrefill, with the exception of Equation 104, which is replaced by the following two equations.

$$\text{for} \quad \theta_e \le \theta \le \theta_f, \quad h_{2,ex} = \int_{2\pi}^{2\pi-\theta_c} \frac{\ell_{ex}^2}{2} d\theta + \frac{N_{in}}{N_{ex}} \int_{\psi_i}^{\pi} \frac{\ell_{in}^2}{2} d\psi + \int_{2\pi-\theta_e}^{2\pi-\theta} \frac{\ell_{ex}^2}{2} d\theta \qquad \textbf{(107)}$$

$$\text{for} \quad \theta_f \le \theta \le \theta_i, \quad h_{2,ex} = \int_{2\pi-0}^{2\pi-\theta_c} \frac{\ell_{ex}^2}{2} d\theta + \frac{N_{in}}{N_{ex}} \int_{\psi_i}^{\pi} \frac{\ell_{in}^2}{2} d\psi + \int_{2\pi-\theta_e}^{2\pi-\theta_f} \frac{\ell_{ex}^2}{2} d\theta \qquad \textbf{(108)}$$

The holdup of the exterior flights for an overloaded drum depends on the initial discharge angle of the exterior flight with respect to the centrefill. If the initial discharge angle θ_i is between θ_e and π, then the exterior flight holdup is given by Equation 101 between 0 and θ_c, by Equation 102 between θ_c and θ_d, by Equation 103 between θ_d and θ_e, by Equation 107 between θ_e and θ_i. If θ_i is between θ_d and θ_e, then the exterior flight holdup is given by Equation 101 between 0 and θ_c, by Equation 102 between θ_c and θ_d and by Equation 103 between θ_d and θ_i. If θ_i is between θ_c and θ_d, then the exterior flight holdup is given by Equation 101 between 0 and θ_c and by Equation 102 between θ_c and θ_i. If θ_i is less than θ_c, then the exterior flight holdup is given by Equation 101 between 0 and θ_i.

Consider the holdup of a non-discharging interior flight in an underloaded drum in which the exterior flights begin to discharge over the centrefill at angle ψ where

$\pi \le \psi \le 2\pi$. The current holdup of the flight is equal to the amount of material discharged from the exterior flight between angles θ_i and θ. The holdup of an interior flight between angles 0 and ψ_i is the total amount received from the exterior flights discharging between θ_i and θ_b. The holdup of a receiving flight at angle ψ is:

for $\pi \le \psi \le \psi_a$,

$$h_{2,in} = 0 \tag{109}$$

for $\psi_a \le \psi \le 2\pi$,

$$h_{2,in} = \frac{N_{ex}}{N_{in}} \int_{\theta_i}^{\theta} \frac{\ell_{ex}^2}{2} d\theta \tag{110}$$

for $0 \le \psi \le \psi_i$,

$$h_{2,in} = \frac{N_{ex}}{N_{in}} \int_{\theta_i}^{\theta_b} \frac{\ell_{ex}^2}{2} d\theta \tag{111}$$

where $$\theta = \pi + \cos^{-1}\left[\frac{D_{in}}{D_{ex}} \cos(\psi - \pi)\right] \tag{112}$$

and $$d\theta = \frac{D_{in}}{D_{ex}} \frac{\sin(\psi - \pi)}{\sin(\theta - \pi)} d\psi \tag{113}$$

Similar treatment provides equations for holdup of non-discharging interior flights

for other loading conditions. Equations describing the holdup of the interior flights for all loading conditions except for an underloaded drum with the initial discharge angle of the exterior flights over the centrefill are:

$$\text{for} \quad \pi \le \psi \le 2\pi, \quad h_{2,in} = \frac{N_{ex}}{N_{in}} \int_{\theta_a}^{\theta} \frac{\ell_{ex}^2}{2} d\theta \tag{114}$$

$$\text{for} \quad 0 \le \psi \le \psi_i, \quad h_{2,in} = \frac{N_{ex}}{N_{in}} \int_{\theta_i}^{\theta_a} \frac{\ell_{ex}^2}{2} d\theta \tag{115}$$

3.6.4 Total Drum Holdup

The average holdup of an exterior flight between the angles of 0 and 2π is

$$\overline{h_{2,ex}} = \frac{\int_0^{2\pi} h_{2,ex} d\theta}{\int_0^{2\pi} d\theta} \tag{116}$$

From Equations 97, 101 through 104, and 116, the average holdup of an exterior flight in an underloaded drum in which the initial discharge angle is over the centrefill becomes

$$\overline{h_{2,ex}} = \frac{1}{4\pi}\left[\int_0^{\theta_c}\int_{2\pi}^{2\pi-\theta} \ell_{ex}^2 d\theta d\theta + \int_{\theta_d}^{\theta_e} \frac{N_{in}}{N_{ex}} \int_{\psi_i}^{\psi} \ell_{in}^2 d\psi d\theta + \int_{\theta_i}^{2\pi}\int_{\theta}^{2\pi} \ell_{ex}^2 d\theta d\theta + (\theta_i-\theta_c)\int_{2\pi}^{2\pi-\theta_c} \ell_{ex}^2 d\theta + (\theta_i-\theta_e)\frac{N_{in}}{N_{ex}}\int_{\psi_i}^{\pi} \ell_{in}^2 d\psi\right] \quad (117)$$

For an underloaded drum in which the initial discharge of the exterior flights is not over the centrefill, Equation 97, 101 through 103, 107, 108, and 116 are combined to give the following equation for the average exterior flight holdup.

$$\overline{h_{2,ex}} = \frac{1}{4\pi}\left[\int_0^{\theta_c}\int_{2\pi}^{2\pi-\theta} \ell_{ex}^2 d\theta d\theta + \int_{\theta_d}^{\theta_e} \frac{N_{in}}{N_{ex}} \int_{\psi_i}^{\psi} \ell_{in}^2 d\psi d\theta + \int_{\theta_i}^{2\pi}\int_{\theta}^{2\pi} \ell_{ex}^2 d\theta d\theta + \int_{\theta_e}^{\theta_f}\int_{2\pi-\theta_e}^{2\pi-\theta} \ell_{ex}^2 d\theta d\theta + (\theta_i-\theta_c)\int_{2\pi}^{2\pi-\theta_c} \ell_{ex}^2 d\theta + (\theta_i-\theta_e)\frac{N_{in}}{N_{ex}}\int_{\psi_i}^{\pi} \ell_{in}^2 d\psi + (\theta_i-\theta_f)\int_{2\pi-\theta_e}^{2\pi-\theta_f} \ell_{ex}^2 d\theta\right] \quad (118)$$

The same method is used to derive equations for the average exterior flight holdup for overloaded drums. The following equations were obtained.

For $\theta_e \leq \theta_i \leq \pi$,

$$\overline{h_{2,ex}} = \frac{1}{4\pi}\left[\begin{array}{l} \int_0^{\theta_c}\int_{2\pi}^{2\pi-\theta} \ell_{ex}^2 d\theta d\theta + \int_{\theta_d}^{\theta_e} \frac{N_{in}}{N_{ex}} \int_{\psi_i}^{\psi} \ell_{in}^2 d\psi d\theta + \int_{\theta_i}^{2\pi}\int_{\theta}^{2\pi} \ell_{ex}^2 d\theta d\theta \\ + \int_{\theta_e}^{\theta_i}\int_{2\pi-\theta_e}^{2\pi-\theta} \ell_{ex}^2 d\theta d\theta + (\theta_i-\theta_c)\int_{2\pi}^{2\pi-\theta_c} \ell_{ex}^2 d\theta + (\theta_i-\theta_e)\frac{N_{in}}{N_{ex}}\int_{\psi_i}^{\pi} \ell_{in}^2 d\psi \end{array}\right] \quad \textbf{(119)}$$

For $\theta_d \le \theta_i \le \theta_e$,

$$\overline{h_{2,ex}} = \frac{1}{4\pi}\left[\begin{array}{l} \int_0^{\theta_c}\int_{2\pi}^{2\pi-\theta} \ell_{ex}^2 d\theta d\theta + \int_{\theta_d}^{\theta_i} \frac{N_{in}}{N_{ex}} \int_{\psi_i}^{\psi} \ell_{in}^2 d\psi d\theta + \int_{\theta_i}^{2\pi}\int_{\theta}^{2\pi} \ell_{ex}^2 d\theta d\theta \\ + (\theta_i-\theta_c)\int_{2\pi}^{2\pi-\theta_c} \ell_{ex}^2 d\theta \end{array}\right] \quad \textbf{(120)}$$

For $\theta_c \le \theta_i \le \theta_d$,

$$\overline{h_{2,ex}} = \frac{1}{4\pi}\left[\int_0^{\theta_c}\int_{2\pi}^{2\pi-\theta} \ell_{ex}^2 d\theta d\theta + \int_{\theta_i}^{2\pi}\int_{\theta}^{2\pi} \ell_{ex}^2 d\theta d\theta + (\theta_i-\theta_c)\int_{2\pi}^{2\pi-\theta_c} \ell_{ex}^2 d\theta\right] \quad \textbf{(121)}$$

For $0 \le \theta_i \le \theta_c$,

$$\overline{h_{2,ex}} = \frac{1}{4\pi}\left[\int_0^{\theta_i}\int_{2\pi}^{2\pi-\theta} \ell_{ex}^2 d\theta d\theta + \int_{\theta_i}^{2\pi}\int_{\theta}^{2\pi} \ell_{ex}^2 d\theta d\theta\right] \quad \textbf{(122)}$$

The average holdup of an interior flight between the angles of 0 and 2π is

$$\overline{h_{2,in}} = \frac{\int_0^{2\pi} h_{2,in} d\psi}{\int_0^{2\pi} d\psi} \tag{123}$$

From Equation 98, Equation 109 through 111 and Equation 123, the average holdup of an interior flight in an underloaded drum in which the exterior flights begin discharging over the centrefill is:

$$\overline{h_{2,in}} = \frac{1}{4\pi}\left[\psi_i \frac{N_{ex}}{N_{in}} \int_{\theta_i}^{\theta_b} \ell_{ex}^2 d\theta + \int_{\psi_i}^{\pi}\int_{\psi}^{\pi} \ell_{in}^2 d\psi d\psi + \int_{\psi_a}^{2\pi} \frac{N_{ex}}{N_{in}} \int_{\theta_i}^{\theta} \ell_{ex}^2 d\theta d\psi\right] \tag{124}$$

Combining Equations 98, 114, 115 and 123 gives the average holdup of an interior flight for any drum loading condition except an underloaded drum in which the exterior flights begin to discharge over the centrefill.

$$\overline{h_{2,in}} = \frac{1}{4\pi}\left[\psi_i \frac{N_{ex}}{N_{in}} \int_{\theta_a}^{\theta_b} \ell_{ex}^2 d\theta + \int_{\psi_i}^{\pi}\int_{\psi}^{\pi} \ell_{in}^2 d\psi d\psi + \int_{\psi_a}^{2\pi} \frac{N_{ex}}{N_{in}} \int_{\theta_a}^{\theta} \ell_{ex}^2 d\theta d\psi\right] \tag{125}$$

The total holdup of the flights is the sum of the product of the number of exterior flights and the average holdup of an exterior flight and the product of the number of interior flights and the average holdup of an interior flight.

$$H_2 = N_{ex}\overline{h_{2,ex}} + N_{in}\overline{h_{2,in}} \tag{126}$$

The total drum holdup is the sum of the holdup in the flights and the holdup in a state of fall.

$$H = H_1 + H_2 \quad \text{where} \quad H_1 = \frac{M_1}{\rho_b} \tag{127}$$

3.7 Change in Holdup with Drum Length

The change in drum holdup due to change in the initial discharge angle of the exterior flights can be determined for each loading condition. For an underloaded drum in which the exterior flights begin to discharge over the centrefill, Equations 117, 124, and 91 are substituted into 126 and 127 to obtain the total drum holdup. The result is differentiated with respect to the initial discharge angle of the exterior flights.

For $\theta_a \leq \theta_i \leq \theta_b$,

$$\frac{dH}{d\theta_i} = \frac{N_{ex}}{4\pi}\left[-\int_{\theta_i}^{2\pi} \ell_{ex}^2 d\theta - \int_{2\pi}^{2\pi-\theta_c} \ell_{ex}^2 d\theta + \frac{N_{in}}{N_{ex}} \int_{\psi_i}^{\pi} \ell_{in}^2 d\psi - \int_{\theta_a}^{\theta_i} \ell_{ex}^2 d\theta - 2(\theta_i - \pi)\ell_{ex,\theta_i}^2 \right] \tag{128}$$

For an underloaded drum in which the exterior flights do not begin to discharge over the centrefill, $\pi \leq \theta_i \leq \theta_a$,

$$\frac{dH}{d\theta_i} = \frac{N_{ex}}{4\pi}\left[\int_{2\pi}^{2\pi-\theta_c} \ell_{ex}^2 d\theta + \frac{N_{in}}{N_{ex}}\int_{\psi_i}^{\pi} \ell_{in}^2 d\psi + \int_{2\pi-\theta_e}^{2\pi-\theta_f} \ell_{ex}^2 d\theta - \int_{\theta_i}^{2\pi} \ell_{ex}^2 d\theta\right] - \frac{N_{ex} n}{2} \ell_{ex,\theta_i}^2 t_{1,ex,\theta_i} \quad \textbf{(129)}$$

The change in drum holdup due to change in the discharge angle for an overloaded drum depends on the position of the initial discharge with respect to the centrefill.

For $\theta_e \leq \theta_i \leq \pi$,

$$\frac{dH}{d\theta_i} = \frac{N_{ex}}{4\pi}\left[\int_{2\pi}^{2\pi-\theta_c} \ell_{ex}^2 d\theta + \frac{N_{in}}{N_{ex}}\int_{\psi_i}^{\pi} \ell_{in}^2 d\psi + \int_{2\pi-\theta_e}^{2\pi-\theta_i} \ell_{ex}^2 d\theta - \int_{\theta_i}^{2\pi} \ell_{ex}^2 d\theta\right] \quad \textbf{(130)}$$

For $\theta_d \leq \theta_i \leq \theta_e$,

$$\frac{dH}{d\theta_i} = \frac{N_{ex}}{4\pi}\left[\int_{2\pi}^{2\pi-\theta_c} \ell_{ex}^2 d\theta + \frac{N_{in}}{N_{ex}}\int_{\psi_i}^{\psi_b} \ell_{in}^2 d\psi - \int_{\theta_i}^{2\pi} \ell_{ex}^2 d\theta\right] \quad \textbf{(131)}$$

For $\theta_c \leq \theta_i \leq \theta_d$,

$$\frac{dH}{d\theta_i} = \frac{N_{ex}}{4\pi}\left[\int_{2\pi}^{2\pi-\theta_c} \ell_{ex}^2 d\theta - \int_{\theta_i}^{2\pi} \ell_{ex}^2 d\theta\right] \quad \textbf{(132)}$$

For $0 \leq \theta_i \leq \theta_c$,

$$\frac{dH}{d\theta_i} = \frac{N_{ex}}{4\pi}\left[\int_{2\pi}^{2\pi-\theta_i} \ell_{ex}^2 d\theta - \int_{\theta_i}^{2\pi} \ell_{ex}^2 d\theta\right] \quad \textbf{(133)}$$

The change in drum holdup due to bed slope along the drum length is approximated by

$$\frac{dH}{dz} \simeq \gamma \ell_{ex,\theta_i} N_{ex,s} \quad \textbf{(134)}$$

where $N_{ex,s}$ is the number of flights with sloped surfaces.

The slope of the bed γ is found by rearranging Equation 79.

$$\gamma = \frac{1}{\cos\phi}\left[\frac{2}{\pi \ell_{ex,\theta_i}^3 N_{ex}}\left[\frac{3F_2 \sin\phi}{n\rho_b} - \alpha\left[N_{ex}J_{ex} + N_{in}J_{in}\right]\right] - \alpha\right] \quad \textbf{(135)}$$

The change in the initial angle of discharge of the exterior flights with respect to the drum length is:

$$\frac{d\theta_i}{dz} = \frac{\frac{dH}{dz}}{\frac{dH}{d\theta_i}} \quad \textbf{(136)}$$

3.8 Residence Time

The mean total holdup for a drum of length L is

$$\bar{H} = \frac{\int_0^L H dz}{\int_0^L dz} \tag{137}$$

The total residence time is the mean total holdup divided by the feed rate.

$$T = \frac{\bar{H} L \rho_b}{F_0} \tag{138}$$

The residence time of particle size i is the holdup of particle size i divided by the feed rate of particle size i.

$$T_i = \frac{\rho_b}{F_0 x_{f,i}} \int_0^L H x_{d,i} dz \tag{139}$$

For the special case when the holdup and the particle size distribution are constant for the length of the drum, then Equation 139 simplifies to the following equation for the residence time of each particle size.

$$T_i = T \frac{x_{d,i}}{x_{f,i}} \quad \text{when} \quad H \text{ and } x_{d,i} \text{ are constant.} \tag{140}$$

CHAPTER 4

EXPERIMENTAL

Most previous workers, who have developed residence time models for rotary dryers and coolers, have assumed that the particles fall independently of each other and that every falling particle encounters the average horizontal gas velocity. Their comparison to experimental results, however, have indicated that these models generally over-estimate the effect of the gas stream on the advance of the falling particles. Previous authors attributed this phenomena to the shielding effect of the particles falling from the flights in sheets. Particles in the sheets do not encounter the full horizontal gas velocity and therefore the distance they are displaced horizontally by the gas stream is reduced.

To study the horizontal displacement of a sheet of particles falling through a horizontal gas stream, some experiments were performed. A long, single sheet of particles was poured into a large wind tunnel with the gas flow parallel to the horizontal length of the sheet.

Work by Langrish (1989) indicates that the parameters which influence the extent of the shielding effect are the sheet thickness, the sheet density and the axial length. The sheet thickness is related to the discharge rate of the flight. High discharge rates produce thick sheets, while low discharge rates produce thin sheets. The discharge rate is a function of the flight shape and the rotational speed.

The density of the sheet is related to the distance the sheet has fallen. The average density of a sheet decreases with increasing fall distance. As the particles fall,

the sheet begins to disperse reducing the density of the sheet. Also, the particles are accelerating vertically which decreases the density further with increased fall height.

The axial length of the sheet affects the number of particles that are shielded by the leading particles. The longer the sheet is, the more particles that are shielded. The length of the sheet is governed by the axial length of the flight.

4.1 Apparatus

The apparatus used to investigate the horizontal displacement of a sheet of particles poured into a horizontally flowing gas stream was a large wind tunnel illustrated in Figure 4.1. The wind tunnel was constructed at UMATAC Industrial Processes pilot plant site in Calgary. The wind tunnel was 10.3 metres high, 1.2 metres wide and 11.2 metres long. At the top of the wind tunnel, near the upstream end, a tilting hopper was used to pour a stream of test material into the wind tunnel. The shape of the hopper and the orientation of the stream of particles to the gas stream is shown in Figure 4.2. The intent was that the discharge from the hopper would be similar to the discharge from a flight in a rotating drum. The hopper profile was analogous to an EAD flight which provides a constant rate of discharge for a constant rate of rotation.

The tilting hopper had a radius of 0.51 metres. By using a vertical divider, the hopper could be 0.98 or 1.96 metres long. An alternate hopper had a 0.35 metre radius and was 1.96 metres long. A hydraulic piston was used to slowly rotate the hopper for discharging. The hopper and the hydraulic piston were mounted on a separate frame. The frame was suspended from the ceiling of the tunnel using three load cells.

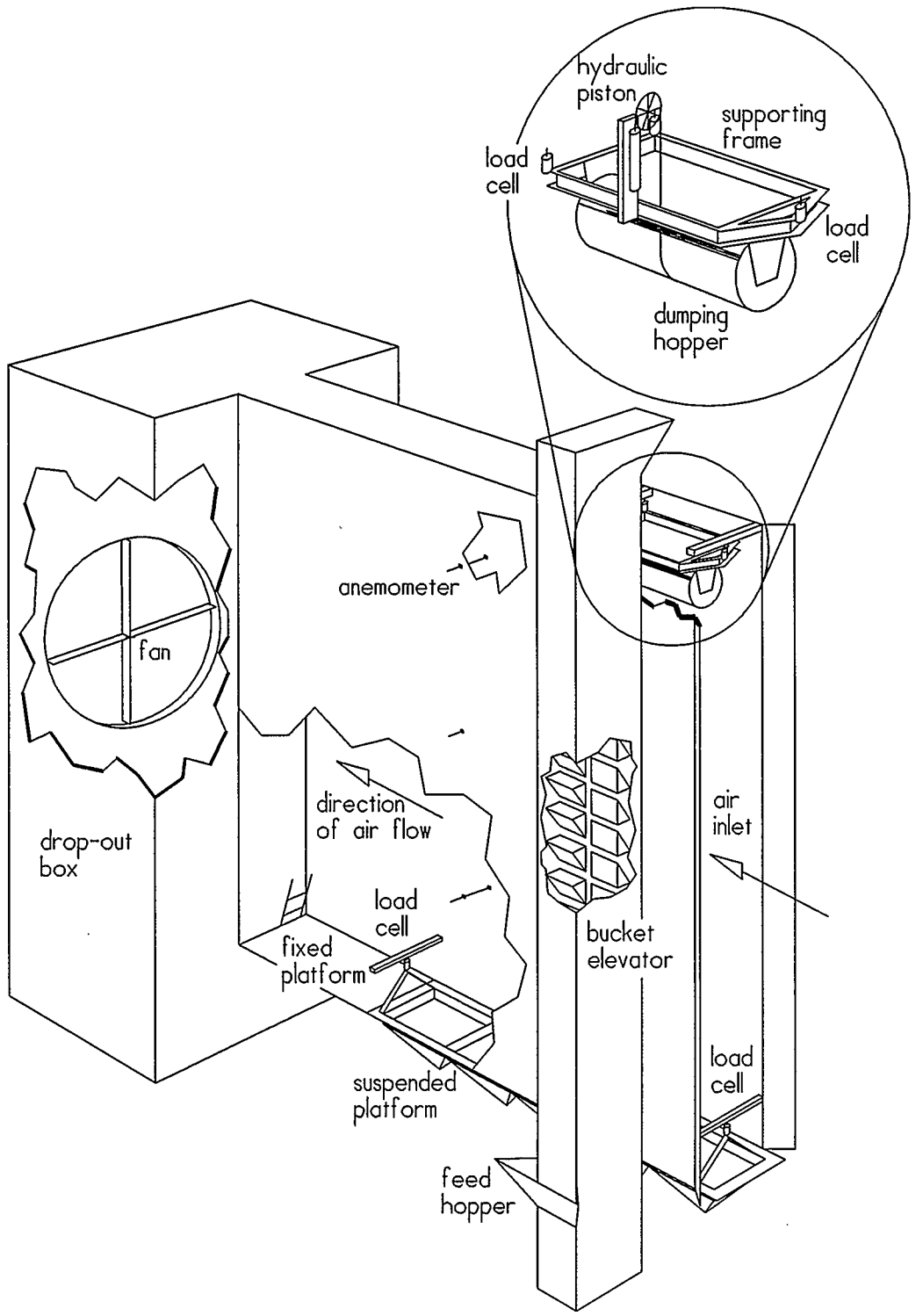
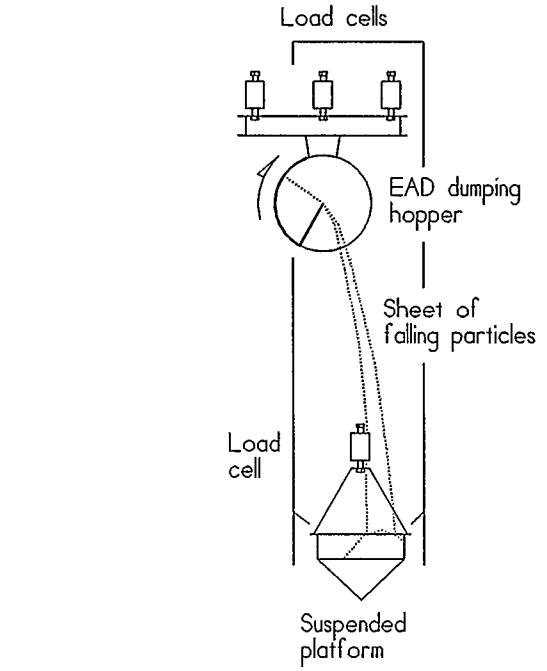
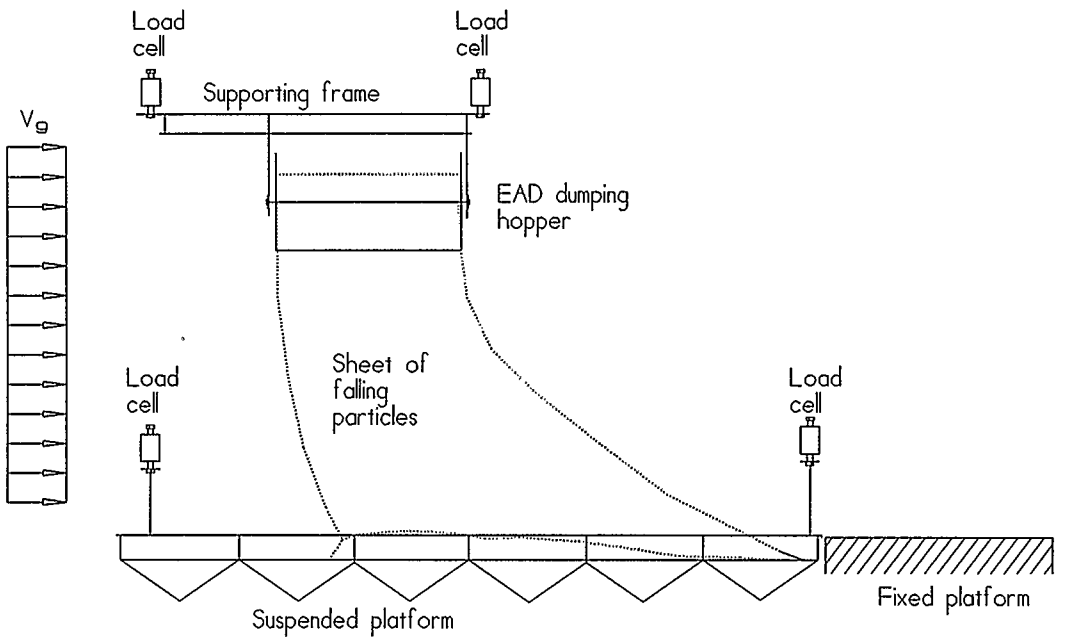


Figure 4.1 Wind tunnel apparatus for investigating the horizontal displacement of a curtain of falling particles



a) FRONT VIEW



b) SIDE VIEW

Figure 4.2 Front and side views of a falling curtain of particles in the wind tunnel

At the bottom of the wind tunnel were two platforms; a platform suspended from two load cells formed a floor for the first 7.5 metres of the tunnel length and a fixed platform formed a floor for the remaining 3.8 metres of the tunnel length. The suspended platform was sectioned into six equal compartments.

At the downstream end of the wind tunnel was a diffuser box and a 3 metre variable pitch fan. The diffuser box assured a uniform gas velocity in the tunnel. Midway along the wind tunnel, three hot-wire anemometers were mounted at various heights. A pitot tube meter was used to calibrate the anemometers and to check the velocity distribution in the wind tunnel.

4.2 Procedure

The procedure for each test was as follows: A feed sample was taken from the loader bucket before dumping the feed material to the bucket elevator. The hopper in its upright position was filled. The material in the hopper was levelled and all spillage was cleared from the platforms at the bottom of the wind tunnel. The weights on each of the three load cells which supported the tilting hopper and the two load cells which supported the empty collection platform were recorded. The wind tunnel fan was started and the fan pitch was adjusted to achieve the desired air velocity. The hopper was discharged by rotating it at a constant angular speed. The time from the start of discharge to when the hopper was empty was recorded. The bulk of the material fell onto the suspended platform and a small amount settled on the fixed platform. Some material was carried past the platforms and either settled in the diffuser box or was swept

through the fan. After the discharge was completed, the fan was stopped and the hopper was returned to its upright position. The load cell weights for the empty hopper and the suspended platform were recorded. The suspended platform was then cleared section by section of the caught material. Each of the six sections was sampled before being emptied. The platform was reweighed after each section was emptied. When the platform was totally cleared, the material on the fixed platform was swept up, sampled and weighed on the suspended platform.

Sieve analyses were performed to determine particle size distributions of the feed and platform samples. A moisture analysis of the feed sample was performed by drying the material in a 105°C oven for three hours. Some the later tests did not include the individual sampling and cleaning of each section of the suspended platform.

One hundred and twenty six tests were completed. Three types of feed material were used. The particle size distribution and other properties of the feed materials are given in Appendix A. The gas velocity, flight discharge rate, and fall distance were varied to cover the ranges that are typical in large industrial dryers. The air velocity was varied from 1.4 m/s to 8.6 m/s. The hopper discharge rate was varied from 7 to 41 kg/s m of hopper length. The vertical fall distance was varied by raising the suspended and fixed platforms. The three fall distances tested were 3.5, 6.6 and 10.3 metres. Sixteen tests were discarded due to instrumentation problems along with ten trial tests.

4.3 Analysis and Results

In the analysis of the results, the mean horizontal displacement of the particles for

each test was determined. The mean horizontal displacement of each particle size grouping in the test was also determined.

4.3.1 Mean Displacement of the Bulk Material

From the difference in the load cell weights of the full and empty hopper, the centre of gravity of the material in the hopper was calculated. From the difference of the load cell weights for the suspended platform, the centre of gravity of the caught material was determined. The material caught on the fixed platform was assumed to have a centre of gravity at the centre of the platform. The lost material, which was determined by a mass balance on the feed, was assumed to have travelled to the far end of the fixed platform. The material caught on the two platforms and the lost material were used to find the overall centre of gravity of the fallen material. The mean horizontal displacement of the bulk material was the difference between the centre of gravities of the material in the hopper and the fallen material. A tabular summary of the tests performed and the resulting mean displacement of the material is given Appendix A.

4.3.2 Particle Size and Displacement

In addition to the mean displacement of the bulk feed material, the mean displacement of each particle size grouping was also ascertained. For each test, the centre of gravity of the material in each section of the suspended platform was determined. Combining this information with the particle size distribution for each

section, the mean displacement of each particle size grouping was determined. The displacement for each size grouping required correction for lost material. Two methods of approximating the particle size distribution of the lost material were examined. The first method found the distribution through a balance for each particle size grouping with the feed distribution. The second method assumed that the lost material had the same distribution of particle sizes as the sample taken from the fixed platform, which was the finest material recovered.

Figure 4.3 shows the average displacement for the individual particle sizes determined by the two methods for a typical test. The plot shows how the two methods of determining the particle size distribution of the lost material affect the results. Mainly, the displacement of the smaller particles are understated for the second method. Generally, however, the second method gave more reasonable results, so this method was used in the remainder of the analysis.

4.3.3 Comparison to Single Particle Behaviour

In Figure 4.4, the mean horizontal displacement for individual particles sizes for a test are compared to the behaviour of a single independent particle predicted by Equation 17 using the measured air velocity. A similar plot was generated for each test but are not presented in this report due to their large number. However, the plot shown in Figure 4.4 is typical of most of the tests.

Some qualitative observations can be made from these plots. For low spill rates and the large fall heights, good agreement was found between the experimental results

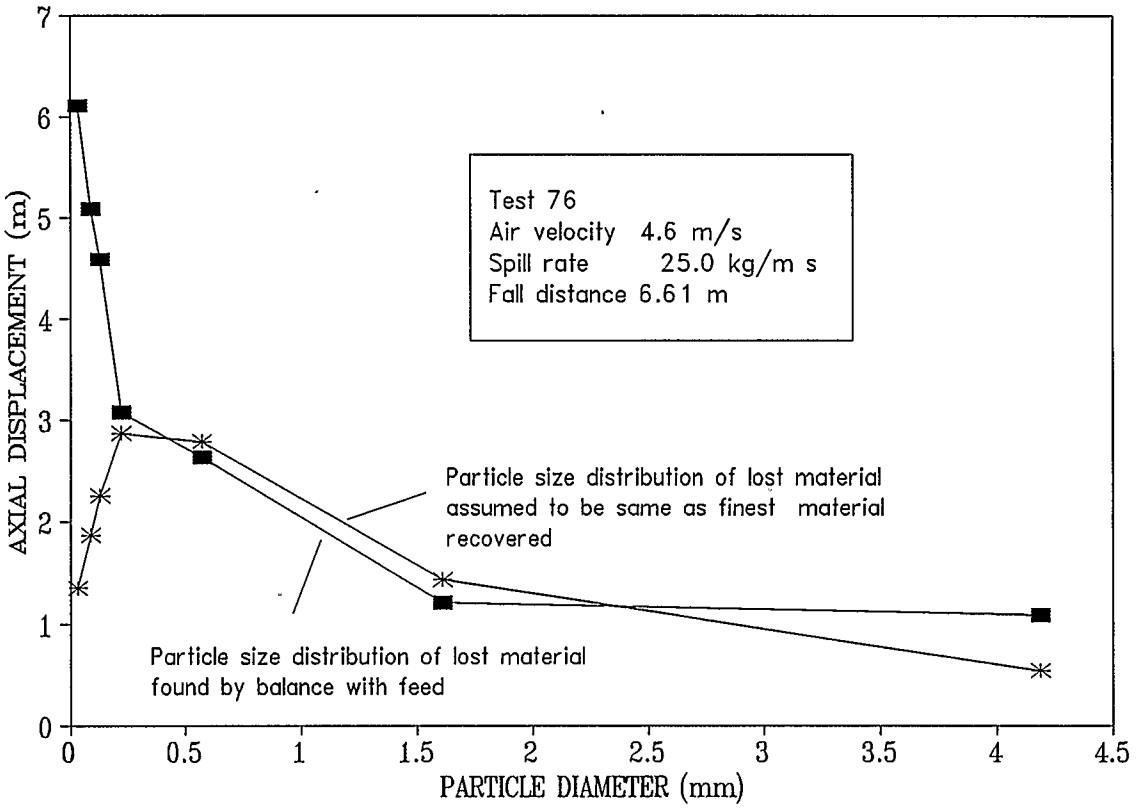


Figure 4.3 Mean horizontal displacement for a falling sheet of particles - Two methods of determining particles size distribution of lost material

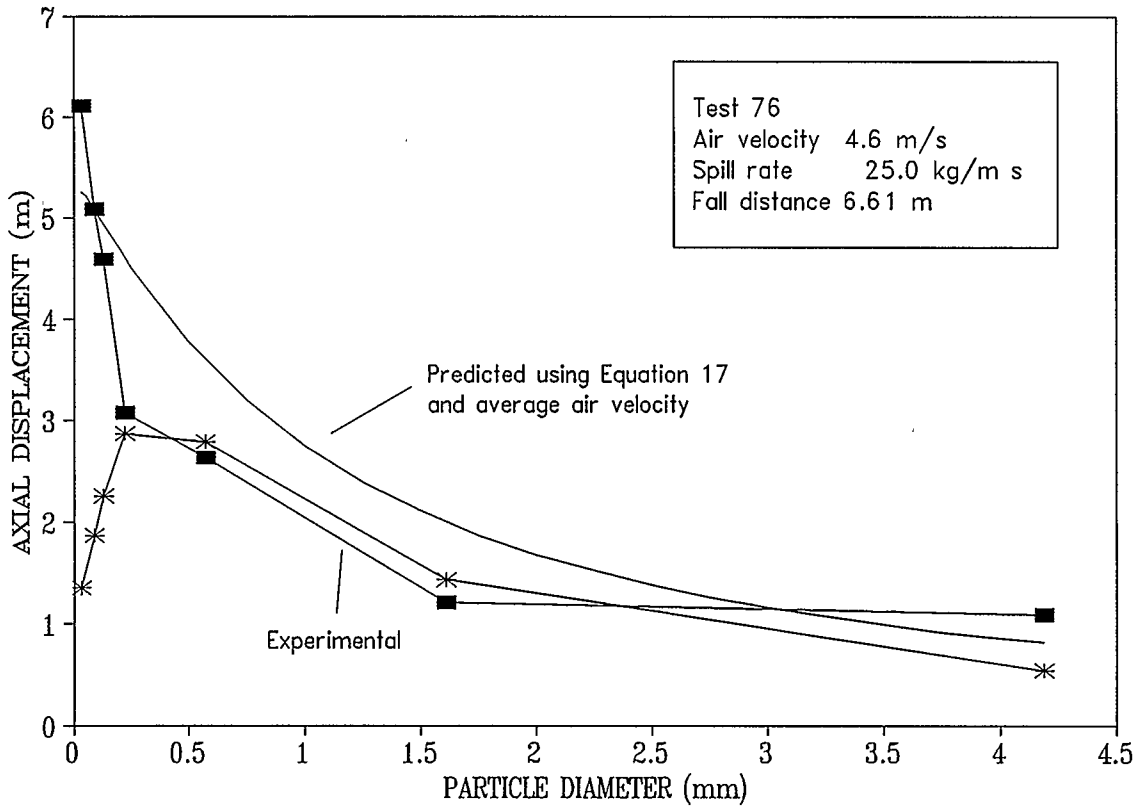


Figure 4.4 Comparison of horizontal displacement of particles in a falling sheet to predicted behaviour of a single independent particle

and the predicted behaviour for single particles. Increasing the spill rate caused the particle displacement to be lower than predicted. Decreasing the fall height also caused the actual particle displacement to be lower than predicted. It is difficult to tell from observing the plots whether the flight length had any effect.

Kamke (1984) had suggested that the falling particles interact with each other to the extent that all particles advance the same distance regardless of the particle size. According to Kamke's deductions, the larger particles would have a tendency to advance more than predicted by single particle behaviour, while smaller than average particles would advance less than predicted. This was not observed in these experiments. The displacement was lower for all particle sizes. The particles did not behave as a group characterized by a mean diameter as Kamke had suggested.

4.3.4 Apparent Gas Velocity Within Sheets

For most of the tests, it appears as if the single particle equation using a lower gas velocity would be able to fit the experimental results. The observation supports the view of O'Donnell (1975) and Langrish (1989) that two gas velocities exist in the drum: one higher than average velocity outside the sheets and one lower than average velocity within the sheets.

For each test, the apparent gas velocity, V_c, within the sheet was found by trial and error as follows. For each particle grouping, the differences between the predicted displacement, $\delta_i(V_c)$, using Equation 17 with a guess for the apparent velocity and the actual displacements, $\delta_{i,actual}$, was found. The squares of the differences were weighted

by the corresponding mass fraction, x_i, in the feed and then summed. The two smallest particles sizes were not used due to the errors caused by the way the size distribution of the lost material was determined. Successive guesses for the gas velocity were performed to minimize the sum.

$$\sum_i x_i\left(\delta_{i,actual}-\delta_i(V_c)\right)^2 = \text{minimum} \qquad \textbf{(141)}$$

The velocity which minimized the sum of the squares of the difference between the actual and predicted displacements is referred to as the apparent velocity within the sheet. The apparent velocity was determined for each test and results are in Appendix A. A correction factor, C_V, which is the ratio between the apparent velocity and the average velocity, V_g, was also determined for each test and is given in Appendix A.

A least squares multiple regression fit was used to derive the following empirical equation for the velocity correction factor dependence on the fall height (m), the spill rate (kg/s m) and the flight length (m).

$$C_V = \frac{V_c}{V_g} = 1.546\ \frac{y^{0.76}}{(dm/dt)^{0.86}L^{0.44}} \qquad \textbf{(142)}$$

The standard error is 0.296 and the regression coefficient squared is 0.631. The velocity correction factor calculated for each test using the above equation is compared to the actual ratio of the apparent velocity to the average velocity in Figure 4.5. The plot illustrates the relatively poor fit of Equation 142 and the experimental results. The

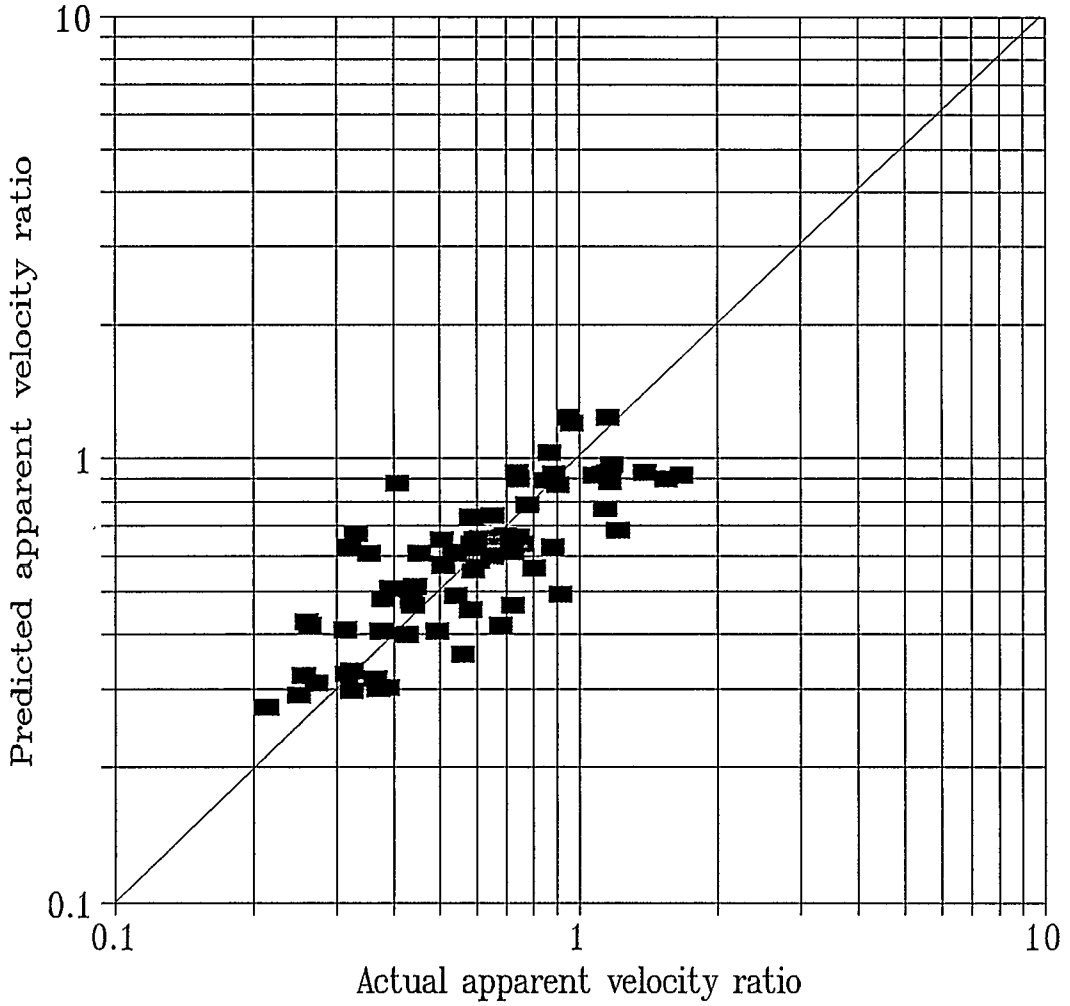


Figure 4.5 Comparison of predicted velocity correction factor C_V and the actual apparent velocity ratio

significant error is attributed to the enormity of the apparatus and the inherent difficultly in controlling gas velocity and spill rate.

For approximating the horizontal displacement of falling particles in a drum due to the gas stream, the correction factor is applied to the average velocity to obtain the velocity encountered by the particles within the sheets. The displacement of the particles are determined using the equation for single particles using the lower velocity.

CHAPTER 5

MODEL APPLICATIONS

The equations reported in the proceeding chapters can be combined to form a solids residence time model for flighted rotating drums. Because the equations can be solved differently depending on whether constant holdup along the drum length can be assumed, three separate computer programs were written. The first program DRUM1 calculates the holdup and flow rate of the particles assuming constant holdup along the drum length. The second program DRUM2 calculates holdup and the change in holdup with respect to the drum length given the total flow rate. The third program DRUM3 calculates the total holdup for a drum in which the holdup is allowed to vary along the length of the drum given the total flow rate.

5.1 Program DRUM1 - Constant Holdup along Drum Length

For a long drum in which the solids flow in the airborne phase is large due to either the drum incline or to the cocurrent gas flow, the holdup can be assumed to be constant. At some axial distance from the solids discharge, the axial slope of the bed surface will approach the slope of the drum. From this point to the feed end of the drum, the holdup is constant. If the length of the constant holdup section is sufficiently long, then the change in holdup near the discharge can be ignored and the holdup can be assumed to be constant for the entire length of the drum. This assumption simplifies the solution method for the residence time.

5.1.1 Solution Algorithm

The following algorithm is used to solve for the solids holdup and axial flow rate in a drum with constant holdup given the initial discharge angle of the exterior flights θ_i. By repeating the procedure over the possible range of the initial discharge angle of the exterior flights, a curve for holdup versus flow rate can be generated.

1. If the drum has a centrefill with flights, a trial and error method is used to solve Equation 44 for the angle ψ_i at which an interior flight will start to discharge material.
2. Equation 91 is used to solve for the mass of material in the airborne phase M_1.
3. Equation 126 is used to solve for the holdup of material in the dense phase H_2.
4. Equation 127 is used to solve for the total holdup of the drum H.
5. Equation 86 is used to solve for the axial flow of material in the dense phase F_2.
6. Equation 55 is used to solve for the total flow rate in the drum F_0.
7. Equation 34 is used to solve for the axial flow of material in the airborne phase F_1.
8. Equation 53 is used to solve for the particle size distribution of the material inside the drum $x_{d,i}$.
9. Equation 138 is used to solve for the average residence time of the granular feed T.
10. Equation 140 is used to solve for the residence time of each particle size T_i.

5.1.2 Computer Program (DRUM1)

The computer program DRUM1 was developed to solve the residence time model using the algorithm above. The program code is given in Appendix B. The program was coded in C language, compiled using Microsoft C Optimizing Compiler Version 6.0 and run on a NEC PowerMate 386.

5.1.3 Inputs

Inputs required by the computer program DRUM1 are the dimensions which define the drum and the flights, properties of the particles and the flow rate and properties of the gas stream. Operating conditions such as the rotational speed and inclination angle are also required.

A list or table of records is used to describe the geometry of the flights. Each record consists of a pair of numbers. Each pair is an angle of rotation, θ, and the corresponding length of the horizontal line from the flight tip back to the wall or the flight itself. The angles in the list must be in ascending order. The interval of the angles does not need to be regular, so long as the data are sufficient to describe the flight.

This method of describing the flight geometry is different than that used by most authors of the previous residence time models. In the earlier models, it was common to derive equations which related the flight volume to the rotational angle to define the flight geometry. These equations were used to calculate the volume of material held by a flight. The rate of discharge was determined by computing the difference in volumes

held in the flight between two angles. The authors usually restricted their studies to one or two types of flights because it is difficult and cumbersome to derive flight volume equations for numerous types of flights.

Defining a flight by the length of the lines extending from the tip of the flight is more convenient than the previous method. Lengths are measured from a sketch of the drum cross section. The computer program determines the length of the particle surface on a flight by referring to the table at the angle of rotation minus the angle of repose. Flight discharge rates are simply calculated using Equations 37 and 38 and flight holdups are determined by numerical integration of the flight discharge rates.

5.1.4 Numerical Methods

The program uses linear interpolation to determine the particle surface lengths at intermediate angles. The data in the length table should be appropriate to accurately describe the geometry through interpolation between the points given.

The Romberg numerical integration method is used to solve the single integrals in Equations 44, 50, 58, 81, 82, 86, 91, 117-121, 124 and 125. The Romberg method was chosen because it is highly efficient and allows the degree of accuracy to be selected as an input parameter.

The double integrals in Equations 117-122 and 124-125 are rearranged to a system of two first-order ordinary differential equations. For example, consider the double integral in the third term on the right hand side of Equation 117.

$$I = \int_{\theta_i}^{\pi} \int_{\theta}^{\pi} \ell_{ex}^2 d\theta d\theta \tag{143}$$

Let $u_2 = I$ and $u_1 = du_2/d\theta$

The double integral can be redefined as the following initial value problem.

$$\frac{du_1}{d\theta} = \ell_{ex}^2 \quad \text{and} \quad \frac{du_2}{d\theta} = u_1 \tag{144}$$

$$u_2 = 0 \quad \text{at} \quad \theta = \pi \tag{145}$$

This initial value problem is solved by the computer program using a fourth order Runge Kutta routine which integrates the two differential equations simultaneously. Initially, an attempt was made to solve the double integrals using the same Romberg routine used to solve the single integrals. The Romberg routine solved the outer integral by calling the Romberg routine numerous times to solve the inner integral. Although recursive functions are possible in the C language, this method proved to be too slow. Using the Runge Kutta routine to integrate the double integrals increased the program speed significantly.

The secant method is used to solve for the initial angle of discharge of the interior flights ψ_i.

5.1.5 Model Verification

Admittedly, the model is complex and there is plenty of opportunity for errors in the derivation of the equations and the development of the computer code. Therefore, great care was taken to verify both the equations and code. The computer code has been highly modularized. Usually a separate routine is used to solve a single equation. Each routine was tested independently of the rest of the program for a large range of inputs.

Results have been checked using inputs that allow solution of the equations by hand calculations. For example, many routines were checked for EAD flights with a radius of unity. When possible, comparisons were also made with solutions given by other authors. Equations and code which apply to ranges of degree of drum loading were checked for agreement at the common limit where two different equations should give the same result. Numerical integration and interpolation routines were tested by solving problems with known results.

5.1.6 Sheikh's Experiments (1987)

The computer program DRUM1 was tested using experimental data reported by Sheikh (1987). Sheikh's long inclined drum is shown schematically in Figure 5.1. The drum was 0.385 metres in diameter and 2.35 metres long. The first 0.24 metres of length contained advancing flights to move material quickly away from the entrance. The remaining 2.10 metres was fitted with longitudinal lifting flights. These flights were removable and Sheikh experimented with configurations with 0 to 12 flights. Two shapes of flights were examined; a two-sided flight with a 90 degree angle between the sides and two-sided flight with a 135 degree angle between the sides. The drum was inclined either

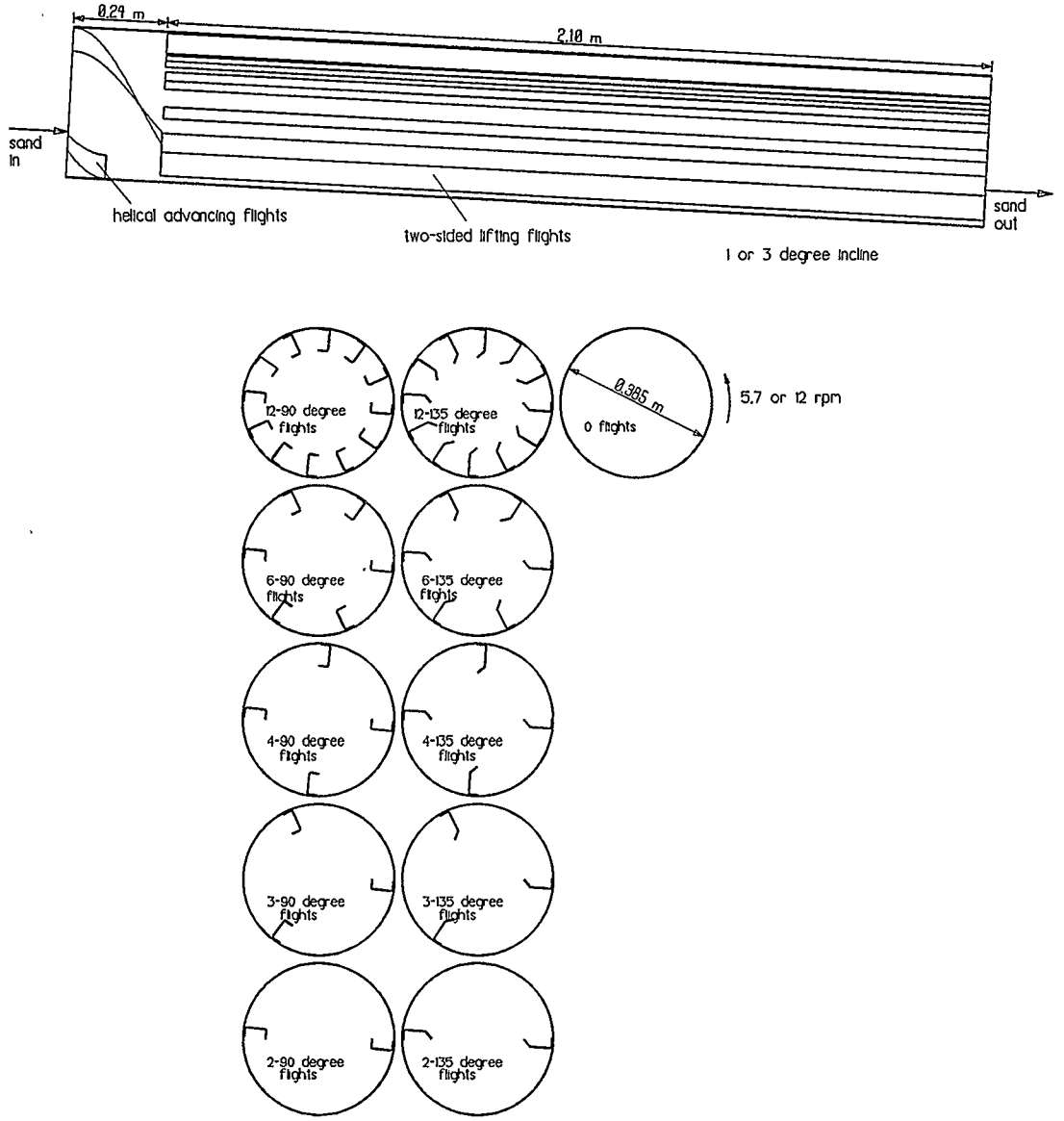


Figure 5.1 Sheikh's long inclined rotating drum and numerous flight configurations

1 or 3 degrees to horizontal and rotated at 5.7 or 12 revolutions per minute. Sheikh experimented with sand and wheat over a wide range of feed rates. All of Sheikh's experiments were without any gas flow. Both the rate and the holdup of the solids were measured by direct weighing. Properties of the sand are in Table 5.1.

Table 5.1 Properties of sand used in Sheikh's experiments

Bulk density	1517 kg/m^3
Angle of repose	35.69 degrees
Particle size distribution (microns)	wt%
>500	11.6
500 - 355	21.6
355 - 300	12.0
300 - 212	29.6
212 - 125	22.1
<125	3.2

Details of the lifting flights are shown in Figure 5.2. The figure shows the method of defining the flight by drawing and measuring the length of the horizontal lines from the flight tip back to the wall or another flight for various angles of rotation. This information is required as input to the computer program.

The computer program DRUM1 was run to determine the holdup and the flow rate over a range of initial discharge angles of the lifting flights. Figure 5.3 illustrates the holdup of sand versus initial discharge angle predicted for Sheikh's drum with twelve 90 degree flights rotated at 5.7 rpm and inclined 1 degree. The figure illustrates the relative amount of material predicted in each phase. Only about 10 per cent of the total

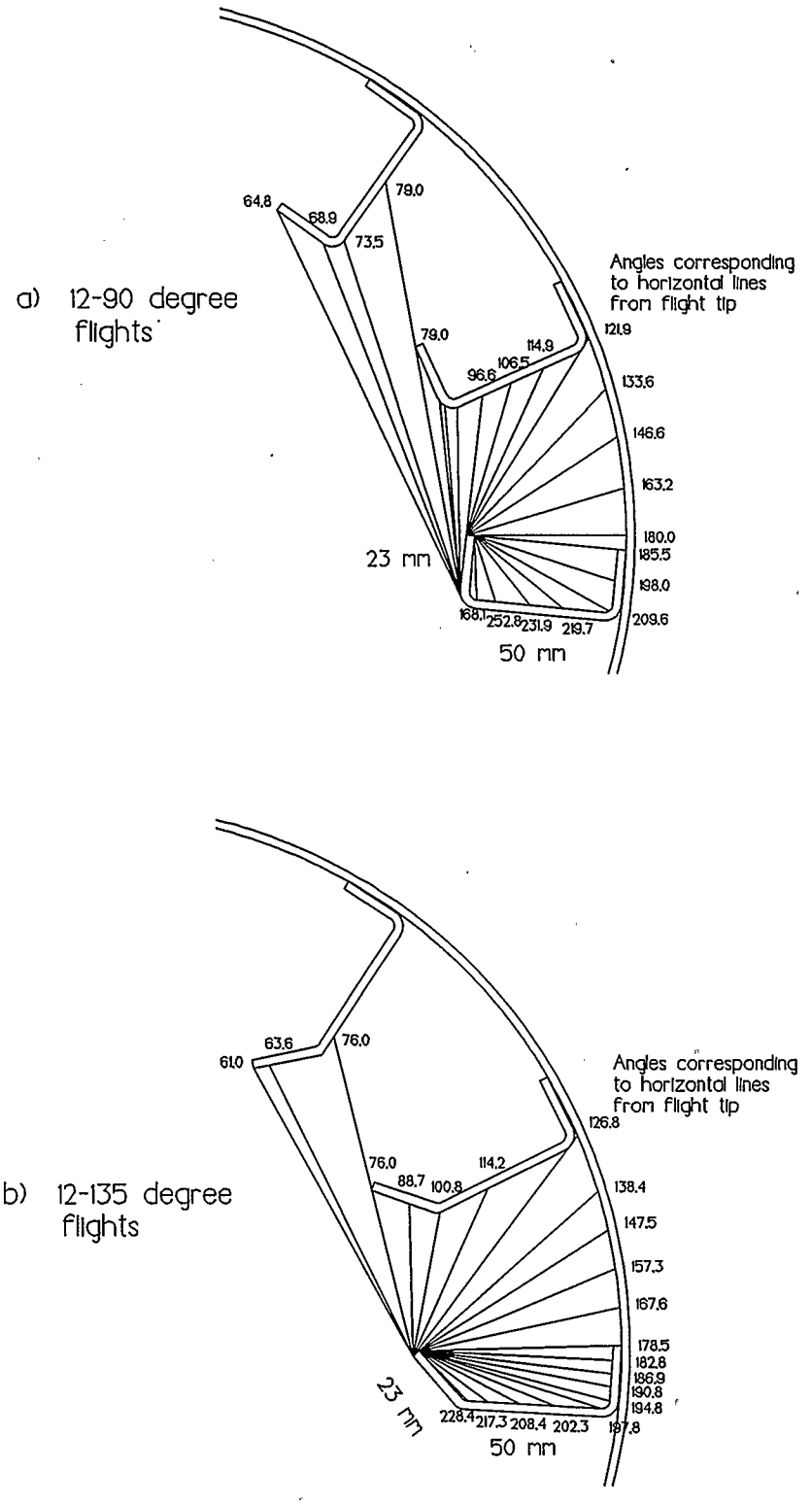


Figure 5.2 Details of lifting flights in Sheikh's drum.

holdup is in the airborne phase. In the underloaded condition, most of the holdup is in the non-discharging flights. Holdup in the discharging flights increases as the drum loading increases. An inflection point exists in the total holdup curve at design loading.

Figure 5.4 illustrates the axial flow rate versus initial discharge angle predicted for the same drum and conditions as in Figure 5.3. For this configuration, the model predicts about 3/4 of the axial flow occurs in the airborne phase when the drum is underloaded or slightly overloaded. Because the flow in the airborne phase is much larger than the flow in the dense phase, the assumption of constant holdup is reasonable for this range of loading. The movement in the dense phase increases dramatically when the drum is extremely overloaded. This corresponds to the condition when some flights at the bottom of the drum become covered. The assumption of constant holdup may not be acceptable when the drum is extremely overloaded because most of the flow now occurs in the dense phase.

In Figure 5.4, not only does the curve for the dense phase flow increase dramatically when the drum becomes extremely overloaded, but the curve is not smooth. In some cases, the same flow rate occurs for different initial discharge angles. This erratic behaviour can be explained by the approximations used for the surface length on the non-discharging flights. This topic is discussed in more detail in Section 5.1.10.

In Figure 5.5, the prediction for total holdup from Figure 5.3 is plotted against the prediction for total flow from Figure 5.4 for the corresponding initial discharge angles. The resulting curve is compared to Sheikh's experimental results. The figure

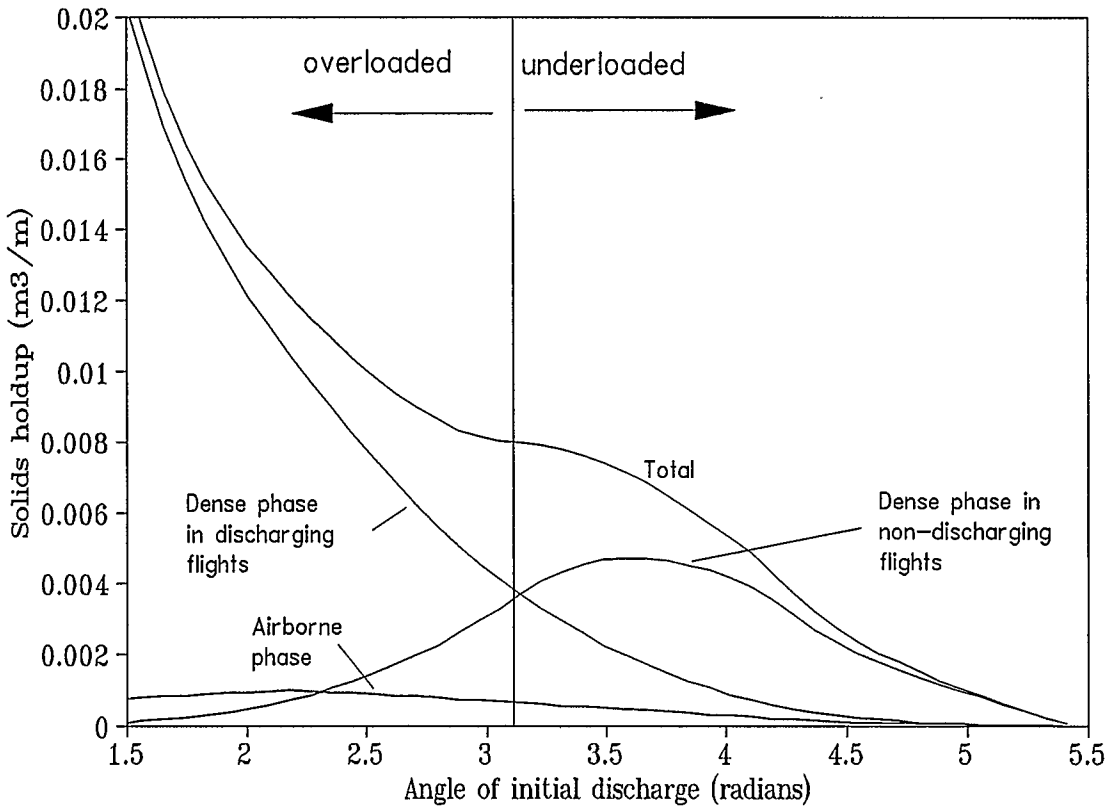


Figure 5.3 Predicted solids holdup for Sheikh's long inclined drum with 12-90 degree flights, sand, no gas flow, any incline, and 5.7 rpm

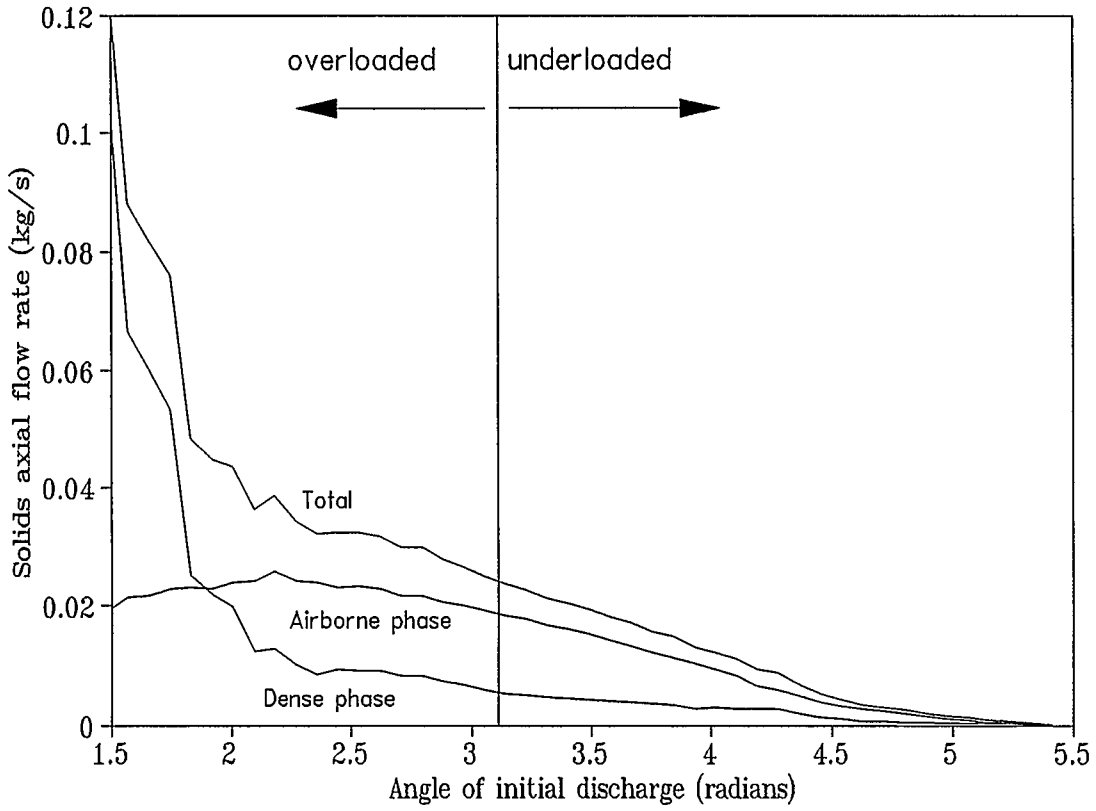


Figure 5.4 Predicted solids flow rates for Sheikh's long inclined drum with 12-90 degree flights, sand, no gas flow, 1 degree incline, and 5.7 rpm

also compares the predicted curves and experimental data for the same drum at different rotational speeds and inclinations. Figures 5.6 through 5.14 contain the same plot for drums with different number and shape of flights. Figure 5.5 indicates good agreement for the drum with twelve 90 degree flights; excellent agreement for the underloaded condition. In Figure 5.6, good agreement for the drum with twelve 135 degree flights is seen at low incline and for the underloaded condition. For the overloaded drum at the higher incline, however, experimental and predicted results stray from one another.

In Figures 5.7 through 5.14, agreement between the experimental and predicted results is good for the underloaded condition but worsens for the overloaded condition as the number of flights decreases. For an overloaded drum with two flights, the predicted residence time is only about half the measured result.

5.1.7 Kelly's Experiments (1969)

The computer program DRUM1 was also tested using data reported by Kelly (1969). Kelly's drum, shown in Figure 5.15, was 0.305 metres in diameter and 1.83 metres in overall length. The first 0.14 metres of length contained advancing flights to move material quickly away from the entrance. The remaining 1.69 metres was fitted with eight equal-angular distribution (EAD) lifting flights. The drum was inclined 2 or 6 degrees to horizontal and rotated at 8 and 24 revolutions per minute. The solids feed material was pumice granules with properties as shown in Table 5.2. Experiments were performed with no air flow and with countercurrent air flow at several rates. Both the rate and the holdup of the solids at steady-state were measured volumetrically.

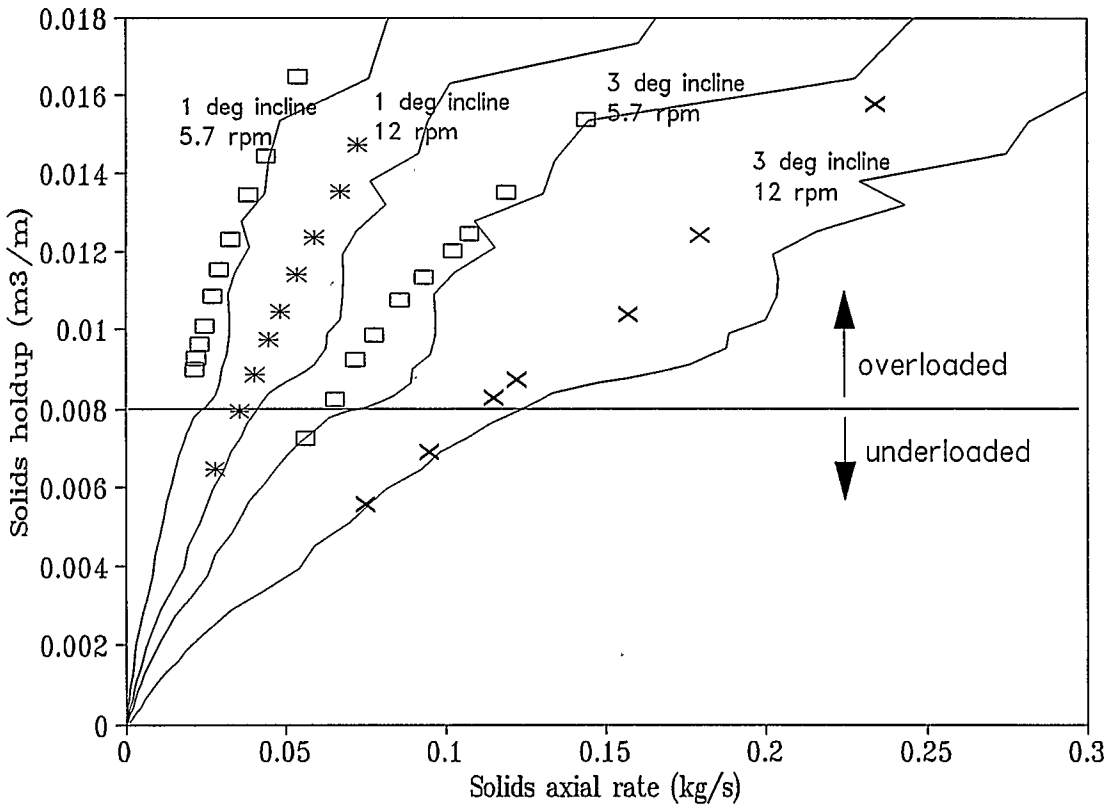


Figure 5.5 Comparison of experimental and predicted results for Sheikh's drum with 12-90 degree flights, sand and no air flow.

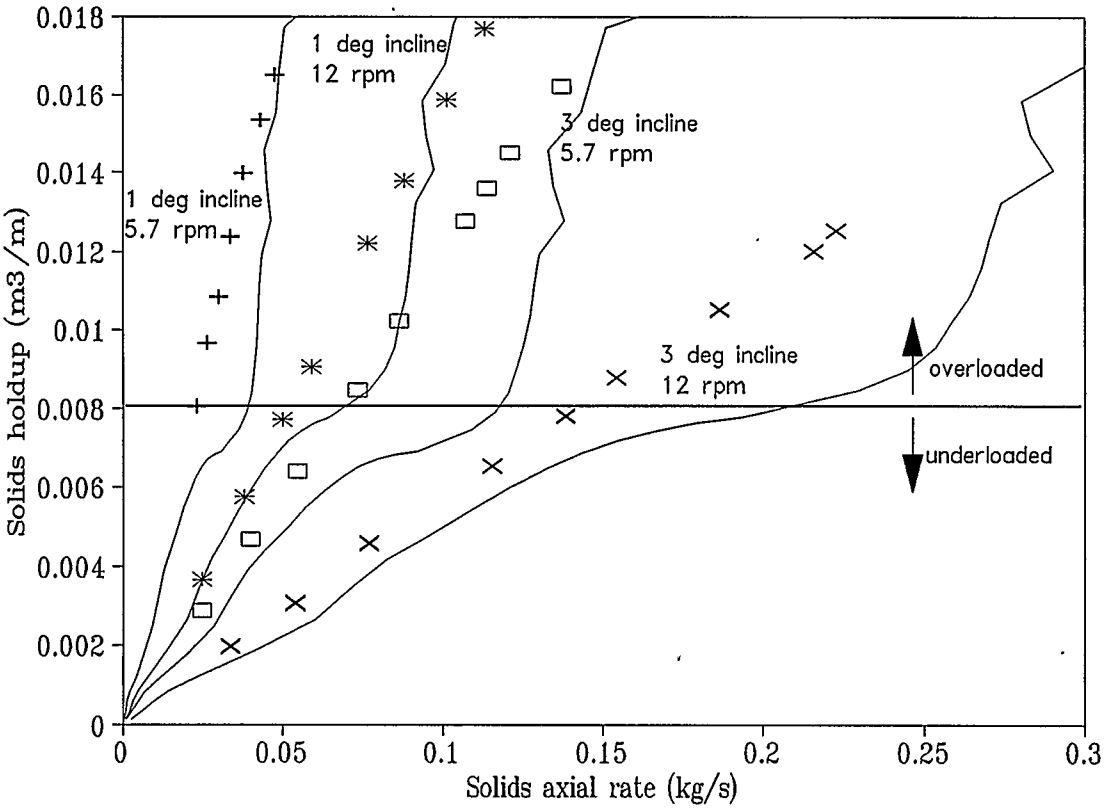


Figure 5.6 Comparison of experimental and predicted results for Sheikh's drum with 12-135 degree flights, sand and no air flow.

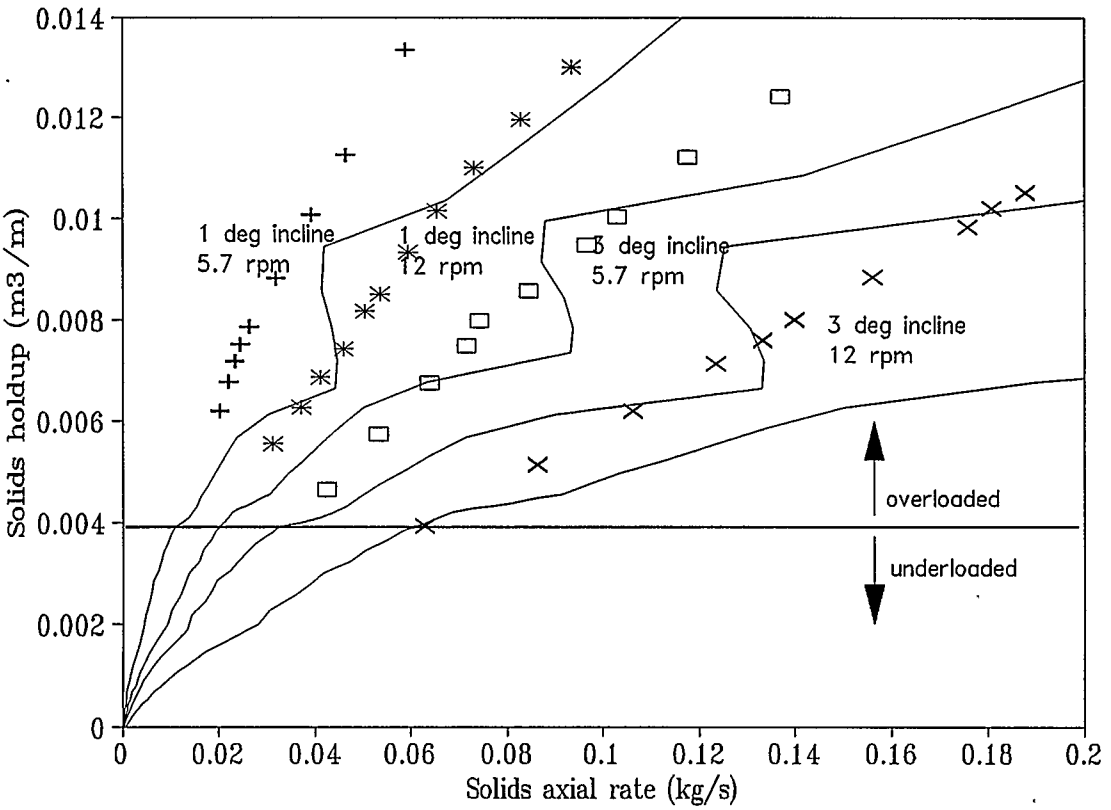


Figure 5.7 Comparison of experimental and predicted results for Sheikh's drum with 6-90 degree flights, sand and no air flow.

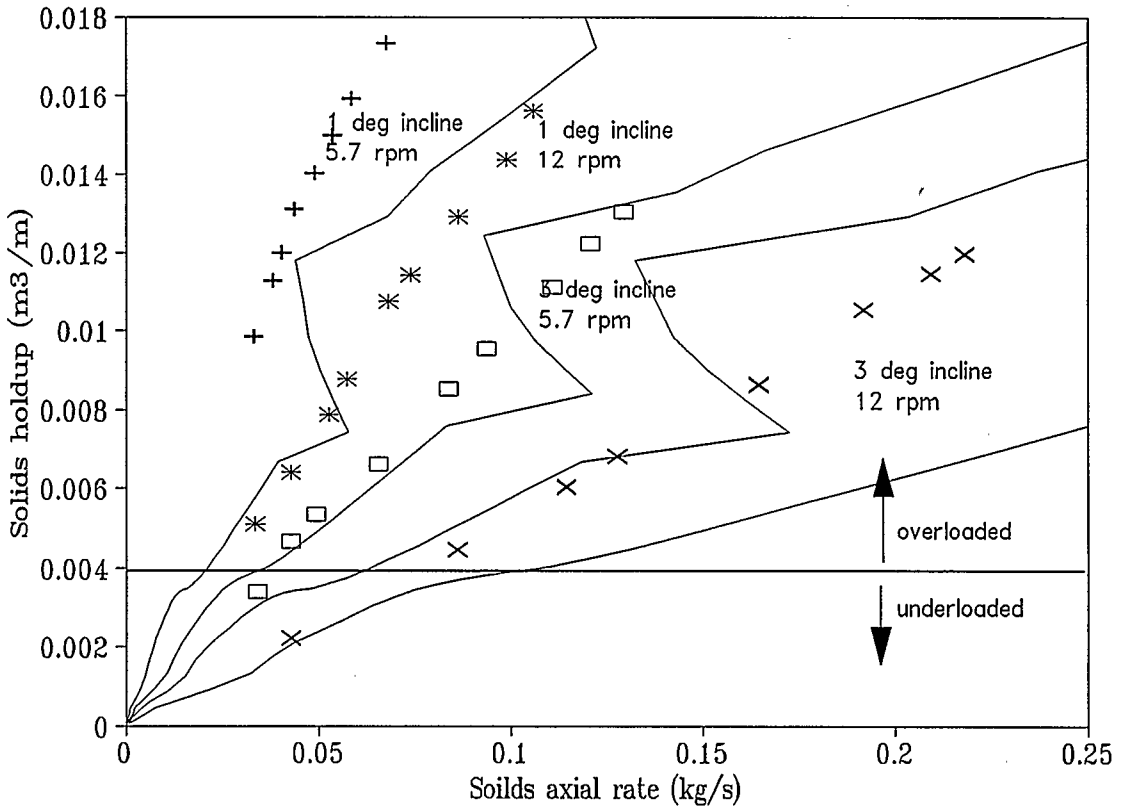


Figure 5.8 Comparison of experimental and predicted results for Sheikh's drum with 6-135 degree flights, sand and no air flow.

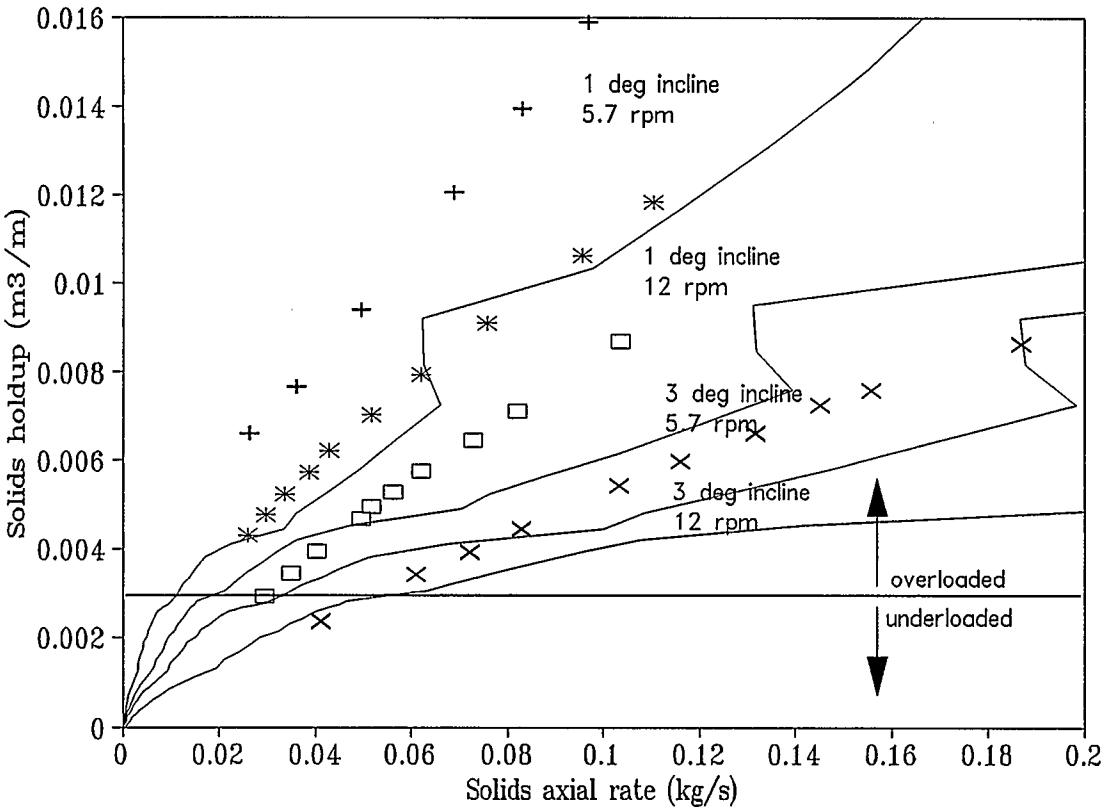


Figure 5.9 Comparison of experimental and predicted results for Sheikh's drum with 4-90 degree flights, sand and no air flow.

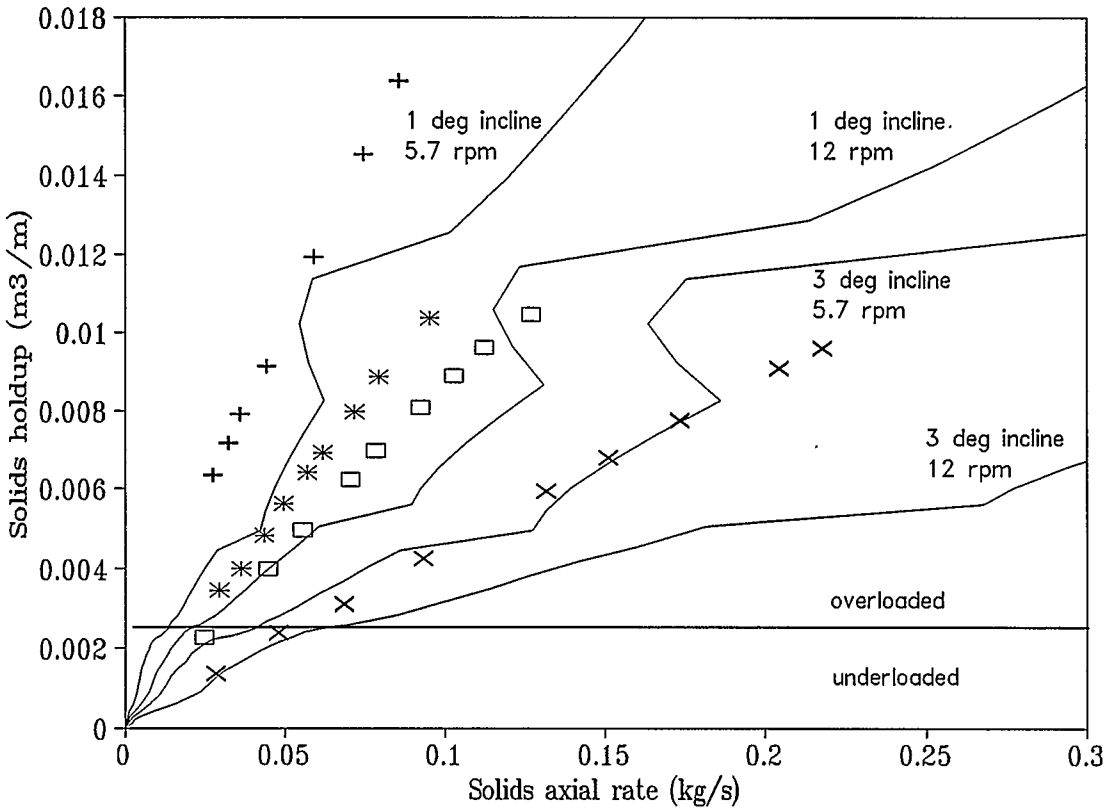


Figure 5.10 Comparison of experimental and predicted results for Sheikh's drum with 4-135 degree flights, sand and no air flow.

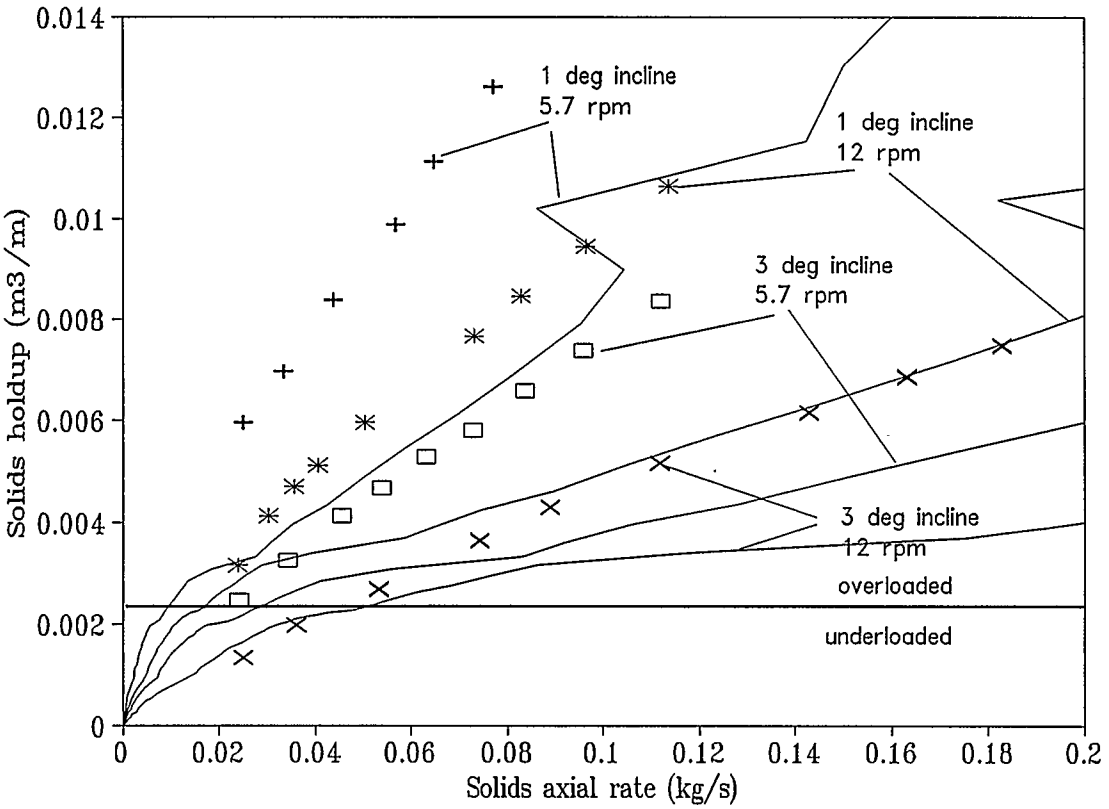


Figure 5.11 Comparison of experimental and predicted results for Sheikh's drum with 3-90 degree flights, sand and no air flow.

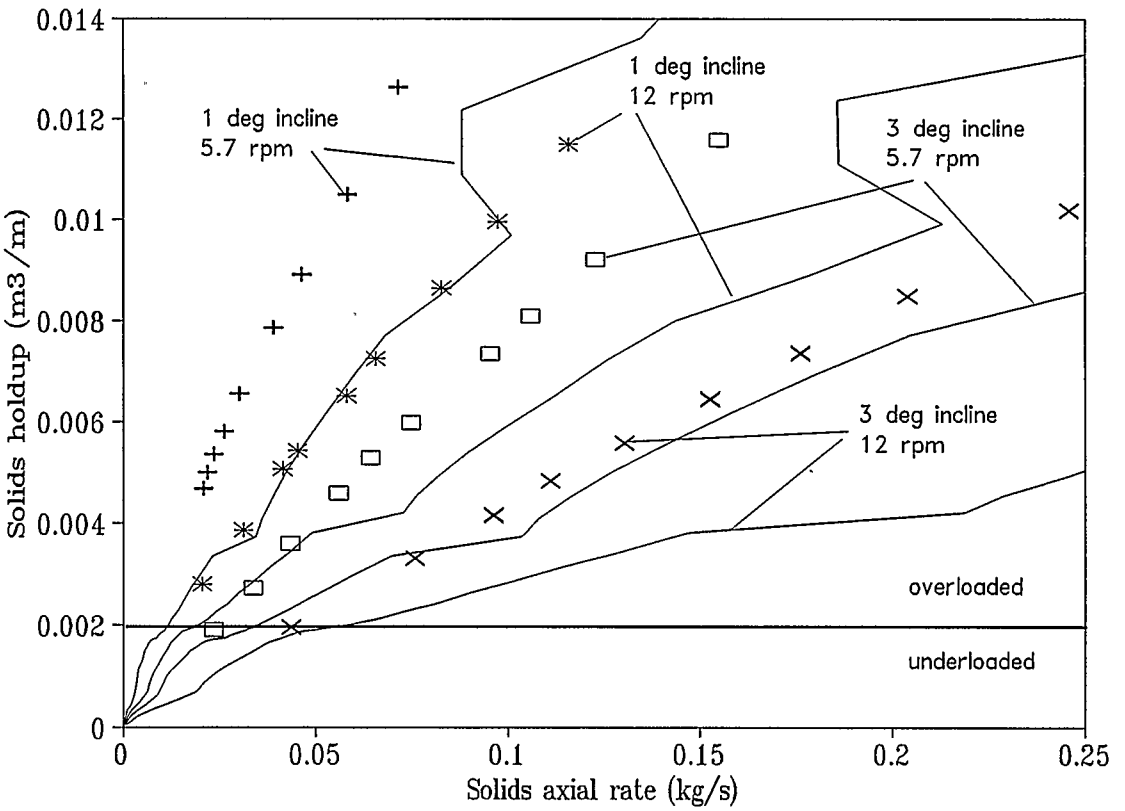


Figure 5.12 Comparison of experimental and predicted results for Sheikh's drum with 3-135 degree flights, sand and no air flow.

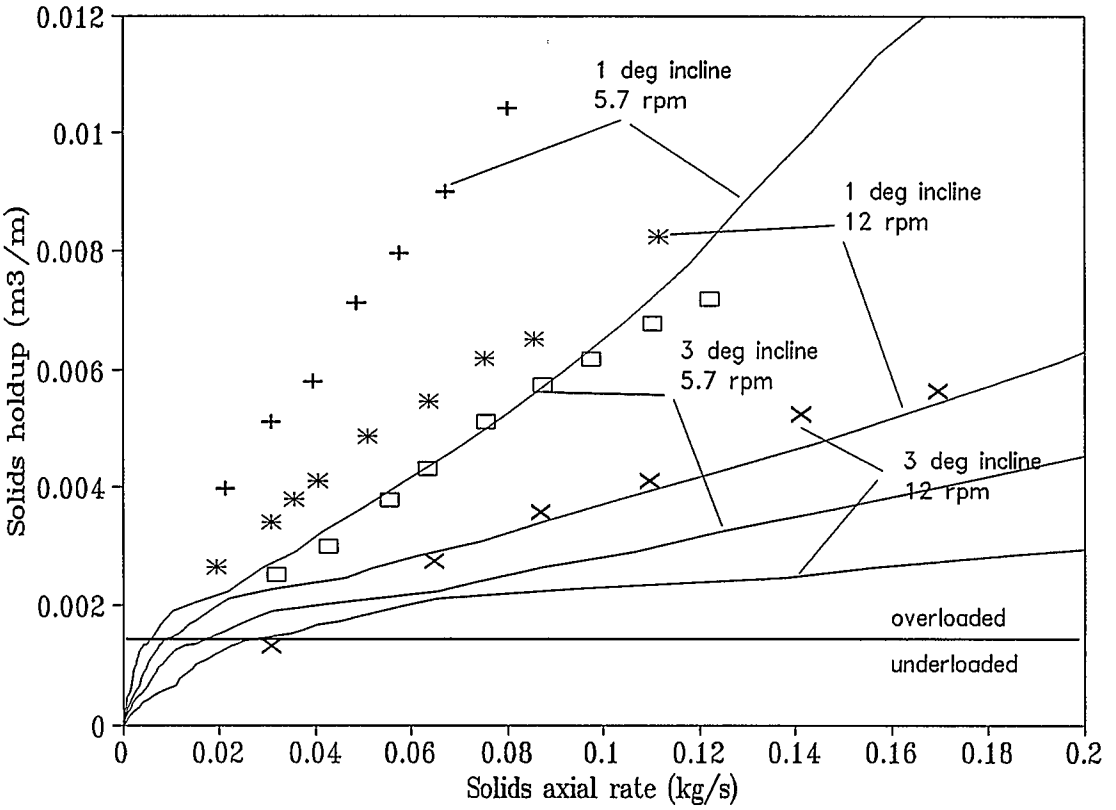


Figure 5.13 Comparison of experimental and predicted results for Sheikh's drum with 2-90 degree flights, sand and no air flow.

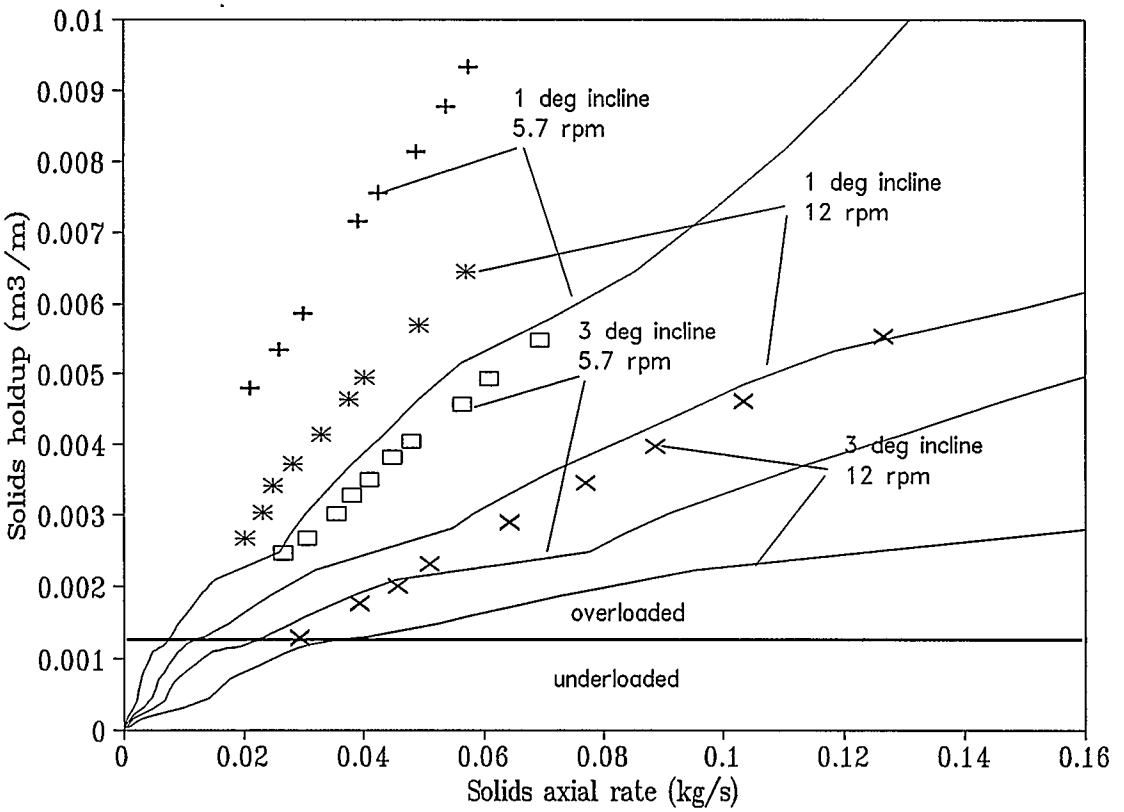


Figure 5.14 Comparison of experimental and predicted results for Sheikh's drum with 2-135 degree flights, sand and no air flow.

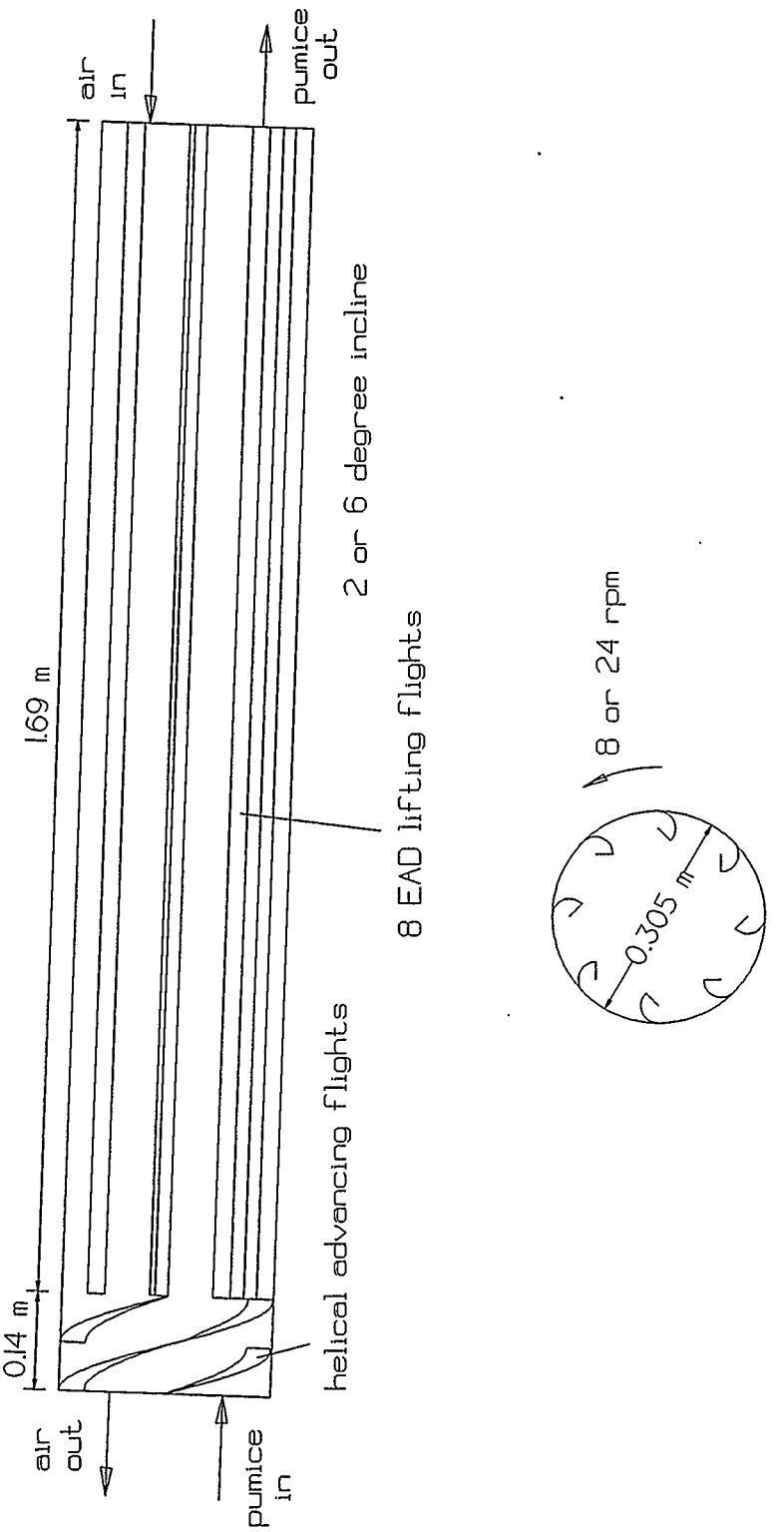


Figure 5.15 Kelly's long inclined drum with 8 EAD flights

Table 5.2 Properties of pumice used in Kelly's experiments

bulk density	591 kg/m^3
particle density	1147 kg/m^3
particle diameter	B.S. sieve range -7+14 mesh 1750 microns (assumed)
angle of repose	45.6 degrees
sphericity	1.0 (assumed)

Details of the EAD flights are shown in Figure 5.16.

Kelly's experimental results are compared to predicted results in Figures 5.17 through 5.20. The predicted residence time is almost always too high. Agreement worsens as the drum incline and the rotational speed increases. However, the predicted increase in holdup due to the countercurrent air flow agrees with the magnitude of the observed effect.

5.1.8 O'Donnell's Experiments (1975)

The computer program DRUM1 was also tested using data reported by O'Donnell (1975). O'Donnell's drum, shown in Figure 5.21, had the same overall dimensions as Kelly's drum; 0.3048 metres in diameter and 1.829 metres in length. The first 0.141 metres of length contained advancing flights to move material quickly away from the entrance. The remaining 1.688 metres were fitted with eight centrally biased distribution

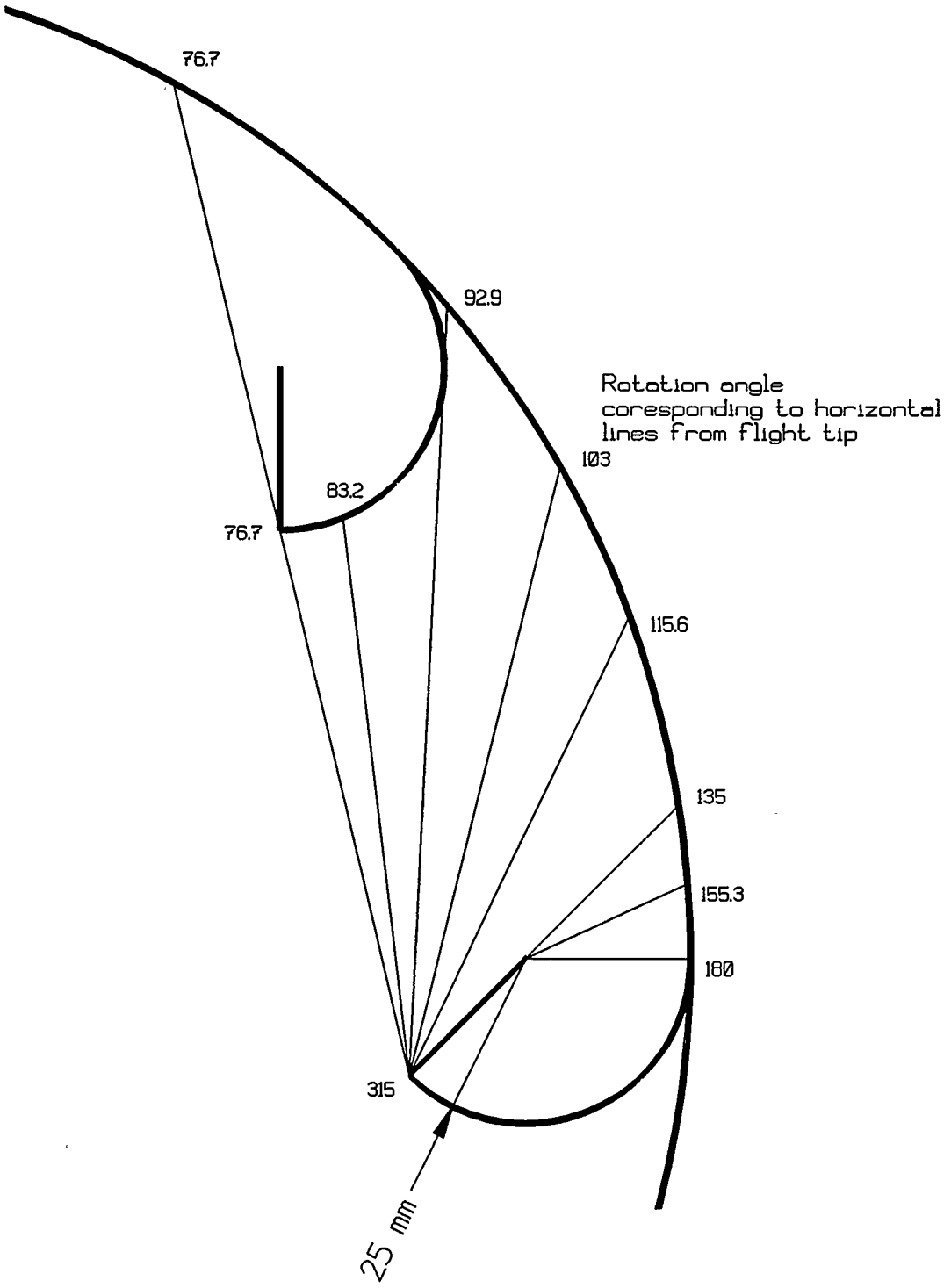


Figure 5.16 Details of flights in Kelly's drum.

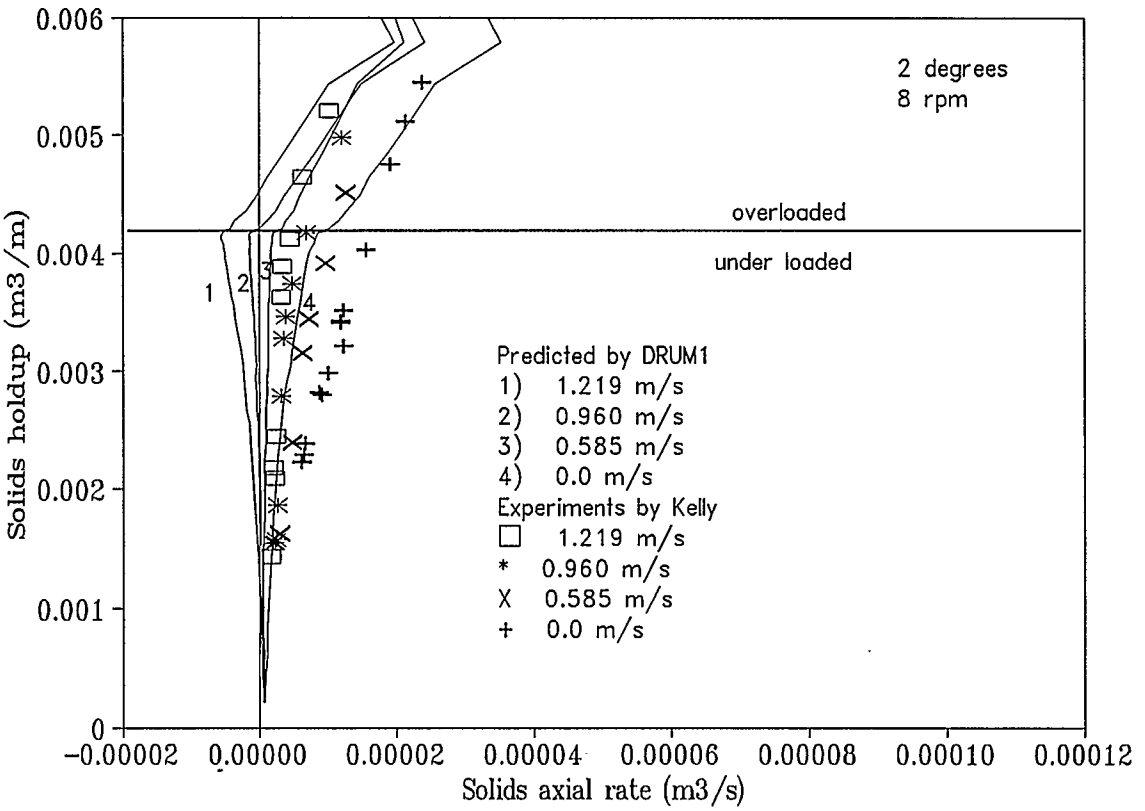


Figure 5.17 Comparison of experimental and predicted results for Kelly's drum with 8 EAD flights, pumice, 2 degree incline and 8 rpm.

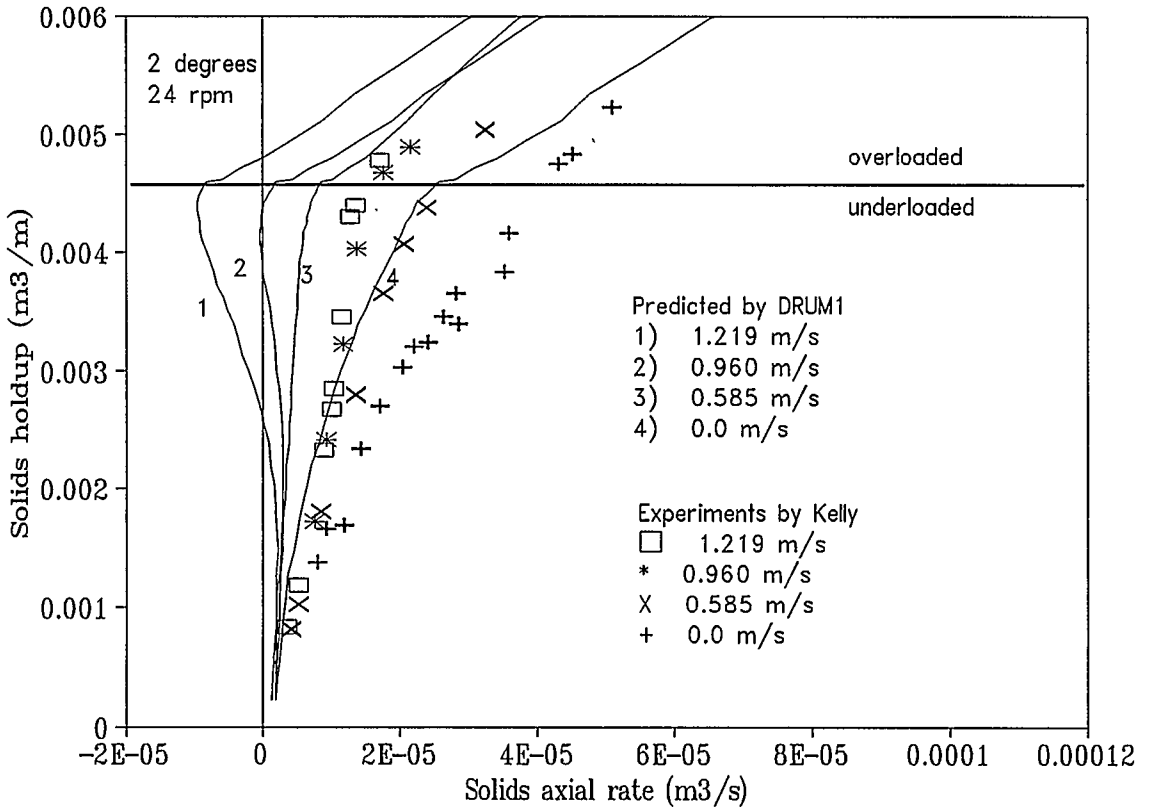


Figure 5.18 Comparison of experimental and predicted results for Kelly's drum with 8 EAD flights, pumice, 2 degree incline and 24 rpm speed.

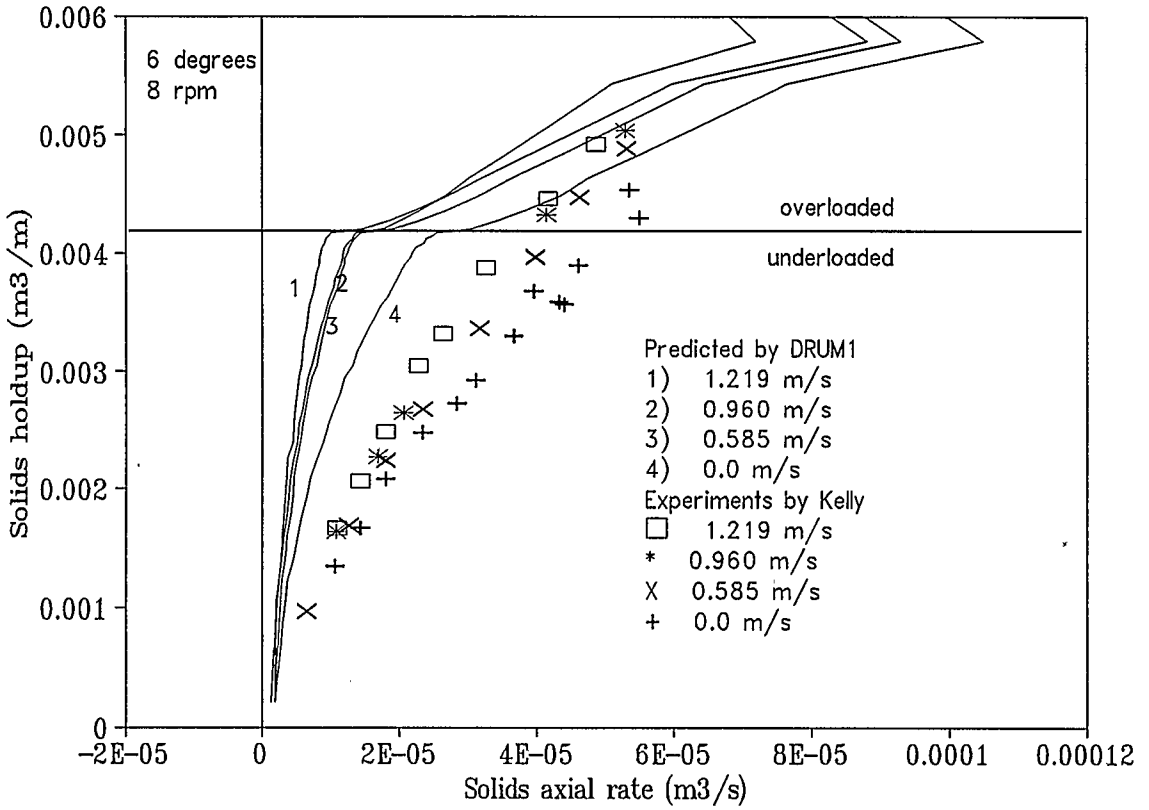


Figure 5.19 Comparison of experimental and predicted results for Kelly's drum with 8 EAD flights, pumice, 6 degree incline and 8 rpm speed.

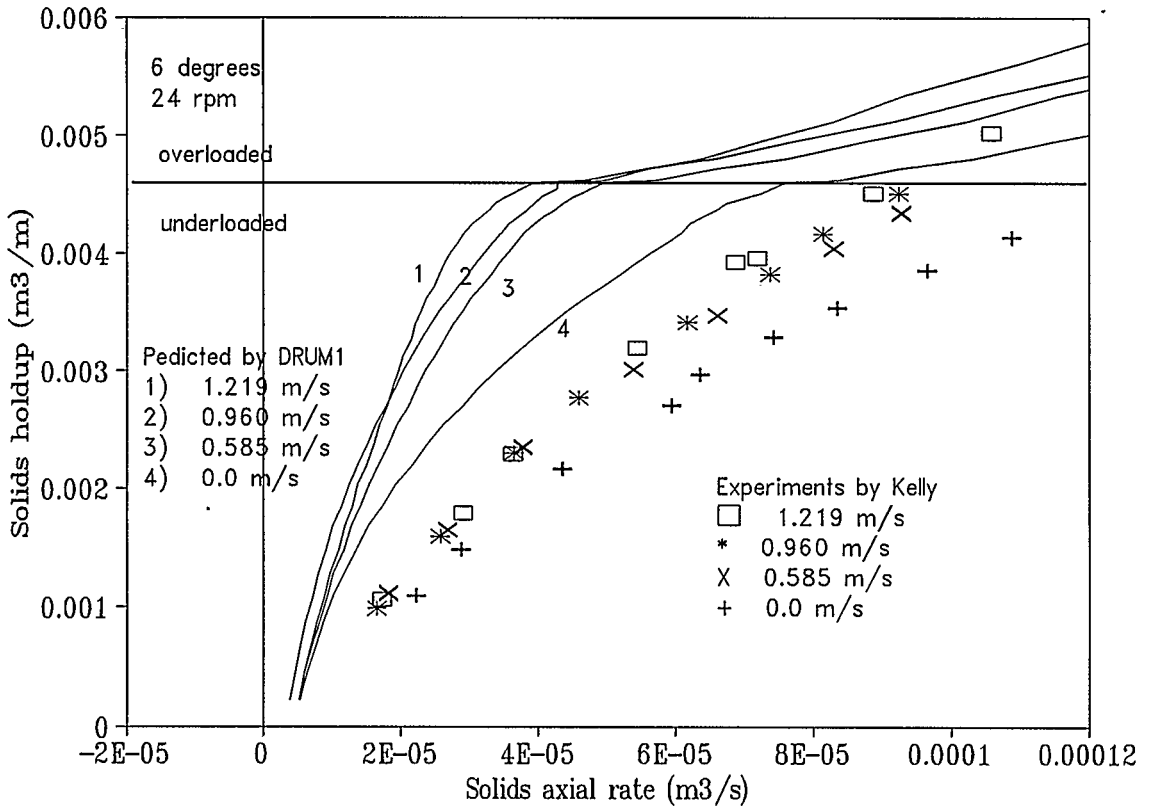


Figure 5.20 Comparison of experimental and predicted results for Kelly's drum with 8 EAD flights, pumice, 6 degree incline and 24 rpm speed.

(CBD) lifting flights. The drum was inclined 2 or 6 degrees to horizontal and rotated at 8 and 24 revolutions per minute. The solids feed material was pumice with properties as shown in Table 5.3. The volume-surface mean diameter was determined using the following equation.

$$\overline{D_p} = \frac{1}{\sum \frac{x_i}{D_{p_i}}} \tag{146}$$

Details of the CBD flights are shown in Figure 5.22. Experiments were performed with no air flow and with countercurrent air flow at several rates. The residence time was determined using radioactive tracers.

Table 5.3 Properties of pumice used in O'Donnell's experiments

bulk density	950.5 kg/m^3	
particle density	1367.3 kg/m^3	
angle of repose	45.6 degrees	
particle size distribution		
microns		wt %
2811-2411		8.4
2411-2057		38.0
2057-1676		37.7
1676-1204		13.4
1204-1003		1.8
1003-853		0.5
diameter (volume-surface mean)		1927 micron
sphericity		1.0 (assumed)

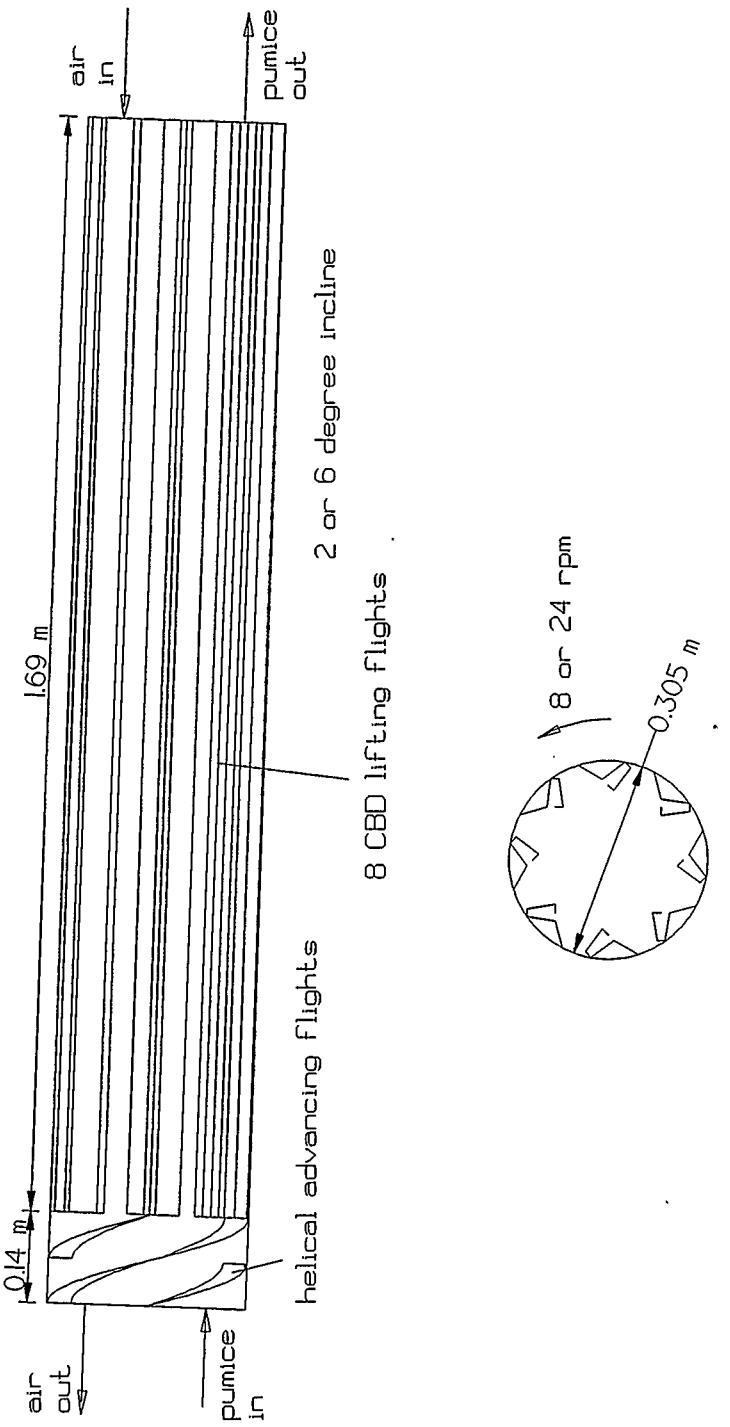


Figure 5.21 O'Donnell's long inclined drum with 8 CBD flights

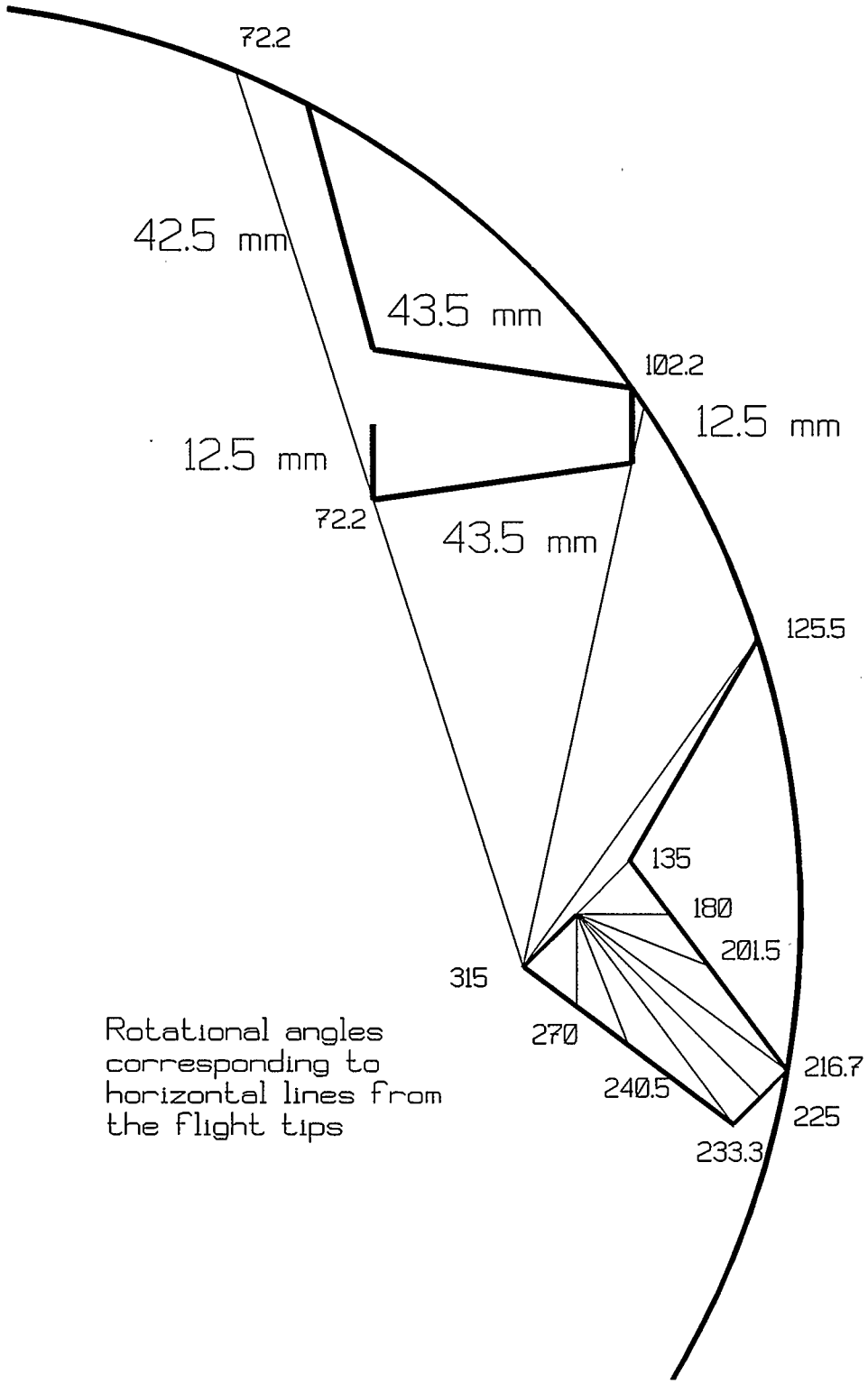


Figure 5.22 Details of flights in O'Donnell's drum.

O'Donnell's experimental results for the drum inclined 6 degrees to horizontal and rotated at 8 rpm are compared to predicted results in Figure 5.23. The predicted residence time for the underloaded condition is always too high, while the predicted residence time for the overloaded condition is always too low. The magnitude of the predicted effect of the gas velocity agrees with the experimental results.

The predictions shown in Figure 5.23 are for a single mean particle size. The computer program was also run with the particle size distribution given in Table 5.3. The program did not always obtain a reasonable solution. At some conditions, the small particles would move in the opposite direction of the main feed resulting in negative particle fractions. However, when this did not occur, the computer program predicted that the residence time was slightly larger when the distribution of particle sizes was used rather than a single mean particle diameter.

5.1.9 Saeman's Experiments (1954)

The computer program DRUM1 was also tested using data reported by Saeman (1954). Saeman's plant-size cooler, shown in Figure 5.24, was 1.83 metres in diameter and 10.67 metres in length. The first 1.52 metres of length contained helical flights to move material quickly away from the entrance. The next 7.01 metres were fitted with twelve two-sided lifting flights. The remaining 2.14 meters was fitted with mixing vanes. The drum was inclined 3.8 degrees to horizontal and was rotated at 4.8 revolutions per minute.

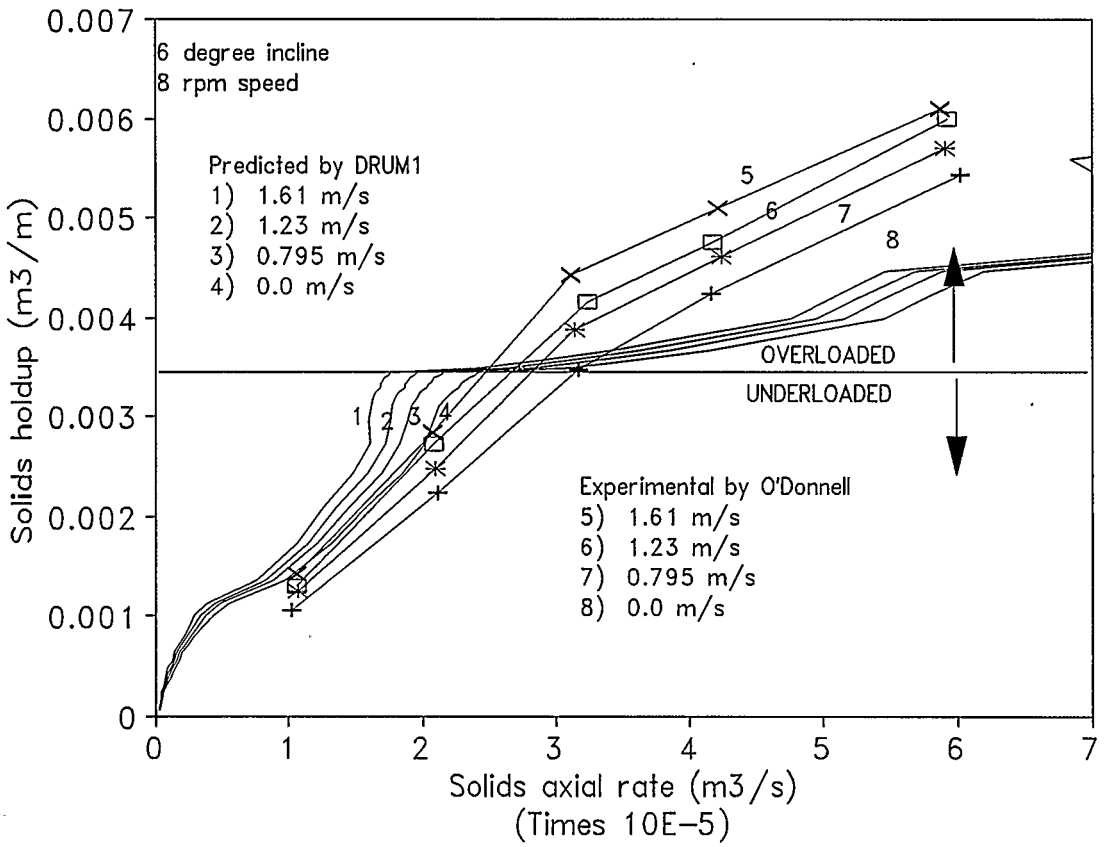


Figure 5.23 Comparison of experimental and predicted results for O'Donnell's drum with 8 CBD flights, pumice, 6 degree incline and 8 rpm speed.

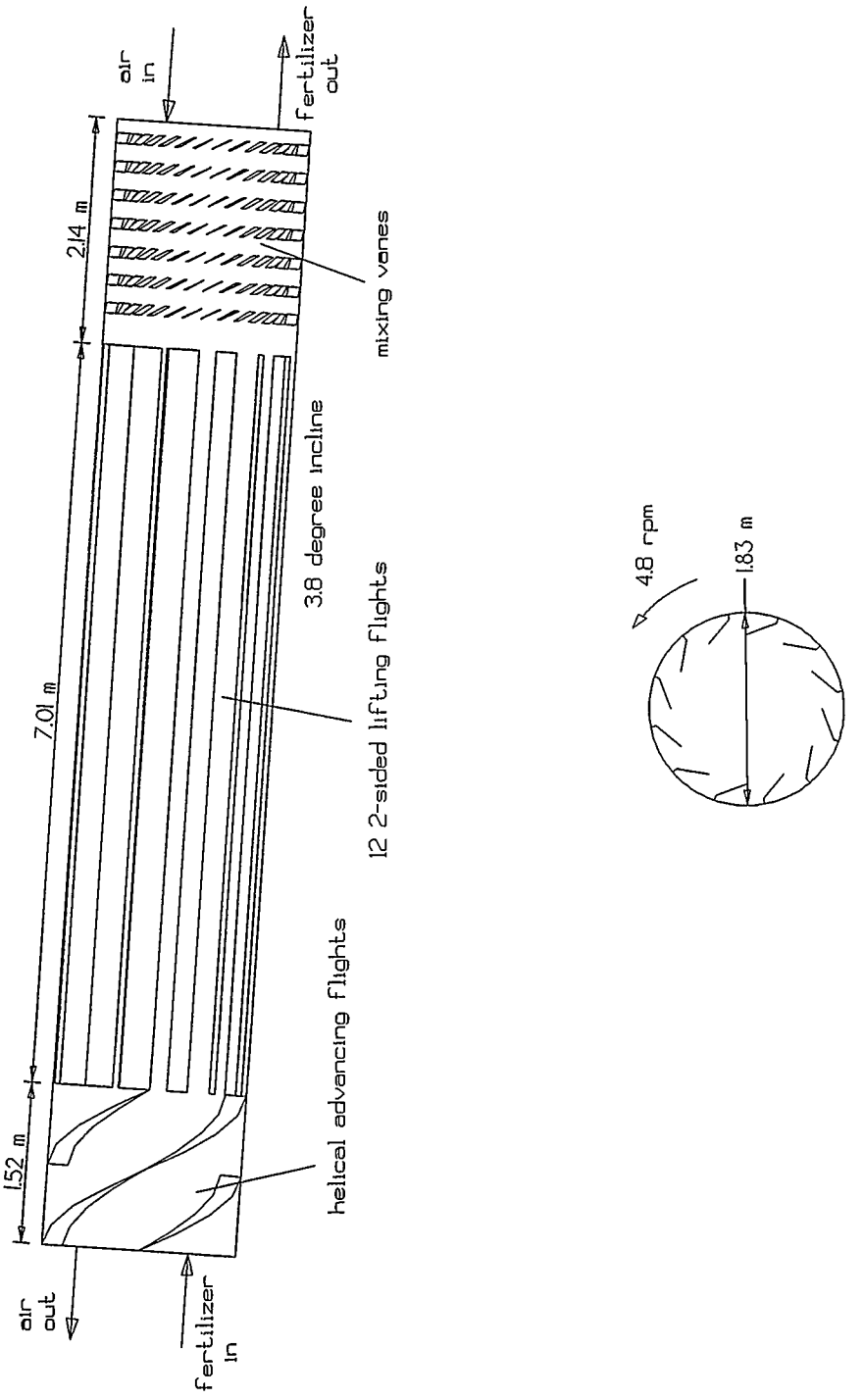


Figure 5.24 Saeman's plant-size cooler with 12 2-sided flights

The solids feed material was ammonium nitrate fertilizer with properties as shown in Table 5.4. Air flow was countercurrent. The variation in air rates and feed rates tested were small.

Table 5.4 Properties of ammonium nitrate fertilizer used in Saeman's cooler

bulk density	962 kg/m^3
particle density	1367.3 kg/m^3 (assumed)
particle diameter	599 micron
angle of repose	45 degrees (assumed)
sphericity	1.0 (assumed)

Details of the lifting flights are shown in Figure 5.25.

The predicted holdup of the drum is compared to Saeman's reported observations in Figure 5.26. The predicted holdup is only slightly larger than the observations reported by Saeman. As observed by Saeman, the range of air rates tested had little effect on the holdup.

5.1.10 Discussion of Results

Application of the model DRUM1 to published experimental data showed that the model's ability to predict the relationship between solids holdup and feed rate depends on the number and shape of the flights and the degree of loading of the drum. Reasonable agreement between predictions and measurements was obtained for underloaded drums with two-sided flights. Reasonable agreement was also obtained for

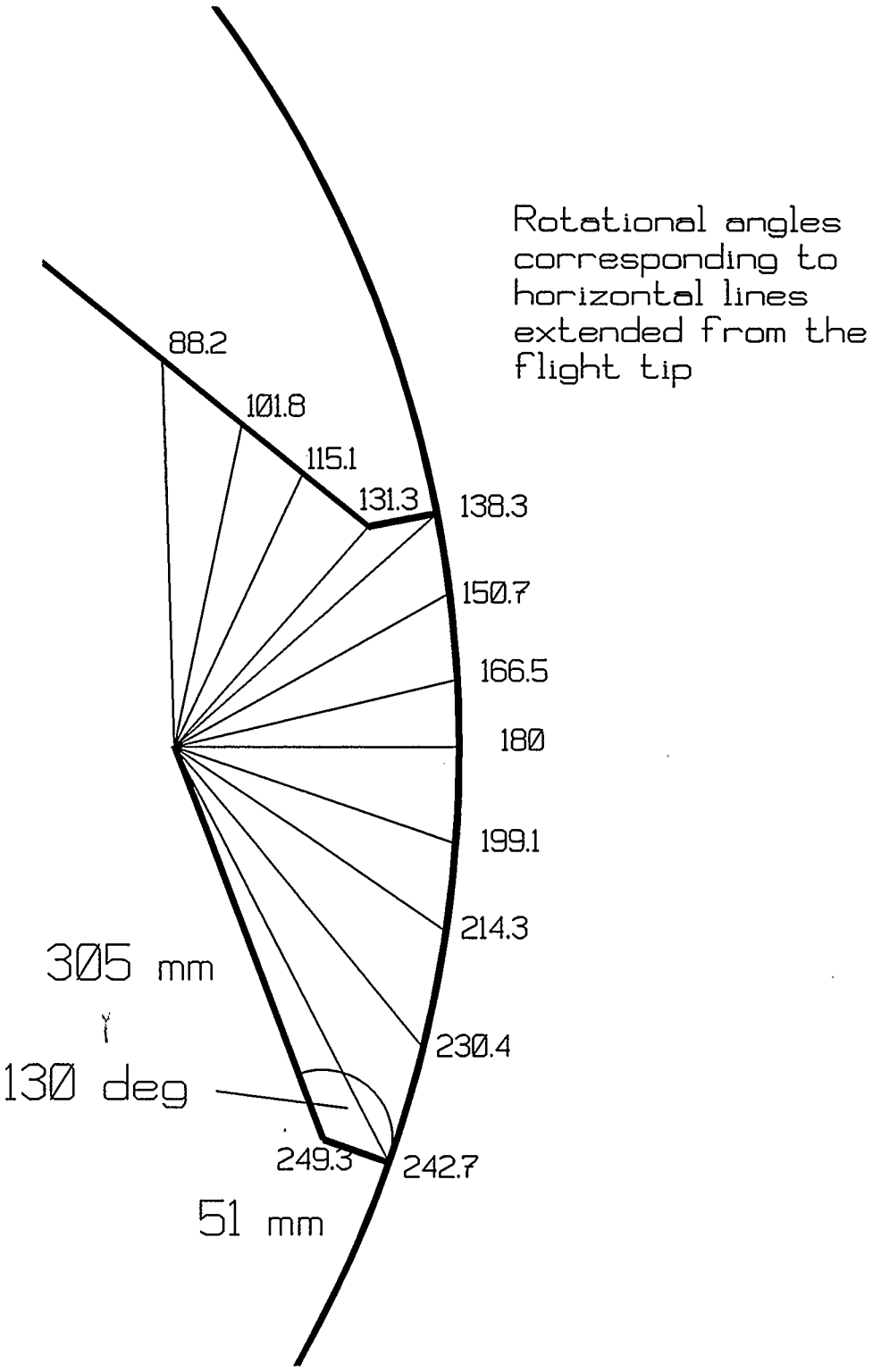


Figure 5.25 Details of lifting flights in Saeman's plant-size cooler

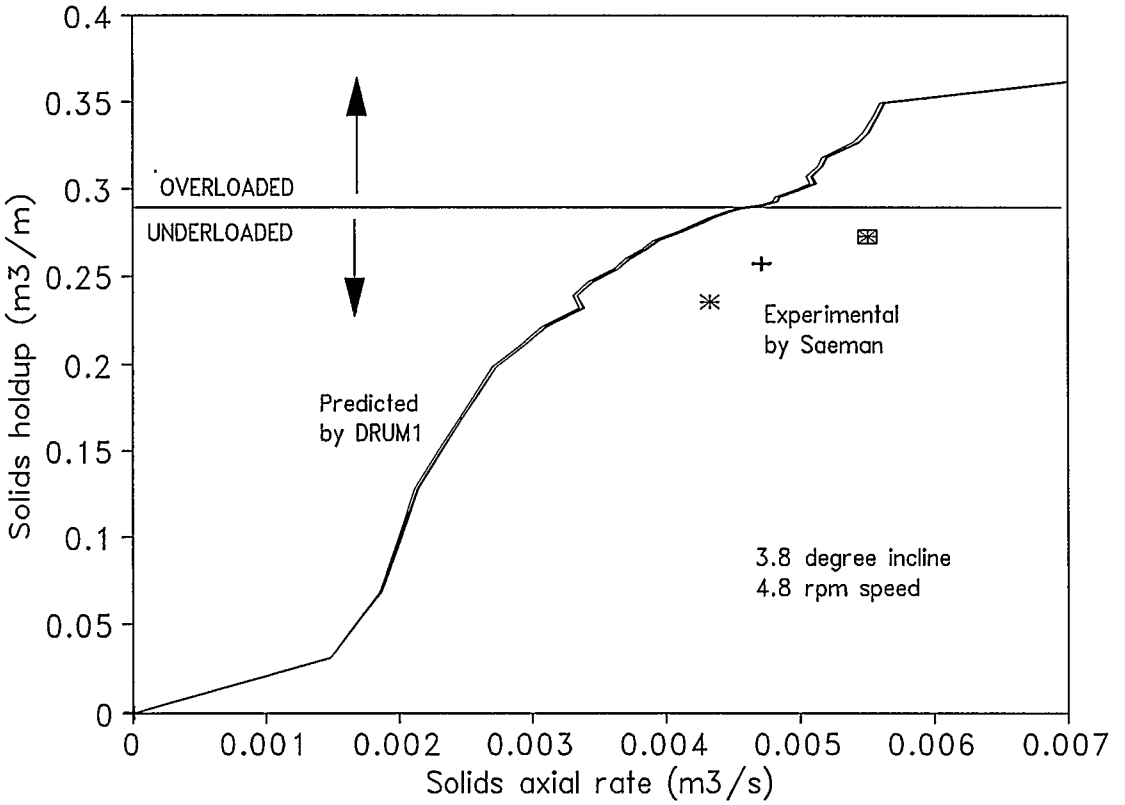


Figure 5.26 Comparison of experimental and predicted results for Saeman's plant-size cooler with 12 2-sided flights, ammonium nitrate fertilizer, 3.8 degree incline and 4.8 rpm speed.

overloaded drums with two-sided flights, but only for a drum with 12 flights. The model predictions for solids residence time were too low for all overloaded drums with fewer than 12 flights. Predictions for residence times were too high for underloaded drums with EAD and CBD flights.

The determination of holdup, given the initial discharge angle, is quite straightforward. No assumptions or approximations need to be made. Therefore, failure of the model to predict the residence time must occur in the determination of the solids flow rate.

The model was able to determine the solids flow rates in underloaded drums with two-sided flights, but calculated flow rates that were too low for underloaded drums with EAD and CBD flights. The main difference between these flights, is that EAD and CBD flights distribute material to the far side of the drum, while two-sided flights discharge little material past the vertical centre line. Due to this behaviour, in drums with EAD or CBD flights, there is significant material in all the non-discharging flights.

Flow in the non-discharging flights is determined from the surface length c_{ex} of material held in the flight. Approximations for the surface length c_{ex} are stated in Equations 76 and 77. In Equation 76, the surface length is stated to be zero (and therefore there is no movement) in non-discharging flights more than one flight spacing from the initial discharge angle. This appears to be a good approximation for drums with two-sided flights but a poor approximation for drums with EAD or CBD flights.

The model calculated flow rates that were too high for all overloaded drums. Disagreement between predictions and measurements increased as the number of flights

was decreased. In the model, most of the flow in overloaded drums occurs in the non-discharging flight that is within one flight spacing of the initial discharge angle. The flow is determined from the surface length c_{ex} which was assumed in Equation 77 to be equal to the surface length $\ell_{ex,\theta i}$ when the flight starts to discharge. This assumption happened to be suitable for the drum with 12 flights but became less so for the drums with fewer flights.

Sometimes the model predicts that for a single flow rate and all other variables remaining constant, three different holdup can occur. This behaviour was predicted only for overloaded drums at some feed rates. Although multiple holdups for a single feed rate may not be impossible, this behaviour has not been observed in any experimental results. The model predicts that holdup always increases as the initial discharge angle is decreased. The flow rate also increases as the initial discharge angle decreases except for overloaded drums at some feed rates. Because the flow rate in overloaded drums is mostly a result of the surface length c_{ex} of material in the non-discharging flights, the approximations for c_{ex} given by Equations 76 and 77 are the most likely cause of this misbehaviour.

It is obvious that the model could be improved if better approximations for the surface length c_{ex} of material in the non-discharging flights were used. However, a more accurate but not too complex method of determining or approximating these lengths is not apparent.

5.2 Programs DRUM2 and DRUM3 - Holdup Varies along Drum Length

If a drum is not long enough or the solids flow in the airborne phase is not large enough to be able to assume constant holdup along the drum length, then the change in holdup along the length of the drum must be considered in solving the residence time of the solids. In programs DRUM2 and DRUM3, the holdup is allowed to vary along the drum length.

Program DRUM2 is very similar to program DRUM1, but instead of determining the solids flow rate, the solids flow rate is an input and the program DRUM2 determines the change in holdup along the drum length. Most of the same routines are called by programs DRUM1 and DRUM2. Only two new routine were written for DRUM2. The main purpose of program DRUM2 was to check these new routines over the range of possible initial discharge angles of the exterior flights.

Program DRUM3 calculates the total holdup of a drum in which the holdup may vary along the drum length. All the routines called by programs DRUM2 and DRUM3 are the same. The difference between the programs is that DRUM3 integrates the change in holdup along the drum length. The integration is done using the same Runge Kutta routine used to solve the double integrals in the holdup equations.

5.2.1 Solution Algorithm (DRUM3)

The following is the algorithm used to solve for the residence time of a drum given the total flow rate and the initial discharge angle of the exterior flights at one end of the drum (usually the solids discharge end).

1. If the drum has a centrefill with flights, a trial and error method is used to solve Equation 44 for the angle ψ_i at which an interior flight will start to discharge material.
2. Equation 91 is used to solve for the mass of material in the airborne phase M_1.
3. Equation 126 is used to solve for the holdup of material in the dense phase H_2.
4. Equation 127 is used to solve for the total holdup H at one end of the drum.
5. The cumulative holdup of the drum is initialized and the Runge Kutta routine is called to integrate holdup over the length of the drum. The mean total holdup of the drum is found using Equation 137 and the residence time is found using Equation 138.

The Runge Kutta routine requires a routine that returns the rate of change of the initial discharge angle and the holdup with respect to the drum length and the holdup. The algorithm for this routine, given the current initial discharge angle of the exterior flights, is:

1. If the drum has a centrefill with flights, the initial discharge angle of the interior flights ψ_i is found by trial and error solution of Equation 44.
2. A guess for the dense phase flow rate F_2 is made and Equation 55 is used to calculate the total rate F_0. If the calculated total rate does not match the total rate given in the inputs, then the guess for the dense phase rate is revised. This step is repeated until the calculated total rate equals the total rate given in the inputs.
3. The flow rate of the airborne material F_1 is determined using Equation 34.
4. Equation 91 is used to solve for the holdup of material in the airborne phase H_1.

5. The slope of the dense phase on the non-discharging flights γ and the change in the holdup along the drum length dH/dz are found using Equations 135 and 134 respectively.

6. The change in holdup with respect to a change in the initial discharge angle of the exterior flights $dH/d\theta$, is determined using Equations 128 to 133 which applies to the current loading condition.

7. The dense phase holdup H_2 is determined using Equation 126 and the total holdup H is determined using Equation 32.

5.2.2 Computer Program (DRUM3)

Only a few routines are added to the program DRUM1 to make the programs DRUM2 and DRUM3 which allows for the holdup to change along the drum length. The additional routines required by these programs are also given in Appendix B.

5.2.3 Inputs

Inputs for program DRUM3 are the same as the inputs required by the program DRUM1, except the initial discharge angle of the exterior flights at one end of the drum and the total solids flow rate must be given. Normally, the initial angle of flight discharge is given at the solids exit end of the drum. If there is no weir at the drum exit, an angle slightly lower than the final discharge angle of the exterior flights can be given as the initial discharge angle at the drum exit. A weir at the exit will decrease the initial discharge angle.

5.2.4 Numerical Methods

The same Runge Kutta integration routine used to solve the double integrals for determining the average flight holdup is used to solve holdup along the length of the drum. The secant method is used in the trial and error solution of the dense phase flow rate F_2.

5.2.5 Model Verification

Results given by the program DRUM3 were checked with results from the program DRUM1 for long inclined drums. Similar results were obtained. Variable holdup along a flightless drum was checked by setting flight dimensions to be very small.

The program DRUM3 does not always obtain a solution. Sometimes the program has difficulty getting past the point of design loading where the change in holdup for change in initial discharge angle $dH/d\theta_i$ approaches zero. Depending on the integration step size, this sometimes caused a division by zero in the solution of Equation 136.

5.2.6 Kamke's Experiments (1984)

The computer program DRUM3 was tested using the rotary wood chip dryer analyzed by Kamke (1984). The industrial-size dryer, illustrated in Figure 5.27, was 5.5 metres long by 1.2 metres diameter. For a portion of its length, the dryer had six interior flights in addition to the twelve conventional exterior flights. The interior flights were single sided and the exterior flights were two-sided as shown in Figure 5.28. Wood chips and air flowed cocurrently through the horizontal dryer. Properties of the wood

chips are given in Table 5.5. The residence time was determined using radioactive tracers which were wood chips of a narrower size distribution range than the regular feed.

Table 5.5 Properties of wood chips used in Kamke's dryer

bulk density	200 kg/m^3	
particle density	450 kg/m^3	
angle of repose	82.6 degrees	
particle size distribution		
microns		wt %
>5140		3.0
3350-5140		16.2
1880-3350		24.7
1530-1880		17.5
1200-1530		11.4
800-1200		8.5
<800		18.7
sphericity		0.75

Figure 5.29 compares the predicted residence times for individual particle size groups to experimental results reported by Kamke. Overall the predicted residence times are larger than actual. The predicted variation in residence time due to particle size is much larger than observed by Kamke.

5.2.7 Discussion of Results

The reason that the model predicted such a large difference in residence times among the particle size groups is that the model predicts that almost all the flow occurs

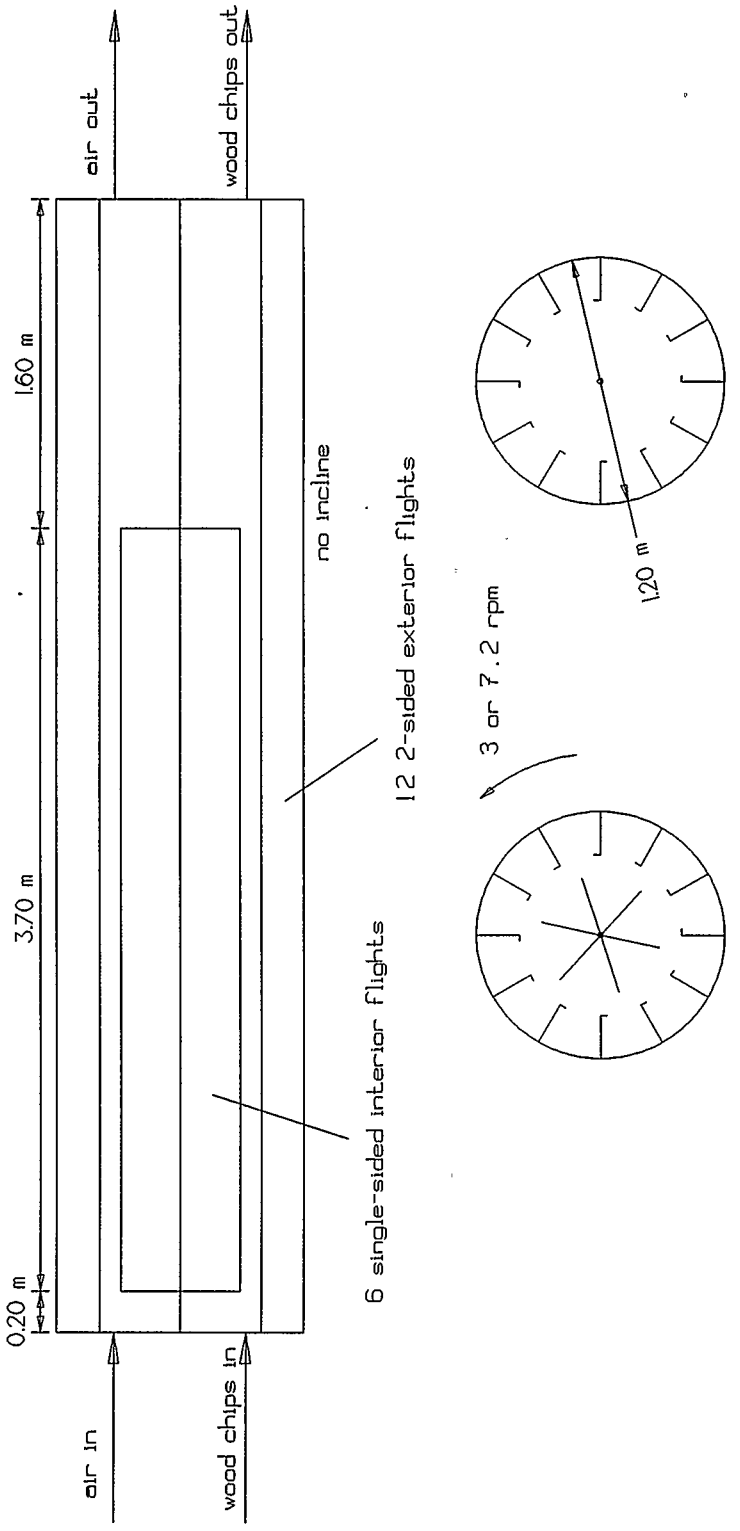


Figure 5.27 Kamke's wood chip dryer with 12 exterior flights and 6 interior flights

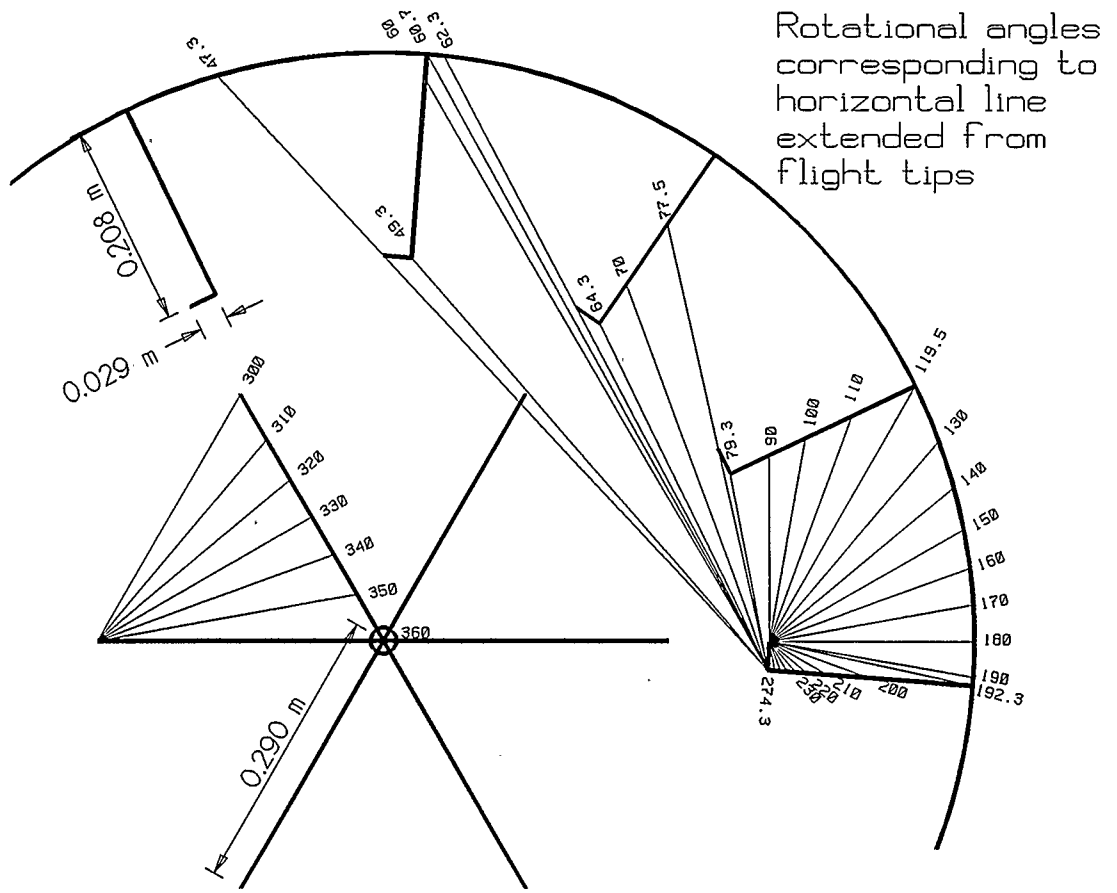


Figure 5.28 Details of lifting flights in Kamke's wood chip dryer

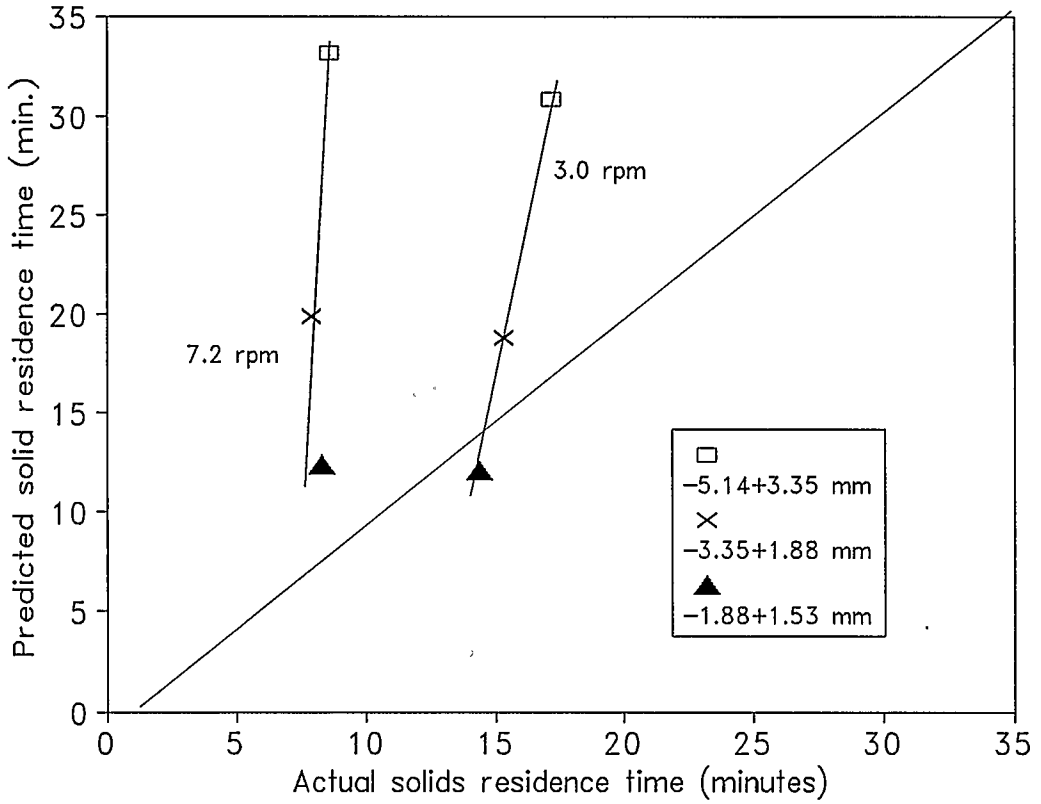


Figure 5.29 Comparison of predicted and reported residence times for Kamke's rotary wood chip dryer

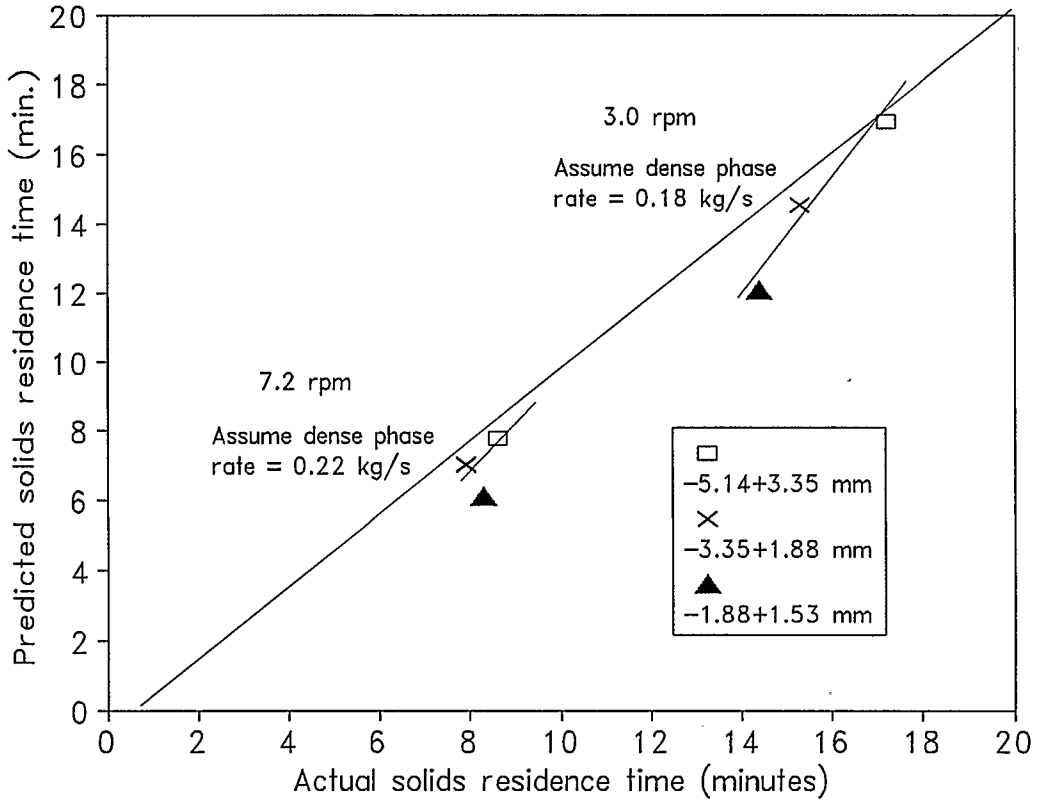


Figure 5.30 Comparison of predicted and reported residence times for Kamke's rotary wood chip dryer if the dense phase flow rate is increased

in the airborne phase. Flow in the dense phase is very low due to the very large angle of repose (82.6 degrees) for the wood chips. Kamke comments on the fact that the wood chips are not a free-flowing material. The particle surface would rotate past vertical and then slump. The new surface would be much lower than the reported angle of repose.

It is conceivable that the angle of repose of the dense phase in the lower half of the drum was much lower than 82.6 degrees. If this is true, then the dense phase flow rate would be much larger than what is predicted by the model.

Figure 5.30 shows the results obtained if a higher dense phase rate is set for the model. Not only is the overall predicted residence time closer to the experimental results, but the variation of residence time with particle size also agrees. Dense phase flows of 0.18 kg/s and 0.22 kg/s were found to give the best match for the tests at 7.2 rpm and 3.0 rpm respectively. This is about two-thirds of the feed rate.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

The model which was developed includes a number of new approaches and techniques to determine the residence time of particles in flighted rotating drums. New in this model are

i) the way of defining the shape of the flights and the way in which the discharge rate and the holdup are calculated,

ii) the method used to determine the flow rate of the dense phase,

iii) the method used to allow for the shielding effect of particles falling in sheets,

vi) consideration of a distribution of particle sizes in the feed and a method to solve for the distribution of residence times,

v) and allowing for the holdup to change with length and a method to determine the holdup when it is not constant.

The model uses a different approach than previous models to define the geometry of the flights. Instead of using equations to describe the flight volumetric holdup relation to the rotation angle, the model requires as input a table of rotation angles and dense phase surface lengths. Instead of calculating flight holdup directly and differentiating to get the flight discharge rate, the model calculates discharge rates directly and integrates to get flight holdup. This alternate method of stating the flight geometry makes it more convenient to model any type of lifting flight. The method also accounts for interference from other flights, a condition which was not considered in previous models.

Previous models either ignored dense phase flow or treated it with empirical

equations fitted to data from a small test drum. The empirical methods were difficult to apply with confidence to drum configurations other than for the drum from which the original data was obtained. The new method which considers the length of the rolling surfaces of the dense phase can be applied to any drum. To reduce the complexity of the model, the surface length on non-discharging flights was assumed to be zero when more than one flight spacing from the position of initial discharge and equal to the length at initial discharge within one flight spacing. Application of the method to the available published data showed that its predictive performance depends on the drum and the loading condition. For underloaded drums with EAD and CBD flights the method gave flow rates that were too low. For overloaded drums, the method gave flow rates that were too high. A better method of approximating the surface length c_{ex} of material on the non-discharging flights would improve the performance of the model. The surface length should be a function of the holdup on the flight as well as the location of the flight. A more accurate but still not too complex method of approximating c_{ex} was not obvious.

Previous workers had commented that there appeared to be a shielding effect caused by the particles falling in sheets. For this report, experiments which involved discharging a single sheet of particle with a wide particle size distribution into a wind tunnel were conducted. The displacements of the fallen particles by the gas stream were studied. The results could best be described by proposing a two velocity system; a lower than average velocity within the sheet of particles and a higher than average velocity outside the sheet. From the wind tunnel experiments, the apparent velocity within the

sheets for a variety of condition was determined. An empirical equation which related the apparent velocity within the sheet to the average velocity, the flight discharge rate, the fall height and the flight length was derived. When the empirical equation was incorporated into the residence time model, the effect of the gas rate on the residence time more closely fit the experimental data from small drums. However, due to the relatively poor fit to experimental data from the wind tunnel, caution should be used in applying the empirical correction. Even greater caution should be used if the empirical correction is being used for parameters outside the experimental range or for other feed materials. Future studies should take an approach similar to Langrish (1989) and relate the lower-than-average gas velocity within the sheets to the flight discharge rate and the fall height by first principles.

This is the first model for rotary drums to consider a feed with a distribution of particle sizes. Previous models always used a mean particle diameter. The equations and an algorithm to solve for residence times for individual particle size groupings of the feed was developed. It was shown that Kamke's observation (1984) that all particles appeared to behave as though they were of a single particle size can be explained by the reduced effect of the gas rate on the falling sheets and that over half the axial movement occurred in the dense phase where velocities are not effected by particle size.

This is also the first model for a flighted drum to allow for the holdup to vary with the length of the drum. The change in holdup with length must be considered when modelling a case in which a significant portion of the total flow occurs in the non-discharging flights. A method of approximating the change in holdup with length was

incorporated into the model and the model was used to simulate a horizontal drum. Not enough data are available to assess the model's ability to accurately simulate drums with holdup which is not constant over its length.

Overall, the model is more general than previous models. It is able to simulate a variety of drum configurations and operation modes. For example, the particle and gas flows may be either cocurrent or countercurrent. The drum may have an open centre or a centrefill with flights. Any flight profile may be used. The drum may be inclined or horizontal. The drum may be overloaded, design loaded or underloaded. The model's robustness was demonstrated by modelling a variety of drums under numerous conditions for which published data was available.

Recommendations for future improvements include a better method of determining or approximating the length of the dense phase surfaces on the non-discharging flights and replacing the empirical correction to the gas velocity by a method based on first principle which relates the gas velocity within the sheets to the flight discharge rate and the fall height.

REFERENCES

ABB Raymond, "Bartlett-Snow™ Rotary Thermal Processing Equipment and Systems", Bulletin No. 873R (1990).

Baker, C.G.J., "Cascading Rotary Dryers", in Advances in Drying, Vol. 2, A.S. Mujumdar (ed), Hemisphere, New York (1983), pp.1-51.

Baker, C.G.J., "The Design of Flights in Cascading Rotary Dryers," Drying Tech. **6**(4), 631-653 (1988).

De, D.S. and S.G. Mukherjee, "The Effect of Circular Dam on Hold-up in a Rotary Dryer," Ind. Chem. Engr. **17**(2), 3-7 (1975).

Friedman, S.J. and W.R. Marshall, "Studies in Rotary Drying Part 1 - Holdup and Dusting," Chem. Eng. Prog. **45**(8), 482-493 (1949).

Glikin, P.G., "Transport of Solids Through Flighted Rotating Drums," Trans. Inst. Chem. Engrs. **56**, 120-126 (1978).

Hallström, A., "Rotary Drying of Fertilizers II Observations at an Industrial Plant", J. Sep. Proc. Tech. **6**, 54-57 (1985).

Hogg, R., K. Shoji and L.G. Austin, "Axial Transport of Dry Powders in Horizontal Rotating Cylinders", Powder Tech. **9**, 99-106 (1974).

Kamke, F.A., Ph.D. Thesis "Engineering Analysis of a Rotary Dryer: Drying of Wood Particles", Oregon State Univ., 1984.

Kamke, F.A. and J.B. Wilson, "Computer Simulation of a Rotary Dryer - Part I: Retention Time", AIChE J., **32**(2), 263-268 (1986).

Kelly, J.J., Ph.D. Thesis "Analysis of the Design and Operating Procedures in Rotary Driers and Coolers", Univ. College Dublin, 1969.

Kelly, J.J., "Rotary Drying", in Handbook of Industrial Drying, A.S. Mujumdar (ed.), Marcel Dekker, New York (1987), pp.133-154.

Kelly, J.J. and J.P. O'Donnell, "Dynamics of Granular Material in Rotary Driers and Coolers," Inst. Chem. Engrs. Sym. Ser. No. 29, 34-44 (1968).

Kelly, J.J. and P.C. O'Donnell, "Residence Time Model for Rotary Drums", Trans. Inst. Chem. Engrs. **55**, 243-252 (1977).

Kennedy Van Saun Corporation, "Rotary Dryer Systems", Bulletin DRY 1/82 (2), (1982).

Kisakuek, B., "Retention Time in a Rotary Dryer", Proceedings of the 3rd Int. Drying Symp., Vol. 2, Univ. of Birmingham, Birmingham, England, J.C. Ashworth (ed.), Drying Reasearch Limited, Wolverhampton, (1982).

Kramer, H. and P. Croockewit, "The Passage of Granular Solids Through Inclined Rotary Kilns", Chem. Eng. Sci., **1**(6), 259-256 (1952).

Langrish, T.A.G., Ph.D. Thesis "The Mathematical Modelling of Cascading Rotary Dryers", Univ. of Oxford, 1989.

Matchett, A.J. and C.G.J. Baker, "Particle Residence Times in Cascading Rotary Dryers Part 1 - Derivation of the Two-Stream Model", J. Sep. Proc. Tech. **8**, 11-17 (1987).

Matchett, A.J. and M.S. Sheikh, "An Improved Model of Particle Motion in Cascading Rotary Dryers", Trans. Inst. Chem. Engrs. **68A**, 139-148 (1990).

Militzer, K.E., "The Rotary Dryer", Chem. Biochem. Eng. Q. **4**(1), 31-38 (1990).

O'Donnell, P.C., Ph.D. Thesis, "Heat Transfer Studies in Rotary Coolers", Univ. College Dublin, 1975.

Platin, B.E., A. Erden and O.L. Gulder, "Modelling and Design of Rotary Dryers", Proceedings of the 3rd International Drying Symposium, Vol. 2, Univ. of Birmingham, Birmingham, England, J.C. Ashworth (ed.), Drying Research Limited, Wolverhampton, (1982).

Prutton, C.F., C.O. Miller and W.H. Schuette, "Factors Influencing the Performance of Rotary Dryers", Trans. AIChE **38**, 123-141 (1942).

Purcell, J.G., "Practical Rotary Cascading Dryer Design", Chem. Engr. **346**, 496-497, 500 (1979).

Reay, D., "Theory in the Design of Dryers", Chem. Engr. **346**, 501-503, 506 (1979).

Saeman, W.C., "Passage of Solids through Rotary Kilns. Factors Affecting Time Passage", Chem. Eng. Prog. **47**(10), 508-514 (1951).

Saeman, W.C. and T.R. Mitchell, Jr., "Analysis of Rotary Dryers and Cooler Performance", Chem. Eng. Prog. **50**(9), 467-475 (1954).

Schiller, L. and A. Naumann, "Uber die grundlegenden Berechnungen bei der Schwerkraftaufbereitung", Verein Deutscher Ingenieure Zeitschrift **77** (12), 318-320 (1933).

Schofield, F.R. and P.G. Glikin, "Rotary Driers and Coolers for Granular Fertilizers", Trans. Inst. Chem. Engrs. **40**, 183-190 (1962).

Sharples, K., P.G. Glikin and R. Warne, "Computer Simulation of Rotary Driers", Trans. Inst. Chem. Engrs. **42**, T275-T284 (1964).

Sheikh, M.S., Ph.D. Thesis, "Prediction of Particle Residence Times in Cascading Rotary Dryers", Department of Chem. Eng., Teesside Polytechnic, Middlesbrough, UK, 1987.

Smith, B.A., "Written Discussion on Factors Influencing the Performance of Rotary Dryers", Trans. AIChE **38**, 251-257 (1942).

Thorne, B., Ph.D. Thesis, "The Computer Simulation of the Rotary Dryer", Chem. Eng. Department, Univ. College Dublin, 1979.

Thorne, B. and J.J. Kelly, "A Mathematical Model for the Rotary Drum", Proceedings of the 1st International Symposium on Drying, Montreal, 160-169 (1978).

Tscheng, S.H., Ph.D. Thesis, "Convective Heat Transfer in a Rotary Kiln", Univ. of British Columbia, 1978.

Vahl, L. and W.G. Kinga, "Transport of Solids Through Horizontal Rotary Cylinders", Chem. Eng. Sci. **1**(6), 253-258 (1952).

van Brakel, J., "The Choice and Design of Dryers - a personal view", Chem. Engr. **346**, 493-495 (1979).

APPENDICES

APPENDIX A. WIND TUNNEL TESTS

Properties of the particles used in the wind tunnel experiments are given in Table A.1 and a summary of the test conditions and the resulting apparent gas velocities inside the sheets are given in Table A.2.

Table A.1 Properties of test material

Material	sand	dried shale	shale ash
Free moisture, wt%	0.12	4.18	0.60
Particle density, kg/m3	2650	1200	1200
Bulk density, kg/m3	1600	860	860
Sphericity	1.0 (assumed)	1.0 (assumed)	1.0 (assumed)
Particle size distribution, microns	wt% in range	wt% in range	wt% in range
>2380	4.80	50.50	17.95
842-2380	12.70	24.55	19.35
297-842	33.55	12.00	30.15
149-297	32.70	5.20	17.45
105-149	3.50	1.05	6.80
74-105	8.50	2.45	4.25
<74	4.15	4.15	3.90

Table A.2 Summary of test conditions and results

Test No.	Feed Type	Air Velocity (m/s)	Spill Rate (kg/m s)	Flight Length (m)	Flight Radius (m)	Fall Height (m)	Mean Advance (m)	V_c (m/s)	V_c/V_t
11	Raw shale	5.2	10.4	2.0	0.510	10	2.963	5.86	1.127
12	Raw shale	3.9	10.9	2.0	0.510	10	2.018	4.55	1.167
13	Raw shale	2.6	10.3	2.0	0.510	10	1.577	3.02	1.162
14	Raw shale	1.8	10.8	2.0	0.510	10	0.624	1.57	0.872
15	Raw shale	4.3	12.9	2.0	0.510	10	2.511	4.91	1.142
16	Raw shale	2.8	14.6	2.0	0.510	10	1.499	3.41	1.218
17	Raw shale	4.8	14.0	1.0	0.510	10	2.905	5.64	1.175
18	Raw shale	3.8	14.7	1.0	0.510	10	2.320	5.28	1.389
19	Raw shale	3.6	15.0	1.0	0.510	10	2.830	5.98	1.661
20	Raw shale	2.2	15.2	1.0	0.510	10	1.583	3.39	1.541
21	Dry sand	5.6	22.9	1.0	0.510	10	3.788	4.01	0.716
22	Dry sand	4.0	24.0	1.0	0.510	10	1.027	1.41	0.353
23	Dry sand	2.0	23.1	1.0	0.510	10	0.516	0.64	0.320
24	Dry sand	5.3	22.0	1.0	0.510	10	1.774	2.68	0.506
25	Dry sand	3.9	21.9	1.0	0.510	10	1.406	2.38	0.610
26	Dry sand	2.0	21.3	1.0	0.510	10	0.435	0.66	0.330
27	Dry sand	5.4	15.2	2.0	0.355	10	3.253	3.97	0.735
28	Dry sand	3.0	15.3	2.0	0.355	10	1.891	1.97	0.657
30	Dry sand	4.7	15.9	2.0	0.355	10	2.913	3.55	0.755
31	Dry sand	3.7	34.3	2.0	0.355	10	0.779	1.19	0.322
32	Dry sand	2.0	35.5	2.0	0.355	10	0.307	0.51	0.255
33	Dry sand	5.0	38.8	2.0	0.355	10	1.100	1.61	0.322
34	Dry sand	6.0	12.5	2.0	0.355	10	3.268	4.63	0.772
35	Spent shale	4.2	7.4	2.0	0.355	10	3.865	4.85	1.155
36	Spent shale	5.7	7.7	2.0	0.355	10	4.989	5.48	0.961
37	Spent shale	2.7	7.4	2.0	0.355	10	2.040	2.56	0.948
38	Spent shale	4.5	11.1	2.0	0.355	10	3.168	4.04	0.898
39	Spent shale	6.1	10.7	2.0	0.355	10	4.015	4.51	0.739
40	Spent shale	2.3	9.1	2.0	0.355	10	1.590	1.98	0.861
41	Dry sand	4.3	10.3	2.0	0.355	10	2.583	3.16	0.735
42	Dry sand	2.3	11.0	2.0	0.355	10	0.690	0.93	0.404
43	Dry sand	6.4	17.7	2.0	0.355	10	3.424	3.87	0.605
44	Dry sand	4.4	18.7	2.0	0.355	10	2.079	2.60	0.591
45	Dry sand	2.7	18.2	2.0	0.355	10	0.929	1.38	0.511
46	Raw shale	4.3	10.8	2.0	0.355	10	1.850	3.66	0.851
47	Raw shale	6.3	10.4	2.0	0.355	10	3.115	5.57	0.884
48	Raw shale	2.4	10.5	2.0	0.355	10	0.724	2.58	1.075
49	Spent shale	4.3	19.2	1.0	0.510	10	1.695	2.51	0.584
50	Spent shale	6.3	24.5	1.0	0.510	10	3.321	4.10	0.651

51	Spent shale	2.4	26.4	1.0	0.510	10	1.301	1.92	0.800
52	Dry sand	4.3	23.7	1.0	0.510	10	1.708	2.29	0.533
53	Dry sand	2.4	23.9	1.0	0.510	10	0.712	1.09	0.454
54	Dry sand	6.3	24.1	1.0	0.510	10	2.816	3.39	0.538
55	Dry sand	5.8	25.8	1.0	0.510	7	1.352	2.17	0.374
56	Dry sand	4.4	25.6	1.0	0.510	7	0.714	1.38	0.314
57	Dry sand	1.9	24.8	1.0	0.510	7	0.284	0.50	0.263
58	Dry sand	1.9	36.8	1.0	0.510	7	0.285	0.70	0.368
59	Dry sand	4.1	41.0	1.0	0.510	7	0.457	0.87	0.212
60	Dry sand	6.0	38.1	1.0	0.510	7	0.751	1.49	0.248
61	Spent shale	2.4	26.4	1.0	0.510	7	0.609	1.02	0.425
62	Spent shale	4.6	29.8	1.0	0.510	7	1.434	2.57	0.559
63	Spent shale	5.9	25.9	1.0	0.510	7	1.590	2.91	0.493
64	Spent shale	1.7	15.7	1.0	0.510	7	0.936	1.00	0.588
65	Spent shale	3.8	14.4	1.0	0.510	7	1.708	2.64	0.695
66	Spent shale	7.4	14.5	1.0	0.510	7	3.338	5.45	0.736
67	Raw shale	1.4	15.9	1.0	0.510	7	0.273	1.01	0.721
68	Raw shale	3.8	15.5	1.0	0.510	7	0.642	3.33	0.876
69	Raw shale	7.6	15.5	1.0	0.510	7	1.873	4.78	0.629
70	Raw shale	1.9	22.1	1.0	0.510	7	0.359	1.37	0.721
71	Raw shale	3.7	20.7	1.0	0.510	7	0.719	3.37	0.911
72	Raw shale	7.5	22.0	1.0	0.510	7	1.342	3.28	0.437
73	Dry sand	6.0	28.3	2.0	0.510	7	0.893		
74	Dry sand	3.8	29.9	2.0	0.510	7	0.785		
75	Spent shale	6.7	22.8	1.0	0.510	7	2.496	3.91	0.584
76	Spent shale	4.6	25.0	1.0	0.510	7	1.947	3.12	0.678
77	Spent shale	2.2	19.8	1.0	0.510	7	0.536	0.97	0.441
78	Spent shale	7.0	12.8	1.0	0.510	7	2.925	4.54	0.649
79	Spent shale	4.9	14.8	1.0	0.510	7	1.709	2.93	0.598
80	Spent shale	2.2	15.3	1.0	0.510	7	0.664	1.29	0.586
91	Spent shale	8.6	19.6	1.0	0.510	4	0.938	3.11	0.362
92	Spent shale	4.2	20.2	1.0	0.510	4	0.357	1.14	0.271
93	Spent shale	2.2	20.9	1.0	0.510	4	0.238	0.85	0.386
94	Spent shale	8.2	11.4	1.0	0.510	4	1.171	3.23	0.394
95	Spent shale	3.9	12.2	1.0	0.510	4	0.503	1.47	0.377
96	Spent shale	1.9	14.0	1.0	0.510	4	0.187	0.49	0.258
97	Raw shale	7.6	21.9	1.0	0.510	4	0.470		
98	Raw shale	3.7	22.4	1.0	0.510	4	0.247		
99	Raw shale	1.8	22.5	1.0	0.510	4	0.143		
100	Raw shale	8.4	13.6	1.0	0.510	4	0.645		
101	Raw shale	3.3	13.1	1.0	0.510	4	0.432		
102	Raw shale	2.0	13.1	1.0	0.510	4	0.285		
103	Dry sand	8.6	26.4	1.0	0.510	4	0.706		
104	Dry sand	4.3	28.3	1.0	0.510	4	0.431		
105	Dry sand	1.7	32.8	1.0	0.510	4	0.293		

106	Dry sand	7.3	19.0	1.0	0.510	4	0.723		
107	Dry sand	5.1	19.2	1.0	0.510	4	0.502		
109	Dry sand	7.9	14.5	2.0	0.355	4	1.089		
110	Dry sand	4.2	14.7	2.0	0.355	4	0.446		
111	Dry sand	2.2	14.7	2.0	0.355	4	0.147		
112	Dry sand	8.2	9.3	2.0	0.355	4	1.353		
113	Dry sand	4.5	10.1	2.0	0.355	4	0.724		
114	Dry sand	2.5	10.1	2.0	0.355	4	0.341		
115	Raw shale	7.3	11.3	2.0	0.355	4	0.835		
116	Raw shale	4.5	10.5	2.0	0.355	4	0.498		
117	Raw shale	2.1	10.6	2.0	0.355	4	0.329		
118	Raw shale	8.1	8.8	2.0	0.355	4	1.753		
123	Spent shale	2.1	13.4	2.0	0.355	4	0.284	0.66	0.314
124	Spent shale	6.2	6.4	2.0	0.355	4	1.692	4.35	0.702
125	Spent shale	4.1	8.4	2.0	0.355	4	0.793	2.23	0.544
126	Spent shale	2.0	8.6	2.0	0.355	4	0.386	0.87	0.435

APPENDIX B. COMPUTER PROGRAM

A summary of the files and routines which are combined to form the residence time models is given in Table B.1. A complete listing of the program files is given in the pages following the table.

Table B.1 Routine summary

SOURCE FILE	FUNCTION	LEVEL	PURPOSE
DRUM .H		header	Declares functions; defines input data structures.
INTERP .H		header	Header file for linear interpolation routine.
RKGS .H		header	Header file for Runge Kutta integration routine.
ROMBERG .H		header	Header file for Romberg integration routine.
DRUM1 .C	main()	high	Loads inputs. Determines holdup and feed rate over a range of initial discharge angles for long inclined drums only.
DRUM2 .C	main()	high	Loads inputs. Determines holdup, airborne phase and dense phase flow rates and length differentials over a range of initial discharge angles for any drum.
DRUM3 .C	main()	high	Loads inputs. Sets drum end conditions. Calls integration routine to determine residence time of drum allowing for holdup to vary with length.
	diffdrum()	high	Computes length differentials.
	drumoutp()	high	Prints intermediate results.
INITANGL .C	initinangle()	middle	Computes initial interior flight discharge angle.
	fct_ex_l2()	low	Integrand for exterior flight holdup.
	fct_in_l2()	low	Integrand for interior flight holdup.

LONGRATE .C	longrate()	middle	Determines dense phase flow rate in a long inclined drum.
	fct_ex_l3()	low	Integrand for dense phase flow rate of exterior flights.
	fct_in_l3()	low	Integrand for dense phase flow rate of interior flights.
FALLRATE .C	fallrate()	middle	Determine airborne phase flow rate and PSD of holdup given the dense phase rate.
FLOWRATE .C	flowrates()	middle	Determines the airborne and dense phase rates and PSD of holdup for any drum.
	fct_ex_l2x()	low	Integrand for advance of particles from exterior flight.
	fct_in_l2x()	low	Integrand for advance of particles from interior flight.
FALLMASS .C	fallmass()	middle	Computes airborne particle mass.
	fct_ex_l2t()	low	Integrand for falling mass from exterior flight.
	fct_in_l2t()	low	Integrand for falling mass from interior flight.
HOLDUPD .C			Computes dense phase holdup of discharging flights.
	Holdup_DEF()	middle	Discharging exterior flights.
	fct_HDEF()	low	Differential equations for average holdup in discharging exterior flight.
	Holdup_DIF()	middle	Disharging interior flight.
	fct_HDIF()	low	Differential equations for average holdup in discharging interior flight.
	out_H()	low	Dumb Runge Kutta output.
HOLDUPND .C			Computes dense phase holdup of non-discharging flights.
	Holdup_NDEF()	middle	Non-discharging exterior flights.
	fct_HNDEF()	low	Differential equations for average holdup in non-discharging exterior flight.

	ex_to_ex_12()	low	Integrand for exterior flight receiving particles from another exterior flight.
	in_to_ex_12()	low	Integrand for exterior flight receiving particles from an interior flight.
	Holdup_NDIF()	middle	Non-discharging interior flights.
	fct_HNDIF()	low	Differential equations for average holdup in non-discharging interior flights.
	ex_to_in_12()	low	Integrand for interior flight receiving material from an exterior flight.
ROMBERG .C	romberg()	low	Romberg integration.
INTERP .C	interp()	low	Linear interpolation.
RKGS .C	rkgs()	low	Runge Kutta integration.
ROUTINES .C	annulusvel()	low	Computes average gas velocity.
	airdens()	low	Computes air density.
	airvisc()	low	Computes air viscosity.
	falldistance()	low	Computes vertical fall distance.
	falltime()	low	Computes vertical fall time.
	facelength()	low	Computes particle face length.
	Re()	low	Computes particle Reynold's number.
	CD()	low	Computes drag coefficient.
	spillrate()	low	Computes flight discharge rate.
	sheetgasvel()	low	Computes gas velocity in sheet.
ADVANCE .C	falladvance()	low	Fallen particle displacement.
DH2DZ .C	dH2dz()	middle	Computes rate of change in dense phase holdup w.r.t. drum length.
DHDANGLE .C	dH2dinitexangle()	middle	Computes rate of change in dense phase holdup w.r.t. the initial discharge angle of the exterior flights.

```
/* HEADER FILE: DRUM.H */
/*****************************************************************************/
/* drum.h: solids movement in a flighted rotating drum header file. #include */
/*   this file near the beginning of source files containing functions that  */
/*   will call any of the functions listed below or refers to data in any of */
/*   the structures listed below.                                            */
/*****************************************************************************/

/*****************************************************************************/
/* Solids Movement in a Rotating Flighted Drum Function Summary:             */
/*                                                                           */
/*     airdens()        Calculates the density of air.                       */
/*     airvisc()        Calculates the viscosity of air.                     */
/*     annulusvel()     Calculates the velocity through an annular passage.  */
/*     CD()             Calculates particle drag coefficient.                */
/*     defaults()       Sets default input data.                             */
/*     dH2dz()          Change in dense phase holdup w.r.t. drum length.     */
/*     dH2dinitexangle()Change in holdup w.r.t. init. ext. disch. angle.     */
/*     facelength()     Calculates surface length of material on flight.     */
/*     falladvance()    Calculates distance advanced by fallen particle.     */
/*     falldistancein() Calculates fall distance from interior flight.       */
/*     falldistanceex() Calculates fall distance from exterior flight.       */
/*     fallmass()       Calculates mass in a state of fall.                  */
/*     fallrate()       Calculates airborne phase rate and PSD.              */
/*     falltime()       Calculates time of fall.                             */
/*     fct_in_l2()      Integrand for interior flight holdup.                */
/*     fct_ex_l2()      Integrand for exterior flight holdup.                */
/*     flowrates()      Calculates airborne and dense phase rates and PSD.   */
/*     HoldupDEF()      Calculates holdup of discharging exterior flights.   */
/*     HoldupDIF()      Calculates holdup of discharging interior flights.   */
/*     HoldupNDEF()     Calculates holdup of non-discharging exterior flights*/
/*     HoldupNDIF()     Calculates holdup of non-discharging interior flights*/
/*     initinangle()     Calculates initial discharge angle of interior flight*/
/*     longrate()        Calculates dense phase rate in long inclined drum.  */
/*     out_H()          Runge Kutta output function.                         */
/*     Re()             Calculates Reynolds number.                          */
/*     sheetgasvel()    Calculates reduced gas velocity within falling sheets*/
/*     spillrate()      Calculates flight discharge rate.                    */
/*                                                                           */
/*****************************************************************************/

/* Declare function return and argument types.                               */
double airdens(double);
double airvisc(double);
double annulusvel(double, double, double, double, double);
double CD(double);
void defaults(void);
double dH2dz(double, double, double);
double dH2dinitexangle(double, double);
double facelength(double, struct flightprofile *, short);
double falladvance(double, double, double);
double falldistancein(double, double, double);
double falldistanceex(double, double, double, double, double);
double fallmass(double, double);
double fallrate(double, double, double, double, double[]);
double falltime(double);
double fct_in_l2(double);
double fct_ex_l2(double);
void flowrates(double, double, double, double *, double *, double[]);
double HoldupDEF(double);
double HoldupDIF(double);
double HoldupNDEF(double, double);
double HoldupNDIF(double, double);
double initinangle(double);
double longrate(double, double);
void out_H(double, double[], double[], short, short, short *, double[]);
double Re(double, double, double, double);
double sheetgasvel(double, double, double, double);
double spillrate(double, double, double);

#define MAXNAMELENGTH              80
#define PI                         3.1415927
#define GRAVITY                    9.81
```

```
#define MAX_NO_ANGLE_INTERVALS      180
#define MAX_NO_PARTICLE_SIZES       10
#define YES                         1
#define NO                          0

struct DRUM_DEFINITION
    {
    char name[MAXNAMELENGTH];                       /* Drum identification */
    short centerfill;                 /* if centerfill exits then 1; else 0 */
    double Do;                              /* Exterior shell diameter, m */
    double Di;                              /* Interior shell diameter, m */
    double Length;                                  /* Drum length, m */
    double incline;                  /* Drum incline to horizontal, radians */
    struct flightdefinition
        {
        short N;                                    /* Number of flights */
        double D;                          /* Flight tip lotus diameter, m */
        double length;                             /* Flight axial length, m */
        short np;                        /* Number of points in flight profile */
        struct flightprofile
            {
            double angle;                  /* Position of flight tip, radians */
            double face;     /* Particle surface ratio on discharging flight */
            }profile[MAX_NO_ANGLE_INTERVALS];
        }exflight;                                  /* Exterior flights */
        struct flightdefinition
         inflight;                                  /* Interior flights */
    struct dischargedefinition
        {
        double init_ex_angle;/* Initial discharge angle of exterior flight   */
        }discharge;          /*     at the drum discharge, radians.          */
    }drum;

struct PARTICLE_PROPERTIES
    {
    char name[MAXNAMELENGTH];
    double part_dens;
    double bulk_dens;
    double repose_angle;
    double spheric;
    int no_of_sizes;
    struct sizedistribution
                {
                double Dp;
                double mf;
                } size[MAX_NO_PARTICLE_SIZES];
    }particle;

struct GAS_PROPERTIES
    {
    char name[MAXNAMELENGTH];
    double dens;
    double visc;
    }gas;

struct OPERATING_PARAMETERS
    {
    char name[MAXNAMELENGTH];
    double solids_rate;
    double gas_rate;
    double temperature;
    double rps;
    }operate;

struct CONVERGENCE_TOLERANCE
    {
    char name[MAXNAMELENGTH];
    double reltol;
    }convergence;

struct INTEG_VARIABLES
    {
    double increment;
```

```
double bound;
}integ;
```

```
/* HEADER FILE: INTERP.H */
/*************************************************************************/
/* interp.h: header file for linear interpolation routine.  Include this */
/*    file near the beginning of source files that contain functions     */
/*    that call the linear interpolation routine with the following      */
/*    statement:                                                         */
/*                                                                       */
/*    #include "interp.h"                                                */
/*************************************************************************/

struct XYTABLE
   {
   double x;
   double y;
   };

double interp(double, struct XYTABLE *, short);
```

```
/* HEADER FILE: RKGS.H */
/*************************************************************************/
/* rkgs.h: header file for Runge Kutta integration routine.  Include     */
/*    this file near the beginning of source files that containing       */
/*    functions that call the Runge Kutta integration routine using the  */
/*    following statement:                                               */
/*                                                                       */
/*    #include "rkgs.h"                                                  */
/*************************************************************************/

#define NOTENOUGHMEMORY     14
#define WRONGSIGN            13
#define ZEROSTEP             12
#define TOOMANYBISECTIONS   11
#define MAXBISECTIONS        10
#define PROCEED               0


short rkgs(double, double, double, double, double[], double[], short,
        short *, void (*)(), void(*)(), double[]);
```

```
/* HEADER FILE: ROMBERG.H */
/*************************************************************************/
/* romberg.h: header file for romberg integration routine. Include this  */
/*    file near the beginning of source files containing functions that  */
/*    call the romberg integration routine with the following statement: */
/*                                                                       */
/*    #include "romberg.h"                                               */
/*************************************************************************/

double romberg (double a, double b, double eps, double(*f)());
```

```
/* SOURCE FILE: DRUM1.C */
/*****************************************************************************/
/* This routine executes the program for determining the holdup of a         */
/* flighted rotating drum. The routine loads the required data both from     */
/* the command line and from files listed on the command line.     The routine  */
/* extecutes the drum holdup routine and prints the result to the display.   */
/*****************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "drum.h"

void drumres(double *);

void main(argc, argv)
   int argc;
   char *argv[];
{
   char drumfile[13];
   char particlefile[13];
   char mathfile[13];
   FILE *fp;
   short i;
    double init_ex_angle, init_in_angle, fall_rate, bed_rate;
    double particle_mf[MAX_NO_PARTICLE_SIZES];
    double H1, H2, H2d, H2nd, dH2_dz, dH2_dinit_ex_angle;
    double gas_vel;

   if (argc < 8)
   {
      printf ("Not enough command line arguments.\n");
      exit(0);
   }

   strcpy( drumfile, *(argv+1));

   if ((fp = fopen(drumfile, "r")) == NULL )
   {
      printf ("Can't open file %s.\n", drumfile);
      exit(0);
   }

   if (fp != NULL)
   {
      rewind(fp);
      fscanf(fp, "%[^\n]\n", drum.name);
      fscanf(fp, "%d %lf %lf %lf\n",
             &drum.centerfill, &drum.Do, &drum.Di, &drum.Length);
      fscanf(fp, "%d %lf %lf %d\n",
             &drum.exflight.N, &drum.exflight.D,
             &drum.exflight.length, &drum.exflight.np);
      for (i=0; i<drum.exflight.np; i++)
      {
         fscanf(fp, "%lf %lf\n", &drum.exflight.profile[i].angle,
                                 &drum.exflight.profile[i].face);
      }
      fscanf(fp, "%lf\n", &drum.discharge.init_ex_angle);

      if (drum.centerfill)
      {
          fscanf(fp, "%d %lf %lf %d\n",
              &drum.inflight.N, &drum.inflight.D,
              &drum.inflight.length, &drum.inflight.np);
          for (i=0; i<drum.inflight.np; i++)
          {
             fscanf(fp, "%lf %lf\n", &drum.inflight.profile[i].angle,
                                     &drum.inflight.profile[i].face);
          }

      }
   }
   fclose(fp);
```

```
strcpy( particlefile, *(argv+2));

if ((fp = fopen(particlefile, "r")) == NULL )
{
   printf ("Can't open file %s.\n", particlefile);
   exit(0);
} .

if (fp != NULL)
{
   rewind(fp);
   fscanf(fp, "%[^\n]\n", particle.name);
   fscanf(fp, "%lf %lf %lf %lf %d\n",
          &particle.part_dens, &particle.bulk_dens,
          &particle.repose_angle, &particle.spheric,
          &particle.no_of_sizes);
   for (i=0; i<particle.no_of_sizes; i++)
   {
      fscanf(fp, "%lf %lf\n", &particle.size[i].Dp,
                              &particle.size[i].mf);
   }
}
fclose(fp);

strcpy( mathfile, *(argv+3));

if ((fp = fopen(mathfile, "r")) == NULL )
{
   printf ("Can't open file %s.\n", mathfile);
   exit(0);
}

if (fp != NULL)
{
   rewind(fp);
   fscanf(fp, "%[^\n]\n", convergence.name);
   fscanf(fp, "%lf %lf %lf\n",
          &convergence.reltol, &integ.increment,
          &integ.bound);
}
fclose(fp);

drum.incline = atof( *(argv+4));
operate.rps = atof( *(argv+5));
operate.gas_rate = atof( *(argv+6));
operate.temperature = atof( *(argv+7));

if (drum.inflight.N == 0 || drum.Di == 0.)
     drum.centerfill = NO;

 gas.dens = airdens(operate.temperature);
 gas.visc = airvisc(operate.temperature);

 bed_rate = 0.0;
 for(i =8; i<72; i++)
 {
     init_ex_angle = i * PI/36.;
     if(drum.centerfill)
         init_in_angle = initinangle(init_ex_angle);
     else
         init_in_angle = PI;
     H1 = fallmass(init_ex_angle, init_in_angle) / particle.bulk_dens;
     H2d = HoldupDEF(init_ex_angle) + HoldupDIF(init_in_angle);
     H2nd = HoldupNDEF(init_ex_angle, init_in_angle)
               + HoldupNDIF(init_ex_angle, init_in_angle);
     H2 = H1 + H2d + H2nd;
     gas_vel = annulusvel(operate.gas_rate, gas.dens, drum.Do, drum.Di, H2);
     bed_rate = longrate(init_ex_angle, init_in_angle);
     fall_rate = fallrate(init_ex_angle, init_in_angle, gas_vel,
         bed_rate, particle_mf);
     printf("%7.3f  %7.3f  %7.4f  %7.4f %7.5f %7.5f %7.5f\n",
         init_ex_angle, init_in_angle, fall_rate, bed_rate, H1, H2d, H2nd);
     }
```

```
/* SOURCE FILE: DRUM2.C */
/*****************************************************************************/
/* This routine executes the program for determining the holdup of a         */
/* flighted rotating drum. The routine loads the required data both from     */
/* the command line and from files listed on the command line.     The routine  */
/* extecutes the drum holdup routine and prints the result to the display.   */
/*****************************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "drum.h"

void drumres(double *);

void main(argc, argv)
   int argc;
   char *argv[];
{
   char drumfile[13];
   char particlefile[13];
   char mathfile[13];
   FILE *fp;
   short i;
    double init_ex_angle, init_in_angle, fall_rate, bed_rate;
    double particle_mf[MAX_NO_PARTICLE_SIZES];
    double H1, H2, H2d, H2nd, dH2_dz, dH2_dinit_ex_angle;
    double gas_vel;

   if (argc < 9)
   {
      printf ("Not enough command line arguments.\n");
      exit(0);
   }

   strcpy( drumfile, *(argv+1));

   if ((fp = fopen(drumfile, "r")) == NULL )
   {
      printf ("Can't open file %s.\n", drumfile);
      exit(0);
   }

   if (fp != NULL)
   {
      rewind(fp);
      fscanf(fp, "%[^\n]\n", drum.name);
      fscanf(fp, "%d %lf %lf %lf\n",
             &drum.centerfill, &drum.Do, &drum.Di, &drum.Length);
      fscanf(fp, "%d %lf %lf %d\n",
             &drum.exflight.N, &drum.exflight.D,
             &drum.exflight.length, &drum.exflight.np);
      for (i=0; i<drum.exflight.np; i++)
      {
         fscanf(fp, "%lf %lf\n", &drum.exflight.profile[i].angle,
                                 &drum.exflight.profile[i].face);
      }
      fscanf(fp, "%lf\n", &drum.discharge.init_ex_angle);

      if (drum.centerfill)
      {
         fscanf(fp, "%d %lf %lf %d\n",
             &drum.inflight.N, &drum.inflight.D,
             &drum.inflight.length, &drum.inflight.np);
         for (i=0; i<drum.inflight.np; i++)
         {
            fscanf(fp, "%lf %lf\n", &drum.inflight.profile[i].angle,
                                    &drum.inflight.profile[i].face);
         }

      }
   }
   fclose(fp);
```

```
strcpy( particlefile, *(argv+2));

if ((fp = fopen(particlefile, "r")) == NULL )
{
   printf ("Can't open file %s.\n", particlefile);
   exit(0);
}

if (fp != NULL)
{
   rewind(fp);
   fscanf(fp, "%[^\n]\n", particle.name);
   fscanf(fp, "%lf %lf %lf %lf %d\n",
          &particle.part_dens, &particle.bulk_dens,
          &particle.repose_angle, &particle.spheric,
          &particle.no_of_sizes);
   for (i=0; i<particle.no_of_sizes; i++)
   {
      fscanf(fp, "%lf %lf\n", &particle.size[i].Dp,
                              &particle.size[i].mf);
   }
}
fclose(fp);

strcpy( mathfile, *(argv+3));

if ((fp = fopen(mathfile, "r")) == NULL )
{
   printf ("Can't open file %s.\n", mathfile);
   exit(0);
}

if (fp != NULL)
{
   rewind(fp);
   fscanf(fp, "%[^\n]\n", convergence.name);
   fscanf(fp, "%lf %lf %lf\n",
          &convergence.reltol, &integ.increment,
          &integ.bound);
}
fclose(fp);

drum.incline = atof( *(argv+4));
operate.rps = atof( *(argv+5));
operate.gas_rate = atof( *(argv+6));
operate.solids_rate = atof( *(argv+7));
operate.temperature = atof( *(argv+8));

if (drum.inflight.N == 0 || drum.inflight.D == 0.)
     drum.centerfill = NO;

 gas.dens = airdens(operate.temperature);
 gas.visc = airvisc(operate.temperature);

 bed_rate = 0.0;
 for(i =26; i<72; i++)
 {
     init_ex_angle = i * PI/36.;
     if(drum.centerfill)
         init_in_angle = initinangle(init_ex_angle);
     else
         init_in_angle = PI;
     H1 = fallmass(init_ex_angle, init_in_angle) / particle.bulk_dens;
     H2d = HoldupDEF(init_ex_angle) + HoldupDIF(init_in_angle);
     H2nd = HoldupNDEF(init_ex_angle, init_in_angle)
              + HoldupNDIF(init_ex_angle, init_in_angle);
     H2 = H1 + H2d + H2nd;
     gas_vel = annulusvel(operate.gas_rate, gas.dens, drum.Do, drum.Di, H2);
     flowrates(init_ex_angle, init_in_angle, gas_vel,
         &fall_rate, &bed_rate, particle_mf);
     dH2_dz = dH2dz(init_ex_angle, init_in_angle, bed_rate);
     dH2_dinit_ex_angle = dH2dinitexangle(init_ex_angle, init_in_angle);
     printf("%7.3f %7.3f %7.3f %7.3f %7.4f %7.4f %7.4f %7.4f %7.4f %7.4f %7.4f %7.4f\n",
```

```
            init_ex_angle, init_in_angle, fall_rate, bed_rate, H1, H2d, H2nd,
            dH2_dz, dH2_dinit_ex_angle, particle_mf[1], particle_mf[2], particle_mf[3]);
        }
    }
```

```
/* SOURCE FILE: DRUM3.C */
/*****************************************************************************/
/* This program executes the end-to-end integration to determine the average */
/* holdup of a flighted rotating drum.                                       */
/*****************************************************************************/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "drum.h"
#include "rkgs.h"

void main(int, char *[]);
void diffdrum(double, double[], double[]);
void drumoutp(double, double[], double[], short, short, short *, double[]);

/*****************************************************************************/
/* main() loads the required data both from the command line and from files  */
/*   listed on the command line. The routine sets the initial contions and   */
/*   calls the integration routine.                                          */
/*****************************************************************************/

void main(argc, argv)
   int argc;
   char *argv[];
{
   char drumfile[13];
   char particlefile[13];
   char mathfile[13];
   FILE *fp;
   short i;
    double init_ex_angle, init_in_angle;
    double particle_mf[MAX_NO_PARTICLE_SIZES];
    double H1, H2, H2d, H2nd;
    double gas_vel;
    double sum_H;
    double y[3], dydx[3], misc[1];
    double zstart, zend;
    short  directive, ret;

   if (argc < 10)
   {
      printf ("Not enough command line arguments.\n");
      exit(0);
   }

   strcpy( drumfile, *(argv+1));

   if ((fp = fopen(drumfile, "r")) == NULL )
   {
      printf ("Can't open file %s.\n", drumfile);
      exit(0);
   }

   if (fp != NULL)
   {
      rewind(fp);
      fscanf(fp, "%[^\n]\n", drum.name);
      fscanf(fp, "%d %lf %lf %lf\n",
            &drum.centerfill, &drum.Do, &drum.Di, &drum.Length);
      fscanf(fp, "%d %lf %lf %d\n",
            &drum.exflight.N, &drum.exflight.D,
            &drum.exflight.length, &drum.exflight.np);
      for (i=0; i<drum.exflight.np; i++)
      {
         fscanf(fp, "%lf %lf\n", &drum.exflight.profile[i].angle,
                                 &drum.exflight.profile[i].face);
      }
      fscanf(fp, "%lf\n", &drum.discharge.init_ex_angle);

      if (drum.centerfill)
      {
         fscanf(fp, "%d %lf %lf %d\n",
             &drum.inflight.N, &drum.inflight.D,
```

```
            &drum.inflight.length, &drum.inflight.np);
        for (i=0; i<drum.inflight.np; i++)
        {
           fscanf(fp, "%lf %lf\n", &drum.inflight.profile[i].angle,
                                   &drum.inflight.profile[i].face);
        }

     }
}
fclose(fp);

strcpy( particlefile, *(argv+2));

if ((fp = fopen(particlefile, "r")) == NULL )
{
   printf ("Can't open file %s.\n", particlefile);
   exit(0);
}

if (fp != NULL)
{
   rewind(fp);
   fscanf(fp, "%[^\n]\n", particle.name);
   fscanf(fp, "%lf %lf %lf %lf %d\n",
          &particle.part_dens, &particle.bulk_dens,
          &particle.repose_angle, &particle.spheric,
          &particle.no_of_sizes);
   for (i=0; i<particle.no_of_sizes; i++)
   {
      fscanf(fp, "%lf %lf\n", &particle.size[i].Dp,
                              &particle.size[i].mf);
   }
}
fclose(fp);

strcpy( mathfile, *(argv+3));

if ((fp = fopen(mathfile, "r")) == NULL )
{
   printf ("Can't open file %s.\n", mathfile);
   exit(0);
}

if (fp != NULL)
{
   rewind(fp);
   fscanf(fp, "%[^\n]\n", convergence.name);
   fscanf(fp, "%lf %lf %lf\n",
          &convergence.reltol, &integ.increment,
          &integ.bound);
}
fclose(fp);

drum.incline = atof( *(argv+4));
operate.rps = atof( *(argv+5));
operate.gas_rate = atof( *(argv+6));
operate.solids_rate = atof( *(argv+7));
operate.temperature = atof( *(argv+8));
drum.discharge.init_ex_angle = atof( *(argv+9));

if (drum.inflight.N == 0 || drum.inflight.D == 0.)
     drum.centerfill = NO;

 gas.dens = airdens(operate.temperature);
 gas.visc = airvisc(operate.temperature);

     init_ex_angle = drum.discharge.init_ex_angle;
     if(drum.centerfill)
         init_in_angle = initinangle(init_ex_angle);
     else
         init_in_angle = PI;
     H1 = fallmass(init_ex_angle, init_in_angle) / particle.bulk_dens;
     H2d = HoldupDEF(init_ex_angle) + HoldupDIF(init_in_angle);
```

```
        H2nd = HoldupNDEF(init_ex_angle, init_in_angle)
                + HoldupNDIF(init_ex_angle, init_in_angle);
        H2 = H1 + H2d + H2nd;

    sum_H = 0.0;

    zstart = 0.0;
    zend = drum.Length;

    y[0] = init_ex_angle;
    y[1] = H2;
    y[2] = sum_H;

    dydx[0] = 1.0;
    dydx[1] = 0.0;
    dydx[2] = 0.0;

    ret = rkgs(zstart, zend, integ.increment, integ.bound, y,
              dydx, 3, &directive, diffdrum, drumoutp, misc);

    sum_H = y[2];

}


/*****************************************************************************/
/* diffdrum() computes the right hand sides of the system of first order     */
/*   ordinary differential equations which describe the change in the        */
/*   initial angle of discharge of the exterior flights, the change in the   */
/*   dense phase holdup, and the change in the accumulated holdup w.r.t.     */
/*   the drum length.                                                        */
/*****************************************************************************/
void diffdrum(z, I, derI)
double z, I[], derI[];
    {
    double init_ex_angle, init_in_angle, fall_rate, gas_vel;
    static double bed_rate = 0.0;
    double particle_mf[MAX_NO_PARTICLE_SIZES];
    double H1, H2, H;
    double dH2_dz, dH2_dinit_ex_angle, dangle_dz;

    init_ex_angle = I[0];
    H2 = I[1];

    if(drum.centerfill)
        init_in_angle = initinangle(init_ex_angle);
    else
         init_in_angle = PI;
    gas_vel = annulusvel(operate.gas_rate, gas.dens, drum.Do, drum.Di, H2);
    flowrates(init_ex_angle, init_in_angle, gas_vel,
        &fall_rate, &bed_rate, particle_mf);
    H1 = fallmass(init_ex_angle, init_in_angle) / particle.bulk_dens;
    dH2_dz = dH2dz(init_ex_angle, init_in_angle, bed_rate);
    dH2_dinit_ex_angle = dH2dinitexangle(init_ex_angle, init_in_angle);
    dangle_dz = dH2_dz/dH2_dinit_ex_angle;
    H = H1 + H2;

    derI[0] = dangle_dz;
    derI[1] = dH2_dz;
    derI[2] = H;
    }

/*****************************************************************************/
/* drumoutp() prints row of results.                                         */
/*****************************************************************************/
void drumoutp(z, I, derI, no_of_bisections, no_of_eqns, directive, misc)
double z, I[], derI[], misc[];
short no_of_bisections, no_of_eqns, *directive;
{

    printf("%7.4f  %7.4f  %7.4f  %7.4f %7.4f %7.4f %7.4f %d\n",
        z, I[0], I[1], I[2], derI[0], derI[1], derI[2], no_of_bisections);
}
```

```
/* SOURCE FILE: INITANGL.C */

#include <math.h>
#include "drum.h"
#include "romberg.h"

/*****************************************************************************/
/* initinangle() returns the initial discharge angle (radians) of interior   */
/*   flights given the initial exterior discharge angle (radians). The       */
/*   routine first computes the volume of material received by an interior   */
/*   flight and then, by trial and error, finds the interior initial discharge */
/*   angle which gives the volume discharge equal to the volume received.    */
/*****************************************************************************/
double initinangle(init_ex_angle)
double init_ex_angle;
    {
    double volume, volume_old, volume_new;
    double angle_a, angle_b, angle_old, angle_new;
    double error, temp;

    angle_a = PI + acos(drum.inflight.D / drum.exflight.D);
    angle_b = PI + acos(-drum.inflight.D / drum.exflight.D);

    /* volume received by interior flight from exterior flights, m3/m. */
    if (init_ex_angle > angle_b)
        return (PI);
    else
    {
        if (init_ex_angle > angle_a)
            volume = drum.exflight.N / drum.inflight.N / 2.
                        * romberg(init_ex_angle, angle_b, convergence.reltol, fct_ex_l2);
        else
                volume = drum.exflight.N / drum.inflight.N / 2.
                        * romberg(angle_a, angle_b, convergence.reltol, fct_ex_l2);
    }

    /* volume of full interior flight at angle 0, m3/m */
    angle_old = 0.;
    volume_old = 1./2. * romberg(angle_old, PI, convergence.reltol, fct_in_l2);

    if (volume_old > volume) /* under filled interior flight */
        {   /* trial and error to find initial interior discharge angle */
        angle_new = 1./2. * PI;
        do
            {
            volume_new = 1./2. * romberg(angle_new, PI, convergence.reltol, fct_in_l2);
            temp = angle_new;
            angle_new = angle_new - (volume_new - volume)
                        * (angle_new - angle_old)
                        / (volume_new - volume_old);
            angle_old = temp;
            volume_old = volume_new;
            error = fabs(volume_new - volume)/volume;
            } while (error > convergence.reltol);
        }
    return (angle_old);
    }


/*****************************************************************************/
/* fct_ex_l2()                                                               */
/*****************************************************************************/
double fct_ex_l2(angle)
double angle;
    {
    double face_length;

    face_length = facelength(angle, drum.exflight.profile, drum.exflight.np);

    return(face_length * face_length);
    }
```

```
/*****************************************************************************/
/* fct_in_l2()                                                               */
/*****************************************************************************/
double fct_in_l2(angle)
double angle;
    {
    double face_length;

    face_length = facelength(angle, drum.inflight.profile, drum.inflight.np);

    return(face_length * face_length);
    }
```

```
/* SOURCE FILE: LONGRATE.C */

#include <math.h>
#include "drum.h"
#include "romberg.h"

double fct_ex_l3(double);
double fct_in_l3(double);

/*************************************************************************/
/* longrate() returns the axial flow rate (kg/s) of the dense phase      */
/*   in a long inclined drum where the bed slope w.r.t thr drum axis can */
/*   be assumed to be zero given the angles (radians) that the exterior  */
/*   and interior angles begin to discharge.                             */
/*************************************************************************/
double longrate(init_ex_angle, init_in_angle)
double init_ex_angle, init_in_angle;
    {
    double angle_d, angle_f, ex_I, in_I, face_length, no_of_surfaces;
    double Din_over_Dex, dense_rate;

    face_length = facelength(init_ex_angle, drum.exflight.profile,
                        drum.exflight.np);

    if (init_ex_angle > PI)
        {
        Din_over_Dex = drum.inflight.D / drum.exflight.D;
        if (drum.centerfill &&
            init_ex_angle>(PI + acos(Din_over_Dex)) &&
            init_ex_angle<(PI + acos(-Din_over_Dex)))
            {
            angle_d = acos(Din_over_Dex * cos(init_in_angle));
            no_of_surfaces = (double) drum.exflight.N
                        * (init_ex_angle - angle_d) / 2./PI + 1.;
            }
        else
            {
            angle_f = 2.*PI - init_ex_angle;
            no_of_surfaces = (double) drum.exflight.N
                    * (init_ex_angle - angle_f) / 2./PI + 1.;
            }
        }
    else
        no_of_surfaces = 1.;

    if(drum.incline != 0.0)
        {
        if(!drum.centerfill)
            {
            in_I = 0.0;
            if(init_ex_angle > PI)
                ex_I = romberg(init_ex_angle, 2.*PI,
                            convergence.reltol, fct_ex_l3);
            else
                {
                if(init_ex_angle > particle.repose_angle + PI/2.)
                    ex_I = romberg(init_ex_angle, 2.*PI,
                                convergence.reltol, fct_ex_l3);
                else
                    ex_I = romberg(init_ex_angle,
                                init_ex_angle+2.*PI/(double)drum.exflight.N,
                                convergence.reltol, fct_ex_l3)
                         + romberg(PI+2.*particle.repose_angle-init_ex_angle,
                                2.*PI, convergence.reltol, fct_ex_l3);
                }
            }
        else
            {
            in_I = 0.0;
            ex_I = romberg(init_ex_angle,
                init_ex_angle+2.*PI/(double)drum.exflight.N,
                convergence.reltol, fct_ex_l3)
                + romberg(PI + 2.*particle.repose_angle - init_ex_angle, 2.*PI,
```

```
                convergence.reltol, fct_ex_l3);
            }

        dense_rate =(operate.rps*particle.bulk_dens*drum.incline)/(6*sin(particle.repose_angle))
            *(PI * face_length * face_length * face_length * no_of_surfaces
            +2.* ((double)drum.exflight.N * ex_I + (double)drum.inflight.N * in_I));
        }
    else
        {
        dense_rate = 0.0;
        }


    return (dense_rate);
    }


/*****************************************************************************/
/* fct_ex_l3() solves the integrand (facelength^3), m3, for dense phase      */
/*    advance for discharging exterior flight at angle (radians).            */
/*****************************************************************************/
double fct_ex_l3(angle)
double angle;
{
    double face_length;

    face_length = facelength(angle, drum.exflight.profile, drum.exflight.np);

    return(face_length * face_length * face_length);
}


/*****************************************************************************/
/* fct_in_l3() solves the integrand (facelength^3), m3, for dense phase      */
/*    advance for discharging interior flight at angle (radians).            */
/*****************************************************************************/
double fct_in_l3(angle)
double angle;
{
    double face_length;

    face_length = facelength(angle, drum.inflight.profile, drum.inflight.np);

    return(face_length * face_length * face_length);
}
```

```
/* SOURCE FILE: FALLRATE.C */
/*****************************************************************************/
/* Contains functions for determining the axial transport rate of airborne   */
/*   phase and the particle size distribution of the drum holdup given the   */
/*   axial transport rate of the dense phase.                                */
/*****************************************************************************/

#include <math.h>
#include "drum.h"
#include "romberg.h"

double Dp;          /* Particle diameter shared by functions within this file */
double avg_gas_vel;/* Average gas velocity also shared within this file.     */
double fct_ex_l2x(double);
double fct_in_l2x(double);

/*****************************************************************************/
/* fallrate() determines the axial transport rate (kg/s) of the airborne     */
/*   phase.  The particle size distribution (mf) of the   drum holdup is also*/
/*   determined.  Inputs to the routine include the initial discharge angles */
/*   of the exterior and interior flights (radians), the gas velocity (m/s)  */
/*   and the dense phase rate (kg/s).                                        */
/*****************************************************************************/
double fallrate(ex_angle, in_angle, gas_vel, bed_rate, conc)
double ex_angle, in_angle, gas_vel, bed_rate, conc[];
{
    double ex_I[MAX_NO_PARTICLE_SIZES], in_I[MAX_NO_PARTICLE_SIZES];
    double sum, fall_rate;
    short i;

    avg_gas_vel = gas_vel;

    for (i = 0, sum = 0.0; i < particle.no_of_sizes; i++)
    {
        Dp = particle.size[i].Dp * particle.spheric;
        if(ex_angle > PI)
           ex_I[i] = romberg(ex_angle, 2.*PI, convergence.reltol, fct_ex_l2x);
        else if(ex_angle > PI/2.+particle.repose_angle)
           ex_I[i] = romberg(ex_angle, 2.*PI, convergence.reltol, fct_ex_l2x);
        else
           ex_I[i] = romberg(PI+2.*particle.repose_angle-ex_angle, 2.*PI,
                                                 convergence.reltol, fct_ex_l2x);
        if (drum.centerfill)
           in_I[i] = romberg(in_angle, PI, convergence.reltol, fct_in_l2x);
        else
           in_I[i] = 0.0;
    }

    for (i=0, sum = 0.0; i < particle.no_of_sizes; i++)
    {
        sum += particle.size[i].mf
            / (operate.rps * particle.bulk_dens / 2.0
            * (drum.exflight.N*ex_I[i] + drum.inflight.N*in_I[i])
            + bed_rate);
    }
    operate.solids_rate = 1.0/sum;
    fall_rate = operate.solids_rate - bed_rate;

    for (i = 0; i < particle.no_of_sizes; i++)
    {
        conc[i] = operate.solids_rate*particle.size[i].mf
            /(operate.rps*particle.bulk_dens/2.0
            *(drum.exflight.N*ex_I[i]+drum.inflight.N*in_I[i])
            + bed_rate);
    }

    return fall_rate;
}
```

```
/* SOURCE FILE: FLOWRATE.C */
/*****************************************************************************/
/* Contains functions for determining the axial transport rates of airborne  */
/*   phase and dense phase and the particle size distribution of the drum    */
/*   holdup.                                                                 */
/*****************************************************************************/

#include <math.h>
#include "drum.h"
#include "romberg.h"

double Dp;          /* Particle diameter shared by functions within this file */
double avg_gas_vel;/* Average gas velocity also shared within this file.     */
double fct_ex_l2x(double);
double fct_in_l2x(double);

/*****************************************************************************/
/* flowrates() determines the axial transport rates (kg/s) of the airborne   */
/*   phase and the dense phase.  The particle size distribution (mf) of the  */
/*   drum holdup is also determined.  Inputs to the routine include the      */
/*   initial discharge angles of the exterior and interior flights (radians) */
/*   and a guess for the dense phase rate (kg/s).                            */
/*****************************************************************************/
void flowrates(ex_angle, in_angle, gas_vel, fall_rate, bed_rate, conc)
double ex_angle, in_angle, gas_vel, *fall_rate, *bed_rate, conc[];
{
    double ex_I[MAX_NO_PARTICLE_SIZES], in_I[MAX_NO_PARTICLE_SIZES];
    double sum, error, temp;
    double bed_rate_old, bed_rate_new;
    double feed_rate_old, feed_rate_new;
    short i;

    avg_gas_vel = gas_vel;

    for (i = 0, sum = 0.0; i < particle.no_of_sizes; i++)
    {
        Dp = particle.size[i].Dp * particle.spheric;
        if(ex_angle > PI)
           ex_I[i] = romberg(ex_angle, 2.*PI, convergence.reltol, fct_ex_l2x);
        else if(ex_angle > PI/2.+particle.repose_angle)
           ex_I[i] = romberg(ex_angle, 2.*PI, convergence.reltol, fct_ex_l2x);
        else
           ex_I[i] = romberg(PI+2.*particle.repose_angle-ex_angle, 2.*PI,
                                                convergence.reltol, fct_ex_l2x);
        if (drum.centerfill)
           in_I[i] = romberg(in_angle, PI, convergence.reltol, fct_in_l2x);
        else
           in_I[i] = 0.0;
    }

    bed_rate_old = *bed_rate;
    for (i=0, sum = 0.0; i < particle.no_of_sizes; i++)
    {
        sum += particle.size[i].mf
            / (operate.rps * particle.bulk_dens / 2.0
            * (drum.exflight.N*ex_I[i] + drum.inflight.N*in_I[i])
            + bed_rate_old);
    }
    feed_rate_old = 1.0/sum;
    bed_rate_new = bed_rate_old - (feed_rate_old-operate.solids_rate)/10.0;

    do
    {
        for (i=0, sum = 0.0; i < particle.no_of_sizes; i++)
        {
            sum += particle.size[i].mf
                / (operate.rps * particle.bulk_dens / 2.0
                * (drum.exflight.N*ex_I[i] + drum.inflight.N*in_I[i])
                 + bed_rate_new);
        }
        feed_rate_new = 1.0/sum;
        temp = bed_rate_new;
        bed_rate_new = bed_rate_new - (feed_rate_new-operate.solids_rate)
```

```
            * (bed_rate_new - bed_rate_old)
            / (feed_rate_new - feed_rate_old);
        bed_rate_old = temp;
        feed_rate_old = feed_rate_new;
        error = fabs((feed_rate_old-operate.solids_rate)/operate.solids_rate);
    }while (error > convergence.reltol);

    *bed_rate = bed_rate_old;
    *fall_rate = operate.solids_rate - *bed_rate;

    for (i = 0; i < particle.no_of_sizes; i++)
    {
        conc[i] = operate.solids_rate*particle.size[i].mf
            /(operate.rps*particle.bulk_dens/2.0
            *(drum.exflight.N*ex_I[i]+drum.inflight.N*in_I[i])
            + *bed_rate);
    }

    return;
}


/*****************************************************************************/
/* fct_ex_l2x() solves the integrand (facelength^2 * falladvance), m3, for   */
/*   mass advance of airborne particles from a discharging exterior flight   */
/*   at angle (radians).                                                     */
/*****************************************************************************/
double fct_ex_l2x(angle)
double angle;
    {
    double face_length, spill_rate, fall_distance, sheet_gas_vel;
    double fall_advance;

    face_length = facelength (angle, drum.exflight.profile,
                    drum.exflight.np);
    spill_rate = spillrate (operate.rps, particle.bulk_dens, face_length);
    fall_distance = falldistanceex(angle, drum.exflight.D, drum.Do,
                                        drum.inflight.D, drum.Di);
    sheet_gas_vel = sheetgasvel (avg_gas_vel, spill_rate,
                            fall_distance, drum.exflight.length);
    fall_advance = falladvance(Dp, fall_distance, sheet_gas_vel);

    return(face_length * face_length * fall_advance);
    }


/*****************************************************************************/
/* fct_in_l2x() solves the integrand (facelength^2 * falladvance), m3, for   */
/*   mass advance of airborne particles from a discharging interior flight   */
/*   at angle (radians).                                                     */
/*****************************************************************************/
double fct_in_l2x(angle)
double angle;
    {
    double face_length, spill_rate, fall_distance, sheet_gas_vel;
    double fall_advance;

    face_length = facelength (angle, drum.inflight.profile,
                    drum.inflight.np);
    spill_rate = spillrate (operate.rps, particle.bulk_dens, face_length);
    fall_distance = falldistancein(angle, drum.inflight.D, drum.Do);
    sheet_gas_vel = sheetgasvel (avg_gas_vel, spill_rate,
                            fall_distance, drum.inflight.length);
    fall_advance = falladvance(Dp, fall_distance, sheet_gas_vel);

    return(face_length * face_length * fall_advance);
    }
```

```
/* SOURCE FILE: FALLMASS.C */
/*****************************************************************************/
/* Functions to determine mass in a state of fall.                           */
/*****************************************************************************/

#include <math.h>
#include "drum.h"
#include "romberg.h"

double fct_ex_l2t(double);
double fct_in_l2t(double);

/*****************************************************************************/
/* fallmass() returns the mass of particles in a state-of-fall (kg/m), given */
/*   the exterior flight initial discharge angle (radians) and the interior  */
/*   flight initial discharge angle (radians).                               */
/*****************************************************************************/
double fallmass(ex_angle, in_angle)
double ex_angle, in_angle;
    {
    double ex_I, in_I;

    if (ex_angle > PI)
       ex_I = romberg (ex_angle, 2.*PI, convergence.reltol, fct_ex_l2t);
    else if(ex_angle > PI/2. + particle.repose_angle)
       ex_I = romberg (ex_angle, 2.*PI, convergence.reltol, fct_ex_l2t);
    else
       ex_I = romberg(PI+2.*particle.repose_angle-ex_angle, 2.*PI,
                                    convergence.reltol, fct_ex_l2t);
    if (drum.centerfill)
        in_I = romberg (in_angle, PI, convergence.reltol, fct_in_l2t);
    else
        in_I = 0.0;
    return (operate.rps * particle.bulk_dens / 2.0
                * (drum.exflight.N * ex_I + drum.inflight.N * in_I));
    }


/*****************************************************************************/
/* fct_ex_l2t() solves the integrand (facelength^2 * falltime), m2 s, for    */
/*   the airborne particles from an exterior flight given the discharge      */
/*   angle (radians).                                                        */
/*****************************************************************************/
double fct_ex_l2t(angle)
double angle;
    {
    double fall_time, face_length, fall_distance;

    face_length = facelength(angle, drum.exflight.profile, drum.exflight.np);
    fall_distance = falldistanceex(angle, drum.exflight.D, drum.Do,
                                    drum.inflight.D, drum.Di);
    fall_time = falltime(fall_distance);
    return (face_length * face_length * fall_time);
    }


/*****************************************************************************/
/* fct_in_l2t() solves the integrand (facelength^2 * falltime), m2 s, for    */
/*   the airborne particles from an interior flight given the discharge      */
/*   angle (radians).                                                        */
/*****************************************************************************/
double fct_in_l2t(angle)
double angle;
    {
    double fall_time, face_length, fall_distance;

    face_length = facelength(angle, drum.inflight.profile, drum.inflight.np);
    fall_distance = falldistancein(angle, drum.inflight.D, drum.Do);
    fall_time = falltime(fall_distance);
    return (face_length * face_length * fall_time);
    }
```

```
/* SOURCE FILE: HOLDUPD.C */
/*****************************************************************************/
/* Function to calculate the dense phase holdup of particles in discharging  */
/*     flights.                                                              */
/*****************************************************************************/

#include <math.h>
#include "drum.h"
#include "rkgs.h"

#define INCREMENT   PI/360.
#define BOUND        0.001

void fct_HDEF(double, double[], double[]);
void fct_HDIF(double, double[], double[]);

/*****************************************************************************/
/* HoldupDEF() returns the holdup (m3/m) of the discharging exterior flights */
/*   given the initial exterior discharge angle (radians).                   */
/*****************************************************************************/
double HoldupDEF(init_ex_angle)
double init_ex_angle;
    {
    double y[2], dydx[2], misc[1];
    short directive, ret;

    y[0] = 0.0;
    y[1] = 0.0;
    dydx[0] = 1.0;
    dydx[1] = 0.0;

    ret = rkgs(2.*PI, init_ex_angle, -INCREMENT, BOUND,
                            y, dydx, 2, &directive, fct_HDEF, out_H, misc);

    return ( drum.exflight.N * y[0] / (4. * PI) );
    }



/*****************************************************************************/
/* fct_HDEF()  Integrand for holdup of discharging exterior flights.         */
/*****************************************************************************/
void fct_HDEF(angle, I, derI)
double angle, I[], derI[];
    {
    double face_length;

    face_length = facelength (angle, drum.exflight.profile, drum.exflight.np);
    derI[0] = I[1];
    derI[1] = face_length * face_length;
    }


/*****************************************************************************/
/* HoldupDIF() returns the holdup (m3/m) of the discharging interior flights */
/*   given the initial interior discharge angle (radians).                   */
/*****************************************************************************/
double HoldupDIF(init_in_angle)
double init_in_angle;
    {
    double y[2], dydx[2], misc[1];
    short directive, ret;

    if ( ! drum.centerfill) return (0.0);

    y[0] = 0.0;
    y[1] = 0.0;
    dydx[0] = 1.0;
    dydx[1] = 0.0;

    ret = rkgs(PI, init_in_angle, -INCREMENT, BOUND, y, dydx,
                2, &directive, fct_HDIF, out_H, misc);
```

```
    return ( drum.inflight.N * y[0] / (4. * PI) );
    }


/*****************************************************************************/
/* fct_HDIF()  Integrand for holdup of interior discharging flights.         */
/*****************************************************************************/
void fct_HDIF(angle, I, derI)
double angle, I[], derI[];
    {
    double face_length;

    face_length = facelength (angle, drum.inflight.profile, drum.inflight.np);
    derI[0] = I[1];
    derI[1] = face_length * face_length;
    }


/*****************************************************************************/
/* out_H()                                                                   */
/*****************************************************************************/
void out_H (angle, I, derI, no_of_bisections, no_of_eqns, directive, misc)
double angle, I[], derI[], misc[];
short no_of_bisections, no_of_eqns, *directive;
    {
    }
```

```
/* SOURCE FILE: HOLDUPND.C */
/*****************************************************************************/
/* Functions to calculate the dense phase holdup in non-discharging flights.*/
/*****************************************************************************/

#include <math.h>
#include "drum.h"
#include "rkgs.h"

#define INCREMENT    PI/360.
#define BOUND        0.001

double init_ex_ang, init_in_ang; /* Initial discharge angles shared by */
                                 /* functions within this file.       */
double ex_to_ex_l2(double);
double ex_to_in_l2(double);
void fct_HNDEF(double, double[], double[]);
void fct_HNDIF(double, double[], double[]);
double in_to_ex_l2(double);

/*****************************************************************************/
/* HoldupNDEF() returns the holdup (m3/m) of the non-discharging exterior   */
/*   flights given the initial exterior discharge angle (radians) and the   */
/*   initial interior discharge angle (radians).                             */
/*****************************************************************************/
double HoldupNDEF(init_ex_angle, init_in_angle)
double init_ex_angle, init_in_angle;
{
    double y[2], dydx[2], misc[1];
    short directive, ret;

    init_ex_ang = init_ex_angle;
    init_in_ang = init_in_angle;

    y[0] = 0.0;
    y[1] = 0.0;
    dydx[0] = 1.0;
    dydx[1] = 0.0;

    ret = rkgs(0.0, init_ex_ang, INCREMENT, BOUND,
                    y, dydx, 2, &directive, fct_HNDEF, out_H, misc);

    return(drum.exflight.N * y[0] / (4. * PI));
}

 9
/*****************************************************************************/
/* fct_HNDEF()  Differential equations for average holdup of non-discharging*/
/*    exterior flights.                                                      */
/*****************************************************************************/
void fct_HNDEF(ex_angle, I, derI)
double ex_angle, I[], derI[];
    {
    double angle_f, angle_c, angle_d, angle_e;
    double angle_a, angle_b, Din_over_Dex;

    derI[0] = I[1];

    if(!drum.centerfill)
        {
        if(init_ex_ang < PI)
            derI[1] = ex_to_ex_l2(ex_angle);
        else
            {
            angle_f = 2.*PI - init_ex_ang;
            if(ex_angle < angle_f)
                derI[1] = ex_to_ex_l2(ex_angle);
            else
                derI[1] = 0.0;
            }
        }
    else
        {
```

```
Din_over_Dex = drum.inflight.D/drum.exflight.D;
angle_c = acos(Din_over_Dex);
if(init_ex_ang < angle_c)
    derI[1] = ex_to_ex_l2(ex_angle);
else
    {
    angle_d = acos(Din_over_Dex * cos(init_in_ang));
    if(init_ex_ang < angle_d)
        {
        if(ex_angle < angle_c)
            derI[1] = ex_to_ex_l2(ex_angle);
        else
            derI[1] = 0.0;
        }
    else
        {
        angle_e = acos(-Din_over_Dex);
        if(init_ex_ang < angle_e)
            {
            if(ex_angle < angle_c)
                derI[1] = ex_to_ex_l2(ex_angle);
            else if(ex_angle < angle_d)
                derI[1] = 0.0;
            else
                derI[1] = in_to_ex_l2(ex_angle);
            }
        else
            {
            if(init_ex_ang < PI)
                {
                if(ex_angle < angle_c)
                    derI[1] = ex_to_ex_l2(ex_angle);
                else if(ex_angle < angle_d)
                    derI[1] = 0.0;
                else if(ex_angle < angle_e)
                    derI[1] = in_to_ex_l2(ex_angle);
                else
                    derI[1] = ex_to_ex_l2(ex_angle);
                }
            else
                {
                angle_a = PI + acos(Din_over_Dex);
                if(init_ex_ang < angle_a)
                    {
                    if(ex_angle < angle_c)
                        derI[1] = ex_to_ex_l2(ex_angle);
                    else if(ex_angle < angle_d)
                        derI[1] = 0.0;
                    else if(ex_angle < angle_e)
                        derI[1] = in_to_ex_l2(ex_angle);
                    else
                        {
                        angle_f = 2.*PI - init_ex_ang;
                        if(ex_angle < angle_f)
                            derI[1] = ex_to_ex_l2(ex_angle);
                        else
                            derI[1] = 0.0;
                        }
                    }
                else
                    {
                    angle_b = PI + acos(-Din_over_Dex);
                    if(init_ex_ang < angle_b)
                        {
                        if(ex_angle < angle_c)
                            derI[1] = ex_to_ex_l2(ex_angle);
                        else if(ex_angle < angle_d)
                            derI[1] = 0.0;
                        else if(ex_angle < angle_e)
                            derI[1] = in_to_ex_l2(ex_angle);
                        else
                            derI[1] = 0.0;
                        }
```

```
                        else
                            {
                            angle_f = 2.*PI - init_ex_ang;
                            if(ex_angle < angle_f)
                                derI[1] = ex_to_ex_l2(ex_angle);
                            else
                                derI[1] = 0.0;
                            }
                        }
                    }
                }
            }
        }
    }


/*************************************************************************/
/* ex_to_ex_l2() solves integrand for holdup of exterior flight at angle    */
/*    (radians) which is receiving material from an exterior flight.       */
/*************************************************************************/
double ex_to_ex_l2(ex_angle)
double ex_angle;
    {
    double disch_angle, face_length;

    disch_angle = 2.*PI - ex_angle;
    face_length = facelength(disch_angle, drum.exflight.profile,
                                          drum.exflight.np);
    return (face_length * face_length);
    }


/*************************************************************************/
/* in_to_ex_l2() solves integrand for holdup of exterior flight at angle    */
/*    (radians) which is receiving material from an interior flight.       */
/*************************************************************************/
double in_to_ex_l2(ex_angle)
double ex_angle;
    {
    double disch_angle, face_length, Dex_over_Din;

    Dex_over_Din = drum.exflight.D / drum.inflight.D;
    disch_angle = acos(Dex_over_Din * cos(ex_angle));
    face_length = facelength(disch_angle, drum.inflight.profile,
                                          drum.inflight.np);
    return ((double)drum.inflight.N/(double)drum.exflight.N
            * Dex_over_Din * sin(ex_angle)/sin(disch_angle)
            * face_length *face_length);
    }


/*************************************************************************/
/* HoldupNDIF() returns the holdup (m3/m) of the non-discharging interior   */
/*   flights given the initial exterior discharge angle (radians) and the   */
/*   initial interior discharge angle (radians).                            */
/*************************************************************************/
double HoldupNDIF(init_ex_angle, init_in_angle)
double init_ex_angle, init_in_angle;
{
    double y[2], dydx[2], misc[1];
    short directive, ret;

    init_ex_ang = init_ex_angle;

    if (!drum.centerfill) return (0.0);

    y[0] = 0.0;
    y[1] = 0.0;
    dydx[0] = 1.0;
    dydx[1] = 0.0;

    ret = rkgs(PI, 2.*PI, INCREMENT, BOUND,
```

```
                    y, dydx, 2, &directive, fct_HNDIF, out_H, misc);

    y[0] += init_in_angle * y[1];

    return(drum.inflight.N * y[0] / (4. * PI));
}


/*****************************************************************************/
/* fct_HNDIF()  Differential equations for average holdup of non-discharging*/
/*    interior flight.                                                       */
/*****************************************************************************/
void fct_HNDIF(in_angle, I, derI)
double in_angle, I[], derI[];
    {
    double angle_a, angle_b, in_angle_a, Din_over_Dex;

    derI[0] = I[1];

    Din_over_Dex = drum.inflight.D / drum.exflight.D;
    angle_a = PI + acos(Din_over_Dex);
    if(init_ex_ang < angle_a)
        derI[1] = ex_to_in_l2(in_angle);
    else
        {
        angle_b = PI + acos(-Din_over_Dex);
        if(init_ex_ang < angle_b)
            {
            in_angle_a = PI + acos(cos(init_ex_ang-PI) / Din_over_Dex);
            if(in_angle < in_angle_a)
                derI[1] = 0.0;
            else
                derI[1] = ex_to_in_l2(in_angle);
            }
        else
            derI[1] = 0.0;
        }
    }


/*****************************************************************************/
/* ex_to_in_l2() solves integrand for holdup of interior flight at angle    */
/*    (radians) which is receiving material from an exterior flight.        */
/*****************************************************************************/
double ex_to_in_l2(in_angle)
double in_angle;
    {
    double disch_angle, face_length, Din_over_Dex;

    Din_over_Dex = drum.inflight.D / drum.exflight.D;
    disch_angle = PI + acos(Din_over_Dex * cos(in_angle-PI));
    face_length = facelength(disch_angle, drum.exflight.profile,
                                          drum.exflight.np);
    return ((double)drum.exflight.N/(double)drum.inflight.N
             * Din_over_Dex * sin(in_angle)/sin(disch_angle)
             * face_length * face_length);
    }
```

```
/* SOURCE FILE: ROMBERG.C */
/*****************************************************************************/
/* romberg() uses the romberg method to solve and return the integral of    */
/*   function f(x) on an interval a <= x <= b.  The relative convergence    */
/*   criterion is given by eps.                                              */
/*   Ref. Hornbeck, R., NUMERICAL METHODS, Quantum Publishers                */
/*   Inc., New York, 1975, p.150.                                            */
/*****************************************************************************/

#include <float.h>
#include <math.h>

#define MAX_POWER_OF_2          14

double romberg (double a, double b, double eps, double (*f)())
{
    double T[17], Told, sum, delx, x;
    int i, j, k, l, n;

    if ((a-b) == 0.0) return 0.0;

    T[0] = (b-a) * ((*f)(a) + (*f)(b))/2.0;
    T[1] = T[0]/2.0 + (b-a) * (*f)((a+b)/2.0)/2.0;
    T[0] = (4.0*T[1] - T[0])/3.0;

    j=1;
    do
    {
        Told = T[0];
        j++;
        delx = (b-a)/(1<<j); /*bit-shift left to get jth power of 2*/
        x = a - delx;
        n = 1<<(j-1);
        sum = 0.0;
        for(i=0; i<n; i++)
        {
            x += 2.0*delx;
            sum += (*f)(x);
        }
        T[j] = T[j-1]/2.0 + delx*sum;
        for(l=1; l<=j; l++)
        {
            k = j - l;
            T[k] = ((1L<<2*l)*T[k+1]-T[k])
                      /((1L<<2*l)-1);
        }
    }while (T[0] != Told
              && (T[0] == 0.0 || fabs((T[0]-Told)/T[0])>eps)
              && j < MAX_POWER_OF_2);
    return(T[0]);
}

#if defined (ROMBERGDEMO)

/* Test driver main() for Romberg integration routine */

#include <stdio.h>

void main()
{
    double a,b,eps;
    double I, romberg(), fct();

    a = -6.0;
    b = 3.0;
    eps = 0.001;

    I = romberg(a, b, eps, fct);
    printf("\n I = %g \n", I);
}


/* Test function to be integrated by Romberg routine */
```

```
double fct(double x)
{
    return (-exp(-x) * cos(exp(-x)));
}

#endif
```

```
/* SOURCE FILE: INTERP.C */
/*****************************************************************************/
/* interp() uses linear interpolation to return a corresponding y value      */
/*   given a x value , a pointer to an array of structures which defines     */
/*   a table of x and y values, and the number of points in the table.  The  */
/*   points in the table must be in ascending order for value x.  If the     */
/*   x value is outside range of x values in the table, then the             */
/*   corresponding y value is found using linear extrapolation of the last   */
/*   two points in the table.                                                */
/*****************************************************************************/

struct xytable
{
    double x;
    double y;
};


double interp(double xvalue, struct xytable *table, short no_of_points)
{
    double yvalue;
    short i=2;

    while ((table+1)->x < xvalue && i < no_of_points)
    {
            table ++;
            i++;
    }
    yvalue = ((xvalue-table->x)/((table+1)->x-table->x)
            *((table+1)->y-table->y) + table->y);
    return yvalue;
}




#if defined (INTERPDEMO)

/* Test driver main() for Linear interpolation routine */

#include <stdio.h>
struct xytable squares[20];

void main()
{
    double interp(), x, x2;
    short np=3;
    squares[0].x = 1.;
    squares[0].y = 1.;
    squares[1].x = 2.;
    squares[1].y = 4.;
    squares[2].x = 3.;
    squares[2].y = 9.;
    x=3.5;
    x2 = interp(x, squares, np);
    printf("\nBy interpolation the square of %g is about %g\n", x, x2);
}
#endif
```

```
/* SOURCE FILE: RKGS.C */
/***************************************************************************/
/* rkgs() solves a system of first order ordinary differential equations    */
/*   with given initial values.                                            */
/*
        Description of parameters
           xstart - lower bound of the interval
           xend   - upper bound of the interval
           h      - initial increment of the independent variable
           errbound - upper error bound. If absolute error is greater than
                    errbound, increment gets halved. If increment is less
                    than h and absolute error less than errbound/50,
                    increment gets doubled.
           y      - input array of initial values. (destroyed) Later on y
                    is the resulting vector of dependent variables computed
                    at intermediate points x.
           dydx   - input vector of error weights. (destroyed) The sum of
                    its components must be equal to 1. Later on dydx is the
                    vector of derivatives, which belong to function values
                    y at a point x.
           no_of_eqns - specifies the number of equations in the system.
           directive - no input value. Function initializes directive to 0.
                    If user wants to terminate integration at any output
                    point, he has to change directive to a non-zero value
                    by means of function outp.
           fct    - the name of an external function used. This function
                    computes the right hand sides dydx of the system to
                    given values x and y. Its parameter list must be x, y,
                    dydx. Function fct should not destroy y.
           outp   - the name of an external output function used. Its
                    parameter list must be x, y, dydx, no_of_bisections,
                    no_of_eqns, directive. If directive is changed to non-
                    zero, function rkgs is terminated.
           misc   - not required or changed by function rkgs, but may be
                    useful for handing result values to the function
                    calling rkgs which are obtained by special manipulations
                    with output data in function outp.
           rkgs   - the function return specifies the number of bisections
                    of the initial increment. If the number gets greater
                    than 10, the function returns 11. 12 or 13 appear in
                    case h = 0 or in case sign(h) != sign(xend-xstart)
                    respectively. 12 if xstart=xend.

        Remarks
           The procedure terminates and returns to the calling program, if
           (1) more than 10 bisections of the initial increment are
               necessary to satisfactory accuracy (return value=11),
           (2) initial increment is equal to 0 or has wrong sign
               (return value 12 or 13),
           (3) the whole integration interval is worked through,
           (4) function outp has changed directive to non-zero.

        Functions required
           The external functions fct(x, y, dydx) and outp(x, y, dydx,
           no_of_bisections, no_of_eqns, directive) must be furnished by
           the user.

        Method
           Evaluation is done by means of forth order Runge-Kutta formulae
           in the modification due to Gill. Accuracy is tested comparing the
           results of the procedure with single and double increment.
           Function rkgs automatically adjusts the increment during the whole
           computation by halving and doubling. If more than 10 bisections of
           the increment are necessary to give satisfactory accuracy, the
           function returns 11 into calling program.
           To get full flexibility in output, an output function must be
           furnished by the user.
           For reference. See
           Ralston/Wilf, Mathematical Methods for Digital Computers,        */
/*         Wiley, New York/London, 1960, pp.110-120.                        */
/***************************************************************************/

#include <float.h>
```

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

#define NOTENOUGHMEMORY     14
#define WRONGSIGN           13
#define ZEROSTEP            12
#define TOOMANYBISECTIONS   11
#define MAXBISECTIONS       10
#define TRUE                 1
#define FALSE                0
#define PROCEED              0

short rkgs(double xstart, double xend, double h, double errbound,
           double y[], double dydx[], short no_of_eqns, short *directive,
                void (*fct)(), void (*outp)(), double misc[])
{
    double a[4], b[4], c[4], *aux;
    short no_of_bisections, ready_to_test, workedthrough;
    short results_ok, i, irec, istep, j;
    double x = xstart;
    double aj, bj, cj, r1, r2, delt;

                                        /* allocate memory for auxillary array */
    aux = (double *) calloc(no_of_eqns * 8, sizeof (double));
    if (aux == NULL)
        return(NOTENOUGHMEMORY);

    for(i = 0; i < no_of_eqns; i++)
        aux[i*8+7] = 0.06666667 * dydx[i];
    *directive = PROCEED;
    (*fct)(x, y, dydx);
    if (h*(xend-x) < 0.0)
        {
        (*outp)(x, y, dydx, WRONGSIGN, no_of_eqns, directive, misc);
        free(aux);
        return (WRONGSIGN);
        }
    if (h*(xend-x) == 0.0)
        {
        (*outp)(x, y, dydx, ZEROSTEP, no_of_eqns, directive, misc);
        free(aux);
        return (ZEROSTEP);
        }
    a[0] = c[0] = c[3] = 0.5;
    a[1] = c[1] = 0.2928932;
    a[2] = c[2] = 1.707107;
    a[3] = 0.1666667;
    b[0] = b[3] = 2.0;
    b[1] = b[2] = 1.0;

                                            /* preparation for first step */
    for (i = 0; i < no_of_eqns; i++)
        {
        aux[i*8+0] = y[i];
        aux[i*8+1] = dydx[i];
        aux[i*8+2] = aux[i*8+5] = 0.0;
        }
    irec = istep = 0;
    workedthrough = FALSE;
    h += h;
    no_of_bisections = (-1);
    do
        {
                                            /* start of a Runge Kutta step */
        if((x+h-xend)*h >= 0.0) workedthrough = TRUE;
        if((x+h-xend)*h > 0.0) h = xend - x;

                                    /* recording of initial values of this step */
        (*outp)(x, y, dydx, irec, no_of_eqns, directive, misc);
        if (*directive != PROCEED)
            {
            free(aux);
```

```
    return (no_of_bisections);
    }
results_ok = ready_to_test = FALSE;

do
    {
    istep ++;
                                /* start of innermost Runge Kutta loop */
    for (j = 0; j < 4; j++)
        {
        aj = a[j]; bj = b[j]; cj = c[j];
        for (i = 0; i < no_of_eqns; i++)
            {
            r1 = h * dydx[i];
            r2 = aj * (r1 - bj*aux[i*8+5]);
            y[i] += r2;
            r2 = r2 + r2 + r2;
            aux[i*8+5] += r2 - cj * r1;
            }
        if (j < 3)
            {
            if (j != 1) x += 0.5 * h;
            (*fct)(x, y, dydx);
            }
        }
                                /* end of innermost Runge Kutta loop */

    if (!ready_to_test)  /* if not ready to test for accuracy */
        {
        for(i = 0; i < no_of_eqns; i++)
            aux[i*8+3] = y[i];
        ready_to_test = TRUE;
        istep = istep + istep - 2;
        no_of_bisections++;
        x -= h;
        h *= 0.5;
        for (i = 0; i < no_of_eqns; i++)
            {
            y[i] = aux[i*8+0];
            dydx[i] = aux[i*8+1];
            aux[i*8+5] = aux[i*8+2];
            }
        }

    else if (istep % 2)                    /* if at an odd step */
        {
        (*fct)(x, y, dydx);
        for (i = 0; i < no_of_eqns; i++)
            {
            aux[i*8+4] = y[i];
            aux[i*8+6] = dydx[i];
            }
        }
    else                                              /* test for accuracy */
        {
        for (i = 0, delt = 0.0; i < no_of_eqns; i++)
             delt += aux[i*8+7] * fabs((aux[i*8+3]-y[i]));
        if (delt > errbound)                           /* error too great */
            {
            if (no_of_bisections > MAXBISECTIONS)
                {
                (*fct)(x, y, dydx);
                (*outp)(x, y, dydx, TOOMANYBISECTIONS,
                                no_of_eqns, directive, misc);
                free(aux);
                return (TOOMANYBISECTIONS);
                }
            for (i = 0; i < no_of_eqns; i++)
                aux[i*8+3] = aux[i*8+4];
            istep = istep + istep - 4;
            x -= h;
            workedthrough = FALSE;
            no_of_bisections++;
```

```
                    x -= h;
                    h *= 0.5;
                    for (i = 0; i < no_of_eqns; i++)
                        {
                        y[i] = aux[i*8+0];
                        dydx[i] = aux[i*8+1];
                        aux[5*8+i] = aux[i*8+2];
                        }
                    }
                else
                    results_ok = TRUE;            /* error within bound */
                }
            }while (!results_ok);

        (*fct)(x, y, dydx);
        for (i = 0; i < no_of_eqns; i++)
            {
            aux[i*8+0] = y[i];
            aux[i*8+1] = dydx[i];
            aux[i*8+2] = aux[i*8+5];
            y[i] = aux[i*8+4];
            dydx[i] = aux[i*8+6];
            }
        (*outp)(x-h, y, dydx, no_of_bisections, no_of_eqns, directive, misc);
        if(*directive != PROCEED)
            {
            free(aux);
            return (no_of_bisections);
            }
        for (i = 0; i < no_of_eqns; i++)
            {
            y[i] = aux[i*8+0];
            dydx[i] = aux[i*8+1];
            }
        irec = no_of_bisections;
        no_of_bisections--;
        istep /= 2;
        h += h;
        if (no_of_bisections >= 0 && !(istep%2) && delt < 0.02 * errbound)
            {                                          /* increment gets doubled */
            no_of_bisections--;
            istep /= 2;
            h += h;
            }
        }while(!workedthrough);
    (*outp)(x, y, dydx, no_of_bisections, no_of_eqns, directive, misc);
    free(aux);
    return (no_of_bisections);
    }

#if defined(RKGSDEMO)

/* Test driver for Runge Kutta integration function.                          */
/* To debug with MSC, compile with cl /D RKGSDEMO rkgs.c                      */

#include <stdio.h>

void main(void);
short rkgs(double xstart, double xend, double h, double errbound,
                    double y[], double dydx[], short no_of_eqns,
                    short *directive, void (*function)(), void (*output)(),
                    double misc[]);
void function(double x, double y[], double dydx[]);
void output(double x, double y[], double dydx[],
                    short no_of_bisections, short no_of_eqns,
                    short *directive, double misc[]);

void main(void)
{
    double xstart = -6.0, xend = 3.0, h = 1.0, errbound = 0.001;
    double y[1], dydx[1], misc[1];
    short no_of_eqns = 1, directive, ret;
```

```
    y[0] = 0.0;          /* initial value */
    dydx[0] = 1.0;       /* error weight  */

    ret = rkgs(xstart, xend, h, errbound, y, dydx, no_of_eqns,
                &directive, function, output, misc);
    switch (ret)
    {
        case TOOMANYBISECTIONS :
            printf("\nIntegration terminated: too many bisections\n");
            break;
        case WRONGSIGN :
            printf("\nIntegration terminated: increment has wrong sign\n");
            break;
        case ZEROSTEP :
            printf("\nIntegration terminated: zero increment or interval\n");
            break;
        default :
            if (directive)
                printf("\nItegration terminated by output function\n");
            else
                printf("\nIntegration interval worked through\n");
    }
}

/* function to solve differential(s) */
void function(x, y, dydx)
double x, y[], dydx[];
    {
    dydx[0] = -exp(-x) * cos(exp(-x));
    }

/* output function */
void output(x, y, dydx, no_of_bisections, no_of_eqns, directive, misc)
double x, y[], dydx[], misc[];
short no_of_bisections, no_of_eqns, *directive;
    {
    printf("%18e %18e %18e %5hd\n", x, y[0], dydx[0], no_of_bisections);
    }

#endif
```

```
/* SOURCE FILE: ROUTINES.C */
/*****************************************************************************/
/* Miscellaneous low level routines for solids movement in flighted rotating */
/* drum program.                                                             */
/*****************************************************************************/

#include <math.h>
#include "drum.h"
#include "interp.h"

/*****************************************************************************/
/* annulusvel() returns the velocity (m/s) in a annular passage given the    */
/*   mass flow rate (kg/s), the density (kg/m3), the inner and outer         */
/*   diameters (m), and area occupied by dense phase (m2).                   */
/*****************************************************************************/
double annulusvel(mass_rate, density, Do, Di, H2)
double mass_rate, density, Do, Di, H2;
    {
    double area, velocity;

    area = 0.25 * PI * (Do * Do - Di * Di) - H2;
    velocity = mass_rate / density / area;

    return (velocity);
    }


/*****************************************************************************/
/* airdens() returns the density (kg/m3) of air at one atmosphere and the    */
/*   given temperature (C).                                                  */
/*****************************************************************************/
double airdens(temperatureC)
double temperatureC;
    {
    return (354.3 / (temperatureC + 273.15));
    }


/*****************************************************************************/
/* airvisc() returns the viscosity (kg/m s) of air at one atmosphere and the */
/*   given temperature (C).                                                  */
/*****************************************************************************/
double airvisc(tempC)
double tempC;
    {
    double TempK, visc;
    double TempK0 = 150.0 + 273.15, visc0 = 2.4e-5, c = 120.0;

    TempK = tempC + 273.15;
    visc = visc0 * ((TempK0 + c) / (TempK + c))
                    * pow(TempK/TempK0, 3./2.);

    return (visc);
    }


/*****************************************************************************/
/* falldistanceex() returns the vertical fall distance (metres) of a         */
/*   particle discharged from an exterior flight tip at an angle (radians)   */
/*   to the horizontal centerline. The outer shell diameter (meters), the    */
/*   exterior flight tip lotus diameter (meters), the inner shell diameter   */
/*   (metres) and the interior flight tip lotus diameter (metres) must be    */
/*   supplied.                                                               */
/*****************************************************************************/
double falldistanceex(double angle, double Dex, double Do,
                                                double Din, double Di)
{
    double y, anglea, angleb, angle2, angle3;

    if (angle <= 0. || angle >= 2.*PI)
        y = 0.0;
    else if (!drum.centerfill || angle <= PI)
        y = Do/2 * sin(acos(Dex/Do * cos(angle))) - Dex/2. * sin(angle);
```

```
    else
    {
        anglea = PI + acos(Din/Dex);
        angleb = PI + acos(-Din/Dex);
        if (angle < anglea || angle > angleb)
            y = Do/2*sin(acos(Dex/Do*cos(angle))) - Dex/2*sin(angle);
        else
        {
            angle2 = PI + acos(Di/Dex);
            angle3 = PI + acos(-Di/Dex);
            if(angle < angle2 || angle > angle3)
                y = -Dex/2.*sin(angle);
            else
                y = -Dex/2.*sin(angle) + Di/2*sin(acos(Dex/Di*cos(angle)));
        }
    }
    return (y);
}


/*************************************************************************/
/* falldistancein() returns the vertical fall distance (metres) of a     */
/*   particle discharged from an interior flight tip at an angle (radians)*/
/*   to the horizontal centerline. The interior drum flight tip lotus    */
/*   diameter (meters) and diameter (metres) of the outer shell must be  */
/*   supplied.                                                           */
/*************************************************************************/
double falldistancein(double angle, double Din, double Do)
    {

    if(angle>=0. && angle<=PI)
        return(Do/2.* sin(acos(Din/Do*cos(angle))) - Din/2*sin(angle));
    else return(0.);
    }


/*************************************************************************/
/* falltime() returns the time (seconds) for a particle to fall a veritcal */
/*   distance (metres).                                                  */
/*************************************************************************/
double falltime(double fall_distance)
    {
    if (fall_distance <= 0.0)
        return (0.0);
    else
        return (sqrt(2.*fall_distance/GRAVITY));
    }


/*************************************************************************/
/* facelength() returns the length (metres) of the surface of the material */
/*   on a discharging flight whose tip is at angle (radians) to the      */
/*   horizontal centerline.  The reference length (metres) and an array of */
/*   structures containing a table of angles (radians) and ratios of the */
/*   surface lengths to the reference length are also supplied.          */
/*************************************************************************/
double facelength(double angle, struct flightprofile *flight, short np)

    {
    double length;
    length = interp(angle-particle.repose_angle, flight, np);
    return length;
    }


/*************************************************************************/
/* Re() returns the dimensionless Reynolds number given the gas density  */
/*   (kg/m3), the gas velocity (m/s), the gas viscosity (kg/m s) and a   */
/*   characteristic length (m).                                          */
/*************************************************************************/
double Re(density, velocity, length, viscosity)
double density, velocity, length, viscosity;
```

```
    {
    return (density*velocity*length/viscosity);
    }


/*****************************************************************************/
/* CD() returns the drag coefficient for a spherical particle given the     */
/*   dimensionless particle Reynolds number. ref.Schiller and Naumann (1933)*/
/*****************************************************************************/
double CD (Rep)
double Rep;
    {
    if (Rep < 0.2) return (24./Rep);
    else return (24./Rep * (1.0 + 0.15*pow(Rep,0.687)));
    }


/*****************************************************************************/
/* spillrate() returns the discharge rate (kg/s m) of a flight. The drum     */
/*   rotational speed (rps), the bulk density of the material (kg/m3), and   */
/*   the length of the surface of the material on the flight (metres) are    */
/*   supplied.                                                               */
/*****************************************************************************/
double spillrate (rps, bulk_dens, face_length)
double rps, bulk_dens, face_length;
    {
    return (PI * rps * bulk_dens * face_length * face_length);
    }


/*****************************************************************************/
/* sheetgasvel() returns the velocity (m/s) of the gas within the sheet of   */
/*   falling particles, given the superficial gas velosity (m/s), the flight */
/*   discharge rate (kg/s m), the vetical fall distance (m), and the axial   */
/*   length (m) of the flight.                                               */
/*****************************************************************************/
double sheetgasvel (ave_gas_vel, spill_rate, fall_distance, flight_length)
double ave_gas_vel, spill_rate, fall_distance, flight_length;
    {
    double ratio;

    if (spill_rate <= 0.0 || flight_length <= 0.0 || fall_distance <= 0.0)
        {
        ratio = 1.0;
        }
    else
        {
        ratio = 1.546 * pow(fall_distance,0.763) / pow(spill_rate,0.857)
                      / pow(flight_length,0.437);
        if (ratio > 1.0) ratio = 1.0;
        }
    return (ratio*ave_gas_vel);
    }
```

```
/* SOURCE FILE: ADVANCE.C */

#include <math.h>
#include "drum.h"

/*****************************************************************************/
/* falladvance() calculates the axial advance (metres) of fallen particle in */
/*   the direction of the gas flow.  The particle diameter (metres), the     */
/*   vertical fall distance (metres) and the gas velocity (m/s) are supplied.*/
/*****************************************************************************/
double falladvance(Dp, fall_distance, gas_vel)
double Dp, fall_distance, gas_vel;
    {
    double Rep, K, a, tf, temp, advance;

    if (gas_vel == 0.0)
        {
        advance = fall_distance * sin(drum.incline);
        }
    else
        {
        Rep = Re (gas.dens, gas_vel, Dp, gas.visc);
        K = 0.75 * gas.dens * CD(Rep) / particle.part_dens / Dp;
        tf = falltime (fall_distance);

        if (drum.incline == 0.0)
            {
                advance = gas_vel*tf + log(1./(K*gas_vel*tf+1.))/K;
            }
        else
            {
            if (drum.incline > 0.0)
                {
                        a = sqrt (GRAVITY * sin(drum.incline) / K);
                        temp = atan (gas_vel/a);
                        advance = gas_vel*tf + log(cos(temp)/cos(-a*K*tf+temp))/K;
                }
            else        /* drum.incline < 0.0  */
                {
                        a = sqrt (-GRAVITY * sin(drum.incline) / K);
                        advance = -(a-gas_vel)*tf
                        -log((a+gas_vel+(a-gas_vel)*exp(-2.*a*K*tf))/(2.*a))/K;
                }
            }
        }

    return (advance);
    }
```

```
/* SOURCE FILE: DH2DZ.C */

#include <math.h>
#include "drum.h"
#include "romberg.h"

double fct_ex_l3(double);
double fct_in_l3(double);

/*************************************************************************/
/* dH2dz() returns the rate of change in the dense phase holdup w.r.t.   */
/*   the drum length (m3/m/m) given the angles (radians) that the        */
/*   exterior and interior angles begin to discharge and the flow rate   */
/*   of the dense phase (kg/s).                                          */
/*************************************************************************/
double dH2dz(init_ex_angle, init_in_angle, bed_rate)
double init_ex_angle, init_in_angle, bed_rate;
    {
    double angle_d, angle_f, ex_I, in_I, face_length, no_of_surfaces;
    double bed_slope, dH2_dz, Din_over_Dex;

    face_length = facelength(init_ex_angle, drum.exflight.profile,
                        drum.exflight.np);

    if (init_ex_angle > PI)
        {
        Din_over_Dex = drum.inflight.D / drum.exflight.D;
        if (drum.centerfill &&
            init_ex_angle>(PI + acos(Din_over_Dex)) &&
            init_ex_angle<(PI + acos(-Din_over_Dex)))
            {
            angle_d = acos(Din_over_Dex * cos(init_in_angle));
            no_of_surfaces = (double) drum.exflight.N
                        * (init_ex_angle - angle_d) / 2./PI + 1.;
            }
        else
            {
            angle_f = 2.*PI - init_ex_angle;
            no_of_surfaces = (double) drum.exflight.N
                    * (init_ex_angle - angle_f) / 2./PI + 1.;
            }
        }
    else
        no_of_surfaces = 1.;

    if(drum.incline != 0.0)
        {
        if(!drum.centerfill)
            {
            in_I = 0.0;
            if(init_ex_angle > PI)
                ex_I = romberg(init_ex_angle, 2.*PI,
                            convergence.reltol, fct_ex_l3);
            else
                {
                if(init_ex_angle > particle.repose_angle + PI/2.)
                    ex_I = romberg(init_ex_angle, 2.*PI,
                                convergence.reltol, fct_ex_l3);
                else
                    ex_I = romberg(init_ex_angle,
                                init_ex_angle+2.*PI/(double)drum.exflight.N,
                                convergence.reltol, fct_ex_l3)
                         + romberg(PI+2.*particle.repose_angle-init_ex_angle,
                                2.*PI, convergence.reltol, fct_ex_l3);
                }
            }
        else
            {
            in_I = 0.0;
            ex_I = romberg(init_ex_angle,
                init_ex_angle+2.*PI/(double)drum.exflight.N,
                convergence.reltol, fct_ex_l3)
                + romberg(PI + 2.*particle.repose_angle - init_ex_angle, 2.*PI,
```

```
                convergence.reltol, fct_ex_l3);
            }

        bed_slope = 1./cos(particle.repose_angle)
            *(1./(PI * face_length * face_length * face_length * no_of_surfaces)
            * (6. * sin(particle.repose_angle) *bed_rate / (operate.rps * particle.bulk_dens)
            - drum.incline * 2.
            * ((double)drum.exflight.N * ex_I + (double)drum.inflight.N * in_I))
            - drum.incline);
        }
    else
        {
        bed_slope = tan(particle.repose_angle)
            /(PI * face_length * face_length * face_length * no_of_surfaces)
            * 6. * bed_rate / (operate.rps * particle.bulk_dens);
        }

    if(bed_slope > particle.repose_angle)
        bed_slope = particle.repose_angle;

    dH2_dz = no_of_surfaces * face_length * bed_slope;

    return (dH2_dz);
    }


/*****************************************************************************/
/* fct_ex_l3() solves the integrand (facelength^3), m3, for dense phase      */
/*    advance for discharging exterior flight at angle (radians).            */
/*****************************************************************************/
double fct_ex_l3(angle)
double angle;
{
    double face_length;

    face_length = facelength(angle, drum.exflight.profile, drum.exflight.np);

    return(face_length * face_length * face_length);
}


/*****************************************************************************/
/* fct_in_l3() solves the integrand (facelength^3), m3, for dense phase      */
/*    advance for discharging interior flight at angle (radians).            */
/*****************************************************************************/
double fct_in_l3(angle)
double angle;
{
    double face_length;

    face_length = facelength(angle, drum.inflight.profile, drum.inflight.np);

    return(face_length * face_length * face_length);
}
```

```
/* SOURCE FILE: DHDANGLE.C */
/*****************************************************************************/
/* Functions to calculate the rate of change in drum dense phase holdup     */
/*   with respect to the initial discharge angle of the exterior flights.   */
/*****************************************************************************/

#include <math.h>
#include "drum.h"
#include "romberg.h"

/*****************************************************************************/
/* dH2dinitexangle() returns the rate of change in the dense phase holdup   */
/*   with respect to the initial discharge angle of the exterior flights    */
/*   (m3/m/radians) given the initial exterior discharge angle (radians)    */
/*   and the initial interior discharge angle (radians).                    */
/*****************************************************************************/
double dH2dinitexangle(init_ex_angle, init_in_angle)
double init_ex_angle, init_in_angle;
    {
    double angle_c, angle_d, angle_e, Din_over_Dex;
    double angle_a, angle_b, in_angle_b, in_angle_a;
    double face_length_ex, face_length_in, dH_dinit_ex_angle;

    if(!drum.centerfill)
        {
        if(init_ex_angle < PI)
            dH_dinit_ex_angle = - (double)drum.exflight.N / (4.*PI)
                * romberg(init_ex_angle, 2.*PI-init_ex_angle,
                    convergence.reltol, fct_ex_l2);
        else
            {
            face_length_ex = facelength(init_ex_angle, drum.exflight.profile,
                            drum.exflight.np);
            dH_dinit_ex_angle = - (double)drum.exflight.N / (4.*PI)
                * 2. * (init_ex_angle - PI)
                 * face_length_ex *face_length_ex;
            }
        }
    else
        {
        Din_over_Dex = drum.inflight.D / drum.exflight.D;
        angle_c = acos(Din_over_Dex);
        if(init_ex_angle < angle_c)
            {
            dH_dinit_ex_angle = - (double)drum.exflight.N / (4.*PI)
                * romberg(init_ex_angle, 2.*PI-init_ex_angle,
                    convergence.reltol, fct_ex_l2);
            }
        else
            {
            angle_d = acos(Din_over_Dex * cos(init_in_angle));
            if(init_ex_angle < angle_d)
                {
                dH_dinit_ex_angle = - (double)drum.exflight.N / (4.*PI)
                    * (romberg(init_ex_angle, 2.*PI,
                        convergence.reltol, fct_ex_l2)
                    + romberg(2.*PI, 2.*PI-angle_c,
                        convergence.reltol, fct_ex_l2));
                }
            else
                {
                angle_e = acos(-Din_over_Dex);
                if(init_ex_angle < angle_e)
                    {
                    in_angle_b = acos(cos(init_ex_angle) / Din_over_Dex);
                    dH_dinit_ex_angle = - (double)drum.exflight.N / (4.*PI)
                        * (romberg(init_ex_angle, 2.*PI,
                            convergence.reltol, fct_ex_l2)
                        + romberg(2.*PI, 2.*PI-angle_c,
                            convergence.reltol, fct_ex_l2)
                        - (double)drum.inflight.N / (double)drum.exflight.N
                        * romberg(init_in_angle, in_angle_b,
                            convergence.reltol, fct_in_l2));
```

```
                }
            else
                {
                if(init_ex_angle < PI)
                    {
                    dH_dinit_ex_angle = - (double)drum.exflight.N / (4.*PI)
                        * (romberg(init_ex_angle, 2.*PI,
                            convergence.reltol, fct_ex_l2)
                        + romberg(2.*PI, 2.*PI-angle_c,
                            convergence.reltol, fct_ex_l2)
                        - (double)drum.inflight.N / (double)drum.exflight.N
                            * romberg(init_in_angle, PI,
                                convergence.reltol, fct_in_l2)
                        + romberg(2.*PI-angle_e, 2.*PI-init_ex_angle,
                            convergence.reltol, fct_ex_l2));
                    }
                else
                    {
                     angle_a = PI + acos(Din_over_Dex);
                    if(init_ex_angle < angle_a)
                        {
                        face_length_ex = facelength(init_ex_angle,
                             drum.exflight.profile, drum.exflight.np);
                        dH_dinit_ex_angle = - (double)drum.exflight.N / (4.*PI)
                            * (romberg(init_ex_angle, 2.*PI,
                                convergence.reltol, fct_ex_l2)
                            + romberg(2.*PI, 2.*PI-angle_c,
                                convergence.reltol, fct_ex_l2)
                            - (double)drum.inflight.N / (double)drum.exflight.N
                                * romberg(init_in_angle, PI,
                                    convergence.reltol, fct_in_l2)
                            + romberg(angle_a, init_ex_angle,
                                convergence.reltol, fct_ex_l2)
                            + 2. * (init_ex_angle - PI) * face_length_ex *face_length_ex);
                        }
                    else
                        {
                         angle_b = PI + acos(-Din_over_Dex);
                        if(init_ex_angle < angle_b)
                            {
                             in_angle_a = PI + acos(cos(init_ex_angle - PI)
                                                / Din_over_Dex);
                            face_length_ex = facelength(init_ex_angle,
                                 drum.exflight.profile, drum.exflight.np);
                            face_length_in = facelength(init_in_angle,
                                 drum.inflight.profile, drum.inflight.np);
                            dH_dinit_ex_angle = - (double)drum.exflight.N / (4.*PI)
                                * (romberg(init_ex_angle, 2.*PI,
                                    convergence.reltol, fct_ex_l2)
                                + romberg(2.*PI, 2.*PI-angle_c,
                                    convergence.reltol, fct_ex_l2)
                                - romberg(init_ex_angle, angle_b,
                                    convergence.reltol, fct_ex_l2)
                                + (- angle_e + angle_b + init_in_angle - in_angle_a + 2.*PI)
                                    * face_length_ex *face_length_ex);
                            }
                        else
                            {
                            face_length_ex = facelength(init_ex_angle,
                                 drum.exflight.profile, drum.exflight.np);
                            dH_dinit_ex_angle = - (double)drum.exflight.N / (4.*PI)
                                * 2. * (init_ex_angle - PI)
                                * face_length_ex *face_length_ex;
                            }
                        }
                    }
                }
            }
        }
    }
return (dH_dinit_ex_angle);
}
```