WestVirginiaUniversity
THE RESEARCH REPOSITORY @ WVU

Graduate Theses, Dissertations, and Problem Reports

1999

Simulation of packed column jigging

Qiang Dai
West Virginia University

Follow this and additional works at: https://researchrepository.wvu.edu/etd

Recommended Citation
Dai, Qiang, "Simulation of packed column jigging" (1999). *Graduate Theses, Dissertations, and Problem Reports*. 966.
https://researchrepository.wvu.edu/etd/966

This Thesis is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Thesis has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

Simulation of Packed Column Jigging

Qiang Dai

Thesis Submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Mining Engineering

David C. Yang, Ph.D., Chair
Eung Ho Cho, Ph.D.
Felicia F. Peng, Ph.D.

Department of Mining Engineering

Morgantown, West Virginia
1999

Keywords: Gravitational Separation, Jig Separation,
Packed Column, Simulation, Fine Particle.

Copyright 1999 Qiang Dai

Abstract

Simulation of Packed Column Jigging

Qiang Dai

Packed column jig has been demonstrated a number of promising results in processing iron ore, coal, chromite and other finely disseminated particulate materials. The packing materials in the jig divide the column into a large number of virtual cells. Each cell performs the similar separation action. A model about the movement of a particle in the packed column jig was derived to interpret the separation mechanism of particles in the virtual cell. A computer program was developed to simulate the particle movement in the jig based on the derived model, which can simulate the packed column jigging with different jig geometry, ore properties and operation conditions. It also allows a visual simulation of particles moving in the packed column jig. By analyzing the simulation results, the effects of jig cell dimensions, pulsating frequency, stroke length, particle size, mineral density, particle packing density in the jig, etc. were obtained.

Acknowledgement

The author is grateful to express his sincere appreciation to his advisor, Dr. David C. Yang, for his persistent encouragement, support and guidance during the period of graduate study and thesis work. The valuable and constructive suggestions, comments and advice by the thesis committee members, Dr. Eung Ha Cho and Dr. Felicia F. Peng, are greatly appreciated. The author also thanks to Dr. Paolo Bozzato, who is a visiting scholar from Italy, for his friendship and helpful suggestions.

Special gratitude is extended to the author's parents, his wife Jun Yang, and his daughter Yang Dai, for their constant encouragement, understanding, support and endurance during the course of the study.

Table Of Content

List Of Symbols

C	The linear concentration of particles
C_d	The size distribution parameter
$\bar{d}$	The average diameter of particles
d	Diameter of particle
d_1	The average particle size of mineral 1
d_2	The average particle size of mineral 2
D_{max}	The maximum particle size of heavy mineral
d_{max}	The maximum particle size of light mineral
F_B	Bagnold dissipation force
F_{D1}	Drag force from velocity difference between a particle and the fluid
F_{D2}	Drag force for accelerating the adjacent fluid
F_f	Buoyant force
F_M	Magnetic force
F_p	Fluid pressure force due to acceleration
F_R	Mechanical resistance force by interaction of particles
$f(\lambda)$	A function related to the solid concentration in stream
G	Gravity force
g	Gravitational acceleration
h	The distance from the bottom of the packed column jig
h_i	The bottom height of ith cell
h_{i+1}	The top height of ith cell

H_1	The height of the feed point of packed column jig
H_2	The height of the packed column jig
k_B	The coefficient related to the particle shape
k_m	A constant of particle momentum transfer coefficient
k_s	A constant related to magnetic force exerted on particle
l	The stroke length
m	The mass of particle
n	Stream velocity distribution coefficient
n'	A constant related to Reynolds number
P	The dissipation pressure between two adjacent layers of stream
ΔP	The dissipation pressure difference along the particle
ΔP_x	The dissipation pressure difference along the particle in X direction
ΔP_y	The dissipation pressure difference along the particle in Y direction
ΔP_z	The dissipation pressure difference along the particle in Z direction
r	The distance from the centerline
R	The maximum distance from the centerline
S.L	The stroke length
t_0	The start point of time counting
T	The period of time counting
T_{in}	The shear between two adjacent layers of stream
u	Stream velocity
u'	Fluctuating stream velocity
u_c	The centerline stream velocity

u_m	The average up-velocity of stream
u_x	Stream velocity in X direction
u_y	Stream velocity in Y direction
u_z	Stream velocity in Z direction
$\overline{u}$	Average stream velocity in certain point
$\frac{du}{dy}$	The stream velocity gradient
v	Particle velocity
V_c	The solid volume concentration in packed column jig
v_x	Particle velocity in X direction
v_y	Particle velocity in Y direction
v_z	Particle velocity in Z direction
$\overline{v}_m$	The average settling velocity of particles in a cell
$\overline{v}$	The average superficial velocity in a cell
Δv	The average velocity difference between the interested particle and interaction particles
Δv_m	The maximum time-average velocity difference between the interested particle and interaction particles
x	Particle size
X_{max}	The maximum particle size
y+	The fraction of particles with particle size larger than certain value
α	The content of heavy mineral in feed
γ_1	The content of mineral 1

γ_2	The content of mineral 2
ρ	The pulp density
ρ_s	Density of particle
$\overline{\rho}_S$	The average density of solid particles
ρ_{s1}	The density of mineral 1
ρ_{s2}	The density of mineral 2
ρ_w	The density of water
$\overline{\rho}_{Si}$	The average particle density in ith cell
μ	Stream viscosity
ξ	Mass combination coefficient
λ	The particle volume concentration
λ_0	The maximum volume concentrate of particles
ζ	Turbulent intensity
ν	The pulsating frequency
ϕ	A parameter of the feasibility of separation in packed column jig
ψ	Drag force coefficient
θ	The "dynamic" friction angle
χ	Magnetic susceptibility

List Of Tables

Table Of Figures

Chapter 1 Introduction

As the mineral resources gradually becomes lower-graded, finer-disseminated and more complex, more efficient and more environmentally-safe separation methods are needed for mineral processing. The research and development of more efficient gravity separation methods meets these needs. It has been concentrated on development of new centrifugal separator, jig separator and spiral separator.

Packed column jig is one of these developments. It comprises a packed column filled with the corrugated packed materials and a pulsating device for generating jigging action within the column. It has attained a number of promising separation results for various ores in laboratory. But still little is known about its many interacting factors that affect fine particle separation. Thus, there is a need to determine the effects of these parameters on the packed column jig performance. Simulation is an efficient approach to do this and it is used in this thesis research work. The objectives of the thesis work include:

- Setting the mathematical model for describing packed column jigging process.
- Simulating the packed column jigging process in computer.
- Running the simulation program and getting the effects of different parameters.

Chapter 2 Literature Review

2.1 Introduction To Jigging Technology

Jigging is a gravity separation method known for centuries. It is based on the difference of settling velocity among particles in a pulsating current and widely used in coal preparation, mineral processing as well as other solid materials separation.

When particles with different density and size are in a pulsating current, each particle will bear gravity force, fluid forces and other mechanic forces by interaction with other particles or the wall of the jig. While the particle movement in the jigging process is complicated and difficult to be accurately described, the average settling velocity of a particle is considered to be mainly determined by the particle density and particle size. By adjusting the frequency and amplitude of the pulsating as well as the net upper velocity of the current, the feed solid particles can be separated by density and size in the jig machine. Figure 2.1 shows the scheme of a conventional jig.

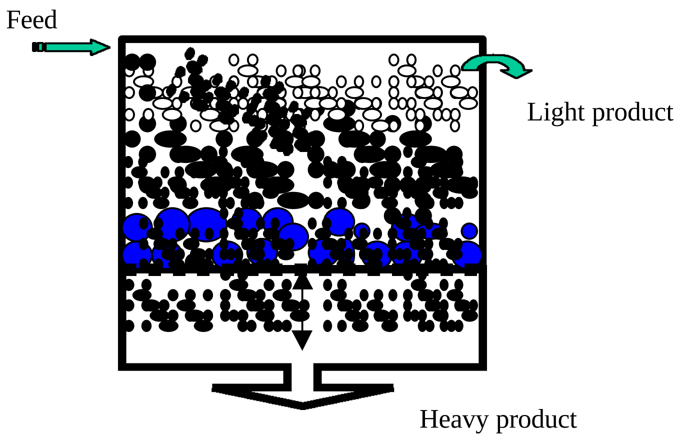


Figure 2. 1 The schematic of a conventional jig

According to the medium used in the jigging process, jigging can be divided into dry jigging (with air as the medium) and wet jigging (with water as the medium). The wet jigging is more applicable to mineral processing today than the dry jigging. The conventional jig was usually used for separating coarse to middle size solid materials. It was generally known that separation of fine particles is beyond the capability of conventional jigs. But the developments of new jigging technology in recent years (Yang, 1996, 1997; Breuer et al. 1986; Campbell & Thomas, 1991), make it possible to process fine particles with a number of promising results.

2.2 Recent Advancement Of Jigging Technology

In recent years, new jigging processes have been developed (Holland-Batt, 1998; Lyman, 1992). These processes not only have improved the conventional jigging process by some modifications, but also result from the inventions of new jigging machines to process fine particles. Packed column jig is one of these inventions.

2.2.1 Conventional Jig

Conventional jig is still most extensively used in industry. There are some developments for improving its separation efficiency. There are:

- **Double-frequency pulsation system** (Fellensiek, E. 1986, Breuer, H. & Jungmann, A. 1986). By making the pulsation to double-frequency, the jig with this novel pulsating system resulted in an improved separation efficiency, higher yield and capacity, and considerable hutch water saving in processing of various coal samples. A finer size down to 0.1mm can be separated in the jig with a high efficiency.
- **Adding an intermediate layer** (Schonert, K. Gerstenberg, R. & Zimmermann, W. 1987; De Jong, T.P.R. & Dalmijn, W.L. 1997). A jig

operation with a density sensitive filter layer proved to be an effective method to improve the separation in conventional jig. It was reported that the particle size could be down to 20 micron and still good separation was attained in ash reduction of coal fines. By adding a continuously recycled intermediate layer fraction to the conventional jig, the layer was reported to form a distinct layer between the heavy and light fractions in the jig, enhancing the separation efficiency.

Some other improvements in conventional jig also include the optimizing the pulsating action and extending its application in industry (Eichholz et al., 1998; Rong et al., 1992; Mesters et al., 1997).

2.2.2 Magnetic Jig

The idea of magnetic jigging is to use both magnetic and gravity fields for fine particle separation. Fig 2.2 shows the schematic of a magnetic jig. The magnetic field has a gradient in the vertical direction to make the magnetic particle exerted by a magnetic force in the vertical direction. According to a research on the magnetic jig, the relatively weak magnetic force gives the ability of separation of refractory mixtures. However, the test data of this research work is still not available in the literature (in et al., 1998, 1997).

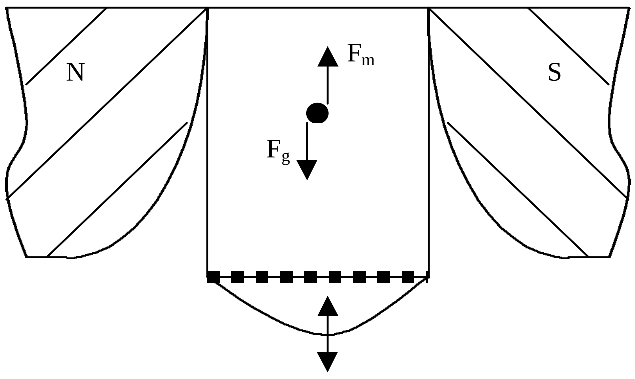


Figure 2. 2 The schematic of a magnetic jig

2.2.3 Centrifugal Jig

The centrifugal jig combines two separation methods together: jigging and centrifuging. The centrifugal jig can be dry or wet jigging according to the medium used (Laser, U. & Wagener, W. 1994; Beniuk, V.G., Vadeikis, C.A. & Enraght-Moony, J.N. 1994; Campbell & Thomas, 1991). Generally, the centrifugal jig operates according to the same principles as conventional jig except that the screen surface is cylindrical and is rotated to subject the particles to centrifugal forces. Figure 2.3 is a view of a centrifugal jig manufactured by Altair Technology.

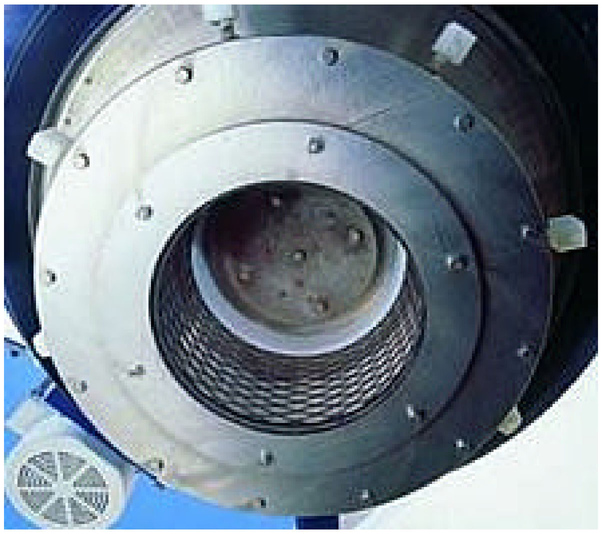


Figure 2. 3 View from the top of an Altair centrifugal jig

According to the design, a slurry feed is introduced into the top of the centrifugal jig. The slurry is diffused across the top of the interior of a rotating vertical cylindrical screen. Water is pulsed through the screen allowing differential separation in the slurry material. Heavy particles pass through the screen and are collected as “concentrate”.
Lighter particles flow down the screen interior and exit from the bottom of the machine as "tails". The Jig effectively drops the minimum particle size threshold for commercial

scale gravity recovery. Some tests for processing gold placer, coal, zircon, and titanium ore showed that a higher recovery and grade could be attained by this jig.

2.2.4 Inline pressure Jig

The so-called “Inline Pressure Jig” (Gray, A.H., 1997) is the encapsulation of jig bed within a closed vessel, allowing the equipment to operate under pressure. This innovation eliminates free water surface within the jig and may improve the particle stratification process. A higher recovery and upgrade ratio in separation of minerals was reported.

2.3 Introduction To The Packed Column Jig

2.3.1The Invention Of Packed Column Jig

Packed column jig was patented by Yang in 1996 (Yang, 1996). It was said that the invention was made when the inventor ran an iron ore flotation using packed flotation column (another invention of the inventor). During that packed flotation test, the froth depth was very low and the test was supposed to be bad. But the analyzed results showed a good separation. Then, the inventor tried to figure out why the separation happened and found that it was a gravitational separation in the packed flotation column. The packed column jig was invented with adding a pulsating device to the packed flotation column. It was devised to process fine particles, and it is especially useful for the separation of small silica particles from small iron mineral particles, for example, upgrading the low-grade, magnetic taconite ores.

2.3.2 The Structure Of Packed Column Jig

Packed column jig is a gravitational separation device constructed with a packed column and a device for vibrating the packed column and the particles. It allows for efficient and effective separation of solid particles having different densities.

The packed column is filled with corrugated diagonal packing. The packing materials in the jig divide the column into a large number of virtual cells. The feed of slurry is fed into the middle of the column. Water enters into the bottom as both a steady state and a pulsating state flow. The pulsating device sets up the jigging action in each virtual cell, segregating particles in the jig. The dense particles will exit from the bottom of the jig while the light particles will rise and exit from the top of the jig.

Because of the characteristics of the packing, the packed column jig reduces the flow vorticity and swirl size in the jig. The packing materials form a large number of small dimensional virtual jigging cells, which make pulp distributed evenly within the packed column. Each virtual cell has the similar shape and separation performance. The whole packed column jig can be considered to be a network of these virtual cells. The separation effect of the packed column jig is the sum of the effects of all these virtual cells. When parameters of the packed column jig are properly set, the particle separation in each virtual cell can be refined to give a very good separation performance for certain ore samples.

Figure 2.4 is the schematic design of the packed column jig (Yang Jig).

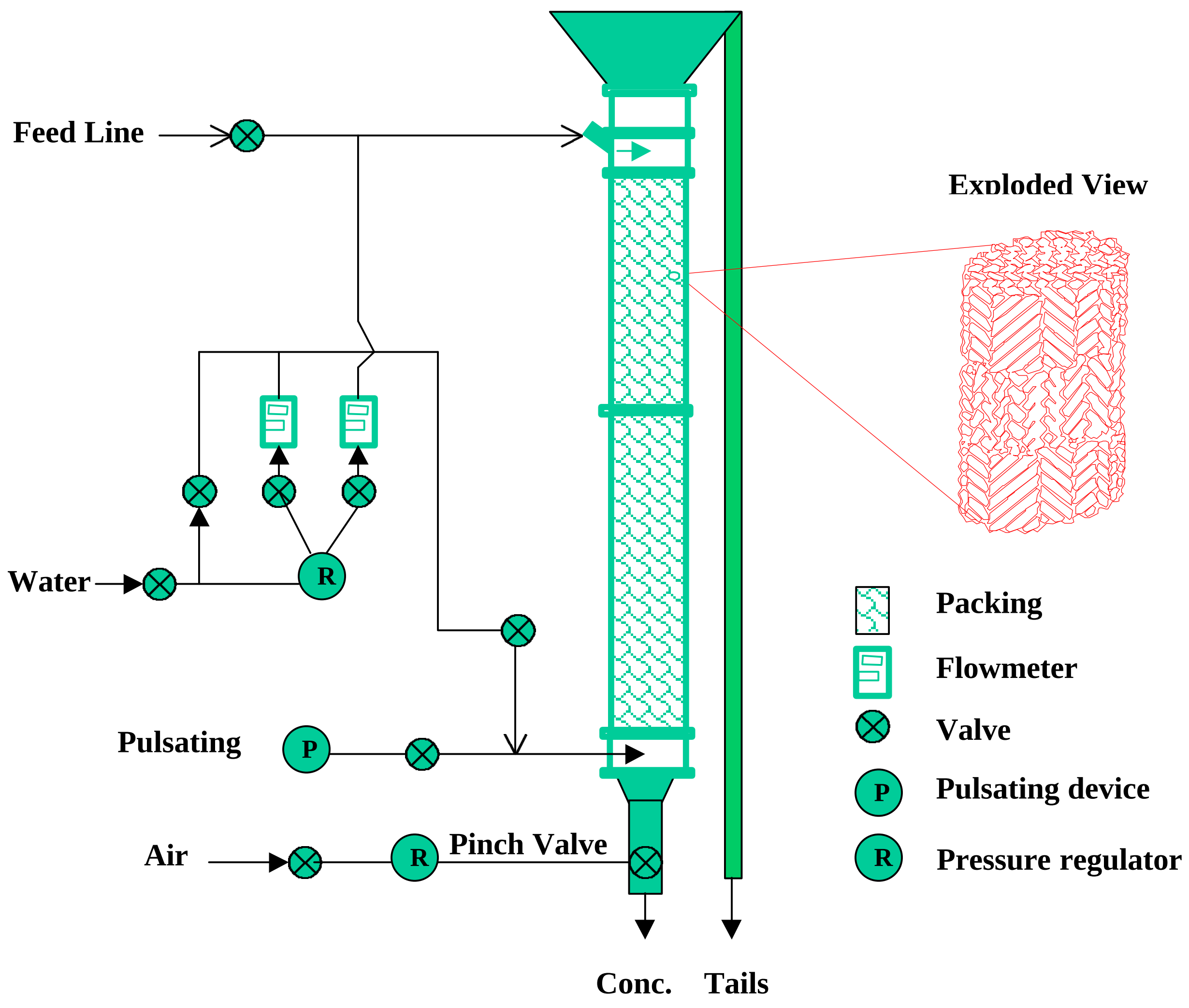


Figure 2. 4 The schematic design of packed column jig

In the separation process by the packed column jig, there are a number of parameters that affect the performance of the separation. The main parameters are listed as follows:

- Jig geometry

  Height, section number of the jig;

  Height, width, length and slope of each virtual cell;

The feed point.

- Ore properties

  The wt % of each mineral in the ore;

  The particle size distribution of each mineral;

  The density of each mineral;

  The magnetic susceptibility of each mineral;

- Operation conditions

  Pulsating frequency;

  Pulsating amplitude;

  Pulsating curve;

  Feed rate;

  Net velocity of upward flow;

  wt % of the feed;

  Solid packing density in the column.

- Fluid dynamic factors

  Stream velocity distribution in a virtual cell.

- Random factors

Random mechanical resistance to particles and random interference to the packed column jig system.

In order to understand the effects of all these parameters to the separation of the particles by actual laboratory tests is time consuming and difficult to perform for some parameters. But they are important for the improvement and application of packed column jig technology. This is the reason for this jig simulation work.

2.3.3 The Performance of the Packed Column Jig

The packed column jig has been tested for various ores. They are iron ore, coal, chromite, titanium ores and so on. A number of good separation results were attained in the laboratory tests. The scale-up tests for chromite, iron ore are preparing to carry out. Some of the test results are listed in Table 2.1 & 2.2 (Yang, 1996, 1997).

Table 2. 1 Packed column jigging result for processing coal

Test	Ash Content %			CMR	Capacity
No.	-16+100M Coal Conc.	-16+100M Refuse	Calc. Feed	%	T/hr/ft^2
1	6.8	80.8	43.1	83.4	0.38
2	6.4	81.5	42.8	84.3	0.47
3	6.1	76.6	37.8	83.1	0.63
4	6.7	79.4	37.2	86.3	0.74
5	7.6	71.5	42.0	73.6	0.72
6	6.7	71.8	39.3	76.7	0.86
7	6.7	78.7	38.6	84.7	0.66
8	6.2	77.4	42.5	79.9	0.53

As can be seen from Table 2.1, the jig can effectively reject the ash minerals and produce low ash coal concentrate just by combining the screening with the jigging. Because it is a gravitational separation, the pyrite is going to be rejected into the ash product more easily as comparing with the flotation method.

Table 2. 2 Packed column jig performance for taconite ore

Sample	Product	%wt	%Fe	%SiO_2	%Fe Dist.
Magnetic taconite crude from Mine A 85%-325 mesh	Conc.	34.87	67.93	4.02	76.68
	Tail	65.13	11.06	67.50	23.32
	Calc.Head	100.00	30.89	45.36	100.00
	Conc.	34.81	70.51	1.20	76.20
	Tail	65.19	11.33	67.29	23.80
	Calc.Head	100.00	31.03	45.28	100.00
*Magnetic taconite crude from Mine B 80%-325 mesh	Conc.	33.31	69.88	1.82	68.45
	Tail	66.69	16.09	66.65	31.55
	Calc.Head	100.00	34.01	45.19	100.00
	Conc.	36.53	67.57	4.38	70.96
	Tail	63.47	15.91	67.19	29.04
	Calc.Head	100.00	34.78	43.81	100.00
**Magnetic taconite conc. from Mine B 80%-325 mesh	Conc.	87.24	69.28	2.23	97.20
	Tail	12.76	13.65	69.14	2.80
	Calc.Head	100.00	62.18	10.77	100.00
	Conc.	87.96	68.83	3.10	98.04
	Tail	12.04	10.05	72.23	1.96
	Calc.Head	100.0	61.75	11.42	100.00

*Plant data (magnetic separation + reverse flotation) is:
Concentrate with %Fe 66.3, %SiO_2 5.4 and %Fe Dist. 58.5.

**Plant data (reverse flotation) is:
Concentrate with %Fe 66.3, %SiO_2 5.4 and %Fe Dist. 85.4.

As can be seen from Table 2.2, the jig can produce a very high iron grade of concentrate with SiO_2 less than 2%. The separation result of packed column jig for these magnetic taconite ore samples is much better than that of conventional magnetic separation or flotation.

Chapter 3 Modeling The Packed Column Jig

In order to scale up the packed column jig, it is necessary to develop a mathematical model for the jig. The packed column jig is different in design and operation from the conventional jig. For setting up the model for packed column jigging process, a simple review of the approaches in modeling the jigging processes will be followed by the force analysis on particle in packed column jig.

3.1 Different Approaches for Modeling The Jigging Processes

Although the jigging has been a well-known operation in mineral processing for more than a century, a fully satisfying theoretical interpretation of the segregation process is still not available (Schubert, 1994).

In recently years, the research for jigging is revitalized. There are mainly two approaches in modeling jigging processes.

1) **Microscopic model:** The microscopic model is the one developed in terms of micro-processes of jigging process. This kind of model is usually based on the particle movement in the jigging process. There are two ways for developing such a model: one is by theoretical derivation and the other is by testing. By detecting the trajectory of single particle, we can develop an empirical model to relate the movement of particles in jig bed to various parameters. But the detecting of single particle is never easy to perform. By analyzing the forces exerted on a particle in a jig, a particle movement model based on the Newtonian mechanics can be developed. By solving the model,

the performance of the jig process can be illustrated. Lin et al (1997) developed a particle movement model for conventional jig and magnetic jig with a new drag force expression, which enabled the model to work well in a quite wide range of Reynolds number. Mishra et al (1998) developed a model explaining particle stratification in jigs. The motion of solid particles is computed using the discrete element method by incorporating the buoyancy and drag forces on them. The accuracy of the force expressions is vital in this kind of modeling. The simulation of actual jigging process by using this discrete model is still difficult for there is so many particles in a jigging system.

2) **Macroscopic model:** The macroscopic model describes the overall performance of the jigging process and usually is an empirical model. It has different forms depending on the developer's concern and the method used.

   Tucker (1995) developed an empirical model of a centrifugal jig. The model describes the partition of feed material to the jig concentrate, taking account of the main jig operating parameters: g-force, ragging characteristics and stroke length and material properties of the feed. Another model based on the potential energy minimization principle allowing for the disperse effects due to particle-particle and particle-fluid interactions was developed and applied to the jigging of coal (Tavares, 1995). This model requires only a single parameter, the specific stratification constant. Also this model generates the equilibrium stratification profile for all components in the jig bed and the separation of each component is then calculated as a function of the total yield of solids. Li (1993) modeled jig process as a Markov stochastic process in order to estimate and predict the optimum production performance of a jig,. The probability vectors in the transition matrix were supposed to be non-stationary and could be determined from actual production data. Meloy (1997) modeled packed column jig as particles of different densities flowing within the packing by a stochastic model. There are three variables in the model: upward velocity of the stream, particles settling velocity and degree of turbulence. The yield probabilities of dense particles and light particles along the cells from top to bottom were calculated and related to the operation parameters.

While the microscopic model is more basic and the macroscopic model is more specific, each of these kinds of models has its own drawbacks in modeling the jigging process. For the microscopic model, the determination of the forces always is difficult, especially when the voidage in the jig is lower (or the solid volume concentrate is high). Thus, the interpretation of the model often can not derive the correct and accurate conclusion. For macroscopic model, its interpretation hardly can tell the details of the process and then reducing its usefulness.

The knowledge about the packed column jig is still limited. The similar work was also limited to one paper (Mackley et al. 1993) dealing with the mixing and separation of particle suspensions by using oscillatory flow in baffled tubes. In this study, a model of packed column jig is derived from theoretical consideration: a microscopic approach.

3.2 Force Analysis of Particle in Packed Column Jig

In conventional jig, there are two actions, fluidization and stratification of particles, which determine the performance of the jig. In one cycle of pulsating, only the period when the jig bed is dilated produces the stratification. The packed column jig has some difference from the conventional jig in that the particles keep well fluidized all the time in packed column jig. The stratification happens any time in the packed column, which may bring higher efficiency. The particle movement in this case is easier to be modeled for the more even fluidization during whole pulsating cycle and the forces are easier to be expressed. The microscopic model could be more suitable for describing the jigging process in the packed column jig. For developing such a model, forces exerted on a particle in packed column jig are discussed below.

1) **Gravity force**. Every particle is subject to the gravity force. The force can be expressed as:

$$G = \frac{1}{6}\pi d^3 \rho_s g \qquad (3.2\text{-}1)$$

2) **Buoyant force**. The force can be expressed as:

$$F_f = \frac{1}{6}\pi d^3 \rho g \qquad (3.2\text{-}2)$$

3) **Drag forces**. There are two kinds of drag force for each particle. One is from the velocity difference with the stream and the other is from the speeding of the adjacent fluid. Usually the first one is dominated when dealing with the particle moving in a static stream, but in the jigging process, the drag force from acceleration of adjacent fluid can not be neglected due to the pulsating stream.

A) ***Drag force from velocity difference between a particle and the fluid***. This can be expressed as:

$$F_{D1} = \pm\psi d^2 (v - u)^2 \rho \qquad (3.2\text{-}3)$$

The direction of F_{D1} is opposite to the direction of (v-u). ψ is dependent on the Reynolds number in the packed column jig. Generally, the drag force can be calculated by the following two equations according to the Reynolds number.

$$F_{D1} = -3\pi\mu d(v - u) \quad \text{for Re<5, (Stokes Equation)} \qquad (3.22\text{-}4a)$$

$$F_{D1} = \pm\left(\frac{\pi}{16} \sim \frac{\pi}{20}\right) d^2 (v - u)^2 \rho \quad \text{for Re>1000, (Newton-Rittinger Equation)}$$

(3.22-4b)

The Reynolds number in the packed column jig is usually greater than 1000. The Newton-Rittinger Equation can be used in this simulation.

B) ***Drag force from accelerating the adjacent fluid to a particle***. When the particle accelerates, the adjacent fluid will be accelerated too. The force for accelerating the fluid will react to the particle as a drag force. This force can be expressed as:

$$F_{D2} = \frac{1}{6}\xi\pi d^3 \rho \frac{dv}{dt} \qquad (3.2\text{-}5)$$

Where, ξ is a mass combination coefficient, which is the ratio of the volume of fluid moving with the particle to the volume of particle. In this simulation, a value of 0.05 is assigned to it.

4) **Fluid pressure force due to acceleration**. When the fluid accelerates, it will exert a pressure force to the particle in the medium. This force can be calculated by the following equation.

$$F_p = \frac{1}{6}\pi d^3 \rho \frac{du}{dt} \tag{3.2-6}$$

5) **Shear dissipation force.** Although the stream movement in different virtual cells can be accounted to be the same, the stream velocity across the section of each cell is truly changeable. There exists a shear between two adjacent layers of stream. The shear will generate a dissipation pressure between two adjacent layers of stream according to Bagnold (1954). The shear between two adjacent layers can be estimated by the following equation.

$$T_{in} = f(\lambda)\overline{\rho}_S (\overline{d})^2 \left(\frac{du}{dy}\right)^2 \cos\theta \tag{3.2-7}$$

More specifically,

$$T_{in} = 0.0013\overline{\rho}_S (C\overline{d})^2 \left(\frac{du}{dy}\right)^2 \tag{3.2-8a}$$

With

$$C \approx \frac{1}{\left(\frac{\lambda_0}{\lambda}\right)^{1/3} - 1} \tag{3.2-8b}$$

where C is the particle linear concentration, which is the ratio of average particle diameter to the average distance between two particles.

The dissipation pressure is proportional to the shear strength, which can be estimated by the following equation.

$$\frac{T_{in}}{P} = 0.32 \sim 0.4 \tag{3.2-8c}$$

The net force by the shear on particle then can be calculated by the following equation:

$$F_B = \frac{k_B}{4}\pi d^2 \Delta P \tag{3.2-9}$$

6) **Mechanical resistance force by interaction of particles**. The particles will interact with each other while jigging in the packed column jig. Then the

mechanical resistance force is generated for each particle. It is difficult to figure out the amplitude of the force and its direction. This is a random force. In this modeling, a random force with relation to the particle size, particle velocity, the volume concentration of particles in pulp and the average particle density is assigned to each particle.

$$F_R = \frac{k_m}{4}\pi d^2 \lambda \bar{\rho} v \Delta v \qquad (\mathbf{3}.2\text{-}10)$$

where λ is the volume concentration of particles in the pulp.

K_m is a constant of particle momentum transfer coefficient. The value of k_m is related to the collision characteristics among particles. Assume k_m=1 in this simulation.

Δv is the average relative velocity difference between the interested particle and other particles. Δv is related to many parameters, such as particle size, pulsating conditions, stream velocity,…etc. The determination of it is so complicated that it is beyond the scope of this thesis. In this simulation, the following equation is assumed to estimate it.

$$\Delta v = \pm \text{random}(\Delta v_m) \qquad (3.2\text{-}11)$$

where Δv_m is the maximum time-average velocity difference that is assumed to be equal to the maximum pulsating velocity in the simulation.

7) **Magnetic force.** In the case of a magnetic feed with feed magnetizing, there exists magnetic flocculation. A magnetic particle will be exerted a magnetic force by other magnetic particles adjacent to it at the vertical direction. In this simulation, this magnetic force is assumed to be directly proportional to the magnetic susceptibility and the volume of the particle.

$$F_M = \frac{k_s}{6}\pi d^3 \chi \tag{3.2-12}$$

3.3 The Particle Movement Model In Packed Column Jig

All these forces exerted on the particles in packed column jig can be used in the force balance equation as given below:

$$m\frac{dv}{dt} = G - F_f \quad F_{D1} - F_{D2} + F_p + F_B - F_R + F_M \tag{3.3-1}$$

By substituting all the expressions of the forces into the above equation, the following momentum equations for a particle moving in a packed column jig can be obtained.

At X direction (From left to right at horizontal direction):

$$\frac{dv_x}{dt} = \pm\frac{6\psi(v_x - u_x)^2\rho}{\pi d(\rho_s + \xi\rho)} + \left(\frac{\rho}{\rho_s + \xi\rho}\right)\frac{du_x}{dt} + \frac{3k_m\bar{\rho}\lambda v_x\left(\pm\,\text{random}(\Delta v_m)\right)}{2d(\rho_s + \xi\rho)} + \frac{3k_B\Delta P_x}{2d(\rho_s + \xi\rho)} \tag{3.3-2}$$

At Y direction (From top to bottom at vertical direction):

$$\frac{dv_y}{dt} = \frac{(\rho_s - \rho)g}{\rho_s + \xi\rho} \pm \frac{6\psi(v_y - u_y)^2\rho}{\pi d(\rho_s + \xi\rho)} + \left(\frac{\rho}{\rho_s + \xi\rho}\right)\frac{du_y}{dt}$$
$$+ \frac{3k_m\bar{\rho}\lambda v_y\left(\pm\,\text{random}(\Delta v_m)\right)}{2d(\rho_s + \xi\rho)} + \frac{3k_B\Delta P_y}{2d(\rho_s + \xi\rho)} + \frac{k_s\chi}{(\rho_s + \xi\rho)} \tag{3.3-3}$$

At Z direction (From backward to forward at horizontal direction):

$$\frac{dv_z}{dt} = \pm\frac{6\psi(v_z - u_z)^2\rho}{\pi d(\rho_s + \xi\rho)} + \left(\frac{\rho}{\rho_s + \xi\rho}\right)\frac{du_z}{dt} + \frac{3k_m\bar{\rho}\lambda v_z\left(\pm\,\text{random}(\Delta v_m)\right)}{2d(\rho_s + \xi\rho)} + \frac{3k_B\Delta P_z}{2d(\rho_s + \xi\rho)} \tag{3.3-4}$$

3.4 Determination of Parameters in Packed Column Jig

For computing the above Equation (3.3-2,3,4), some parameters need to be determined or related to the operation conditions.

3.4.1 The Fluid Velocity Distribution In The Virtual Cell Of Jig Column.

In turbulent fluid dynamics, we know that the velocity distribution across the section of a pipe has the following equation.

$$\frac{\bar{u}}{u_c} = \left(1 - \frac{r}{R}\right)^{1/n} \tag{3.4-1}$$

with n=6-10, generally n=7.

u_c is the centerline stream velocity.

The above equation can be adopted for computing the fluid velocity profile across the section of a virtual cell in packed column jig with R being the maximum distance from the centerline of the virtual cell and r being the distance of interested point from the centerline. Also, we need to estimate the turbulent intensity in the virtual cell of packed column jig. According to the definition of turbulent intensity (ζ),

$$\zeta = \frac{\sqrt{\overline{(u')^2}}}{\bar{u}} = \frac{\left[\frac{1}{T}\int_{t_0}^{t_0+T}(u')^2 dt\right]^{1/2}}{\bar{u}} \tag{3.4-2}$$

u' is the fluctuating part of the velocity.

For the flow in the atmosphere and rivers, the turbulent intensity is usually greater than 0.1. In the packed column jig, the turbulence is relative low as compared with that in the rivers. According to the theory of the turbulence (Davies, 1972), the turbulent intensity can be assumed to be equal to 0.04 in packed column jig. Thus in the simulation, a fluctuating of velocity with ζ=0.04 is taken into consideration.

Then at a certain point, the velocity of the stream can be expressed as:

$$u = \bar{u} + u' = \bar{u} \pm (3\zeta)\mathrm{Random}(\bar{u}) \tag{3.4-3}$$

In this simulation, a pulsating with harmonic wave is assumed.

Then,

$$\bar{u} = \left(1 - \frac{r}{R}\right)^{1/n} u_c = \left(1 - \frac{r}{R}\right)^{1/n} \frac{(n+1)(2n+1)}{2n^2}\left(u_{up} + 2\pi\nu l \sin(2\pi\nu t)\right) \tag{3.4-4}$$

3.4.2 The Average Density And Volume Concentrate Of Solid In Virtual Cell

The particle separation in the packed column jig produces various average particle densities as a function of height inside the jig. Here, a parameter of the feasibility of gravity separation (ϕ) in packed column jig is defined as the one dependent on the density, size and some operation conditions of the packed column jig.

$$\phi = f\left(d_1, d_2\rho_{s1}, \rho_{s2}, \nu, l, \bar{u}, \ldots\right) \tag{3.4-5}$$

A larger value of ϕ means that the separation of particles is more feasible.

The distribution of average particle density along the height then can be estimated with the following empirical equations.

$$\bar{\rho}_s = \rho_{s2} - \left(\frac{h}{H_1}\right)^{\phi}\left(\rho_{s2} - \frac{\rho_{s1}\rho_{s2}}{\gamma_2\rho_{s1} + \gamma_1\rho_{s2}}\right) \quad \text{when } h<H_1 \tag{3.4-6}$$

$$\bar{\rho}_s = \frac{\rho_{s1}\rho_{s2}}{\gamma_2\rho_{s1} + \gamma_1\rho_{s2}} - \left(\frac{h - H_1}{H_2 - H_1}\right)^{\phi}\left(\frac{\rho_{s1}\rho_{s2}}{\gamma_2\rho_{s1} + \gamma_1\rho_{s2}} - \rho_{s1}\right) \quad \text{when } H_1<h<H_2 \tag{3.4-7}$$

H_1 is the height of feed point.

The average particle density in each virtual cell is:

$$\bar{\rho}_{si} = \frac{\int_{h_i}^{h_{i+1}} \rho_s dh}{h_{i+1} - h_i} \approx \frac{\rho_{s(h=h_{i+1})} + \rho_{s(h=h_i)}}{2} \quad (3.4\text{-}8)$$

From the bottom to the top, the average particle density decreases. The gradient of this density change depends on the separation feasibility of the feed particles.

For computing the solid volume concentration of each cell, the following assumptions are made.

- The average particle size in each cell is approximately the same.
- The particle dispersion in each cell is uniform.

Then, the average terminal settling velocity of particles of a virtual cell can be determined by the following equation.

$$\bar{v}_m = \sqrt{\frac{\pi(\bar{\rho}_{si} - \rho_w)\bar{d}g}{6\psi\rho_w}} \quad (3.4\text{-}9)$$

The solid volume concentration λ in a virtual cell then can be related to the average superficial velocity $\bar{v}$ of stream by the following equation.

$$\frac{\bar{v}}{\bar{v}_m} = (1-\lambda)^{n'} \quad (3.4\text{-}10)$$

n' is a constant related to the Reynolds number.

In the packed column jig with Reynolds number usually being greater than 500, n' can take the value of 2.65 according to Richardson and Zaki (1954).

In the packed column jig, the $\bar{v}_m$ decreases from bottom to top, and the $\bar{v}$ also decreases from the bottom to top especially on the upper section. The solid volume concentration actually increases slightly from the bottom to the top.

In this simulation, the multiply of $\overline{\rho}_s\lambda$ is assumed to the unchangeable for all virtual cells of the packed column jig.

3.4.3 The Pulp Density In Each Virtual Cell

When particles are dispersed in medium, it will change the performance of the medium. For fine particles, the particles can form a suspension in the medium. This is the case in the packed column jig system. The pulp density in the packed column is supposed to be the density of the suspension. It is dependent on the density of the particles and the particle packing density in the medium.

$$\rho = \lambda(\overline{\rho}_s - \rho_w) + \rho_w \tag{3.4-11}$$

From the top to the bottom, the average particle density increases while the volume concentration of the solid decreases. In each virtual cell of the packed column jig, the pulp density is slightly different but not so much. In this simulation, for simplifying the problem, the pulp densities in all cells of the packed column jig are assumed to be equal.

3.4.4 The Particle Size Distribution Of Feed

In this research work, the Gaudin-Schulman equation is used for example to express the particle size distribution of particles in the feed with the following equation.

$$y_+ = 100\left[1 - \left(\frac{x}{X_{Max}}\right)^{C_d}\right] \tag{3.4-12}$$

where

x is the particle size;

y_+ is the fraction by percent with particle size larger than x;

X_{max} is the maximum particle size;

C_d is a parameter for describing the size distribution. The larger the C_d value, the narrower the particle size distribution.

Chapter 4 Simulation of the Packed Column Jig

Simulation is the process to realize the model in the computer. There is no way we can realize every details of a complicated system, like the packed column jigging, in computer right now. The complicated process usually need to be simplified by make some assumptions before the simulation could be carried out. There were some simulations of jigging process (Beck et al., 1993; De Jong et al., 1996; Li et al., 1993; Mishra et al., 1998), but each simulation only gave a small view of the actual jigging process. In this thesis work, the simulation of packed column jig was based on the particle movement equations derived in Chapter 3.

4.1 Simulation Method and Algorithm

Figure 4.1 shows a block diagram of particle movement in packed column jig.

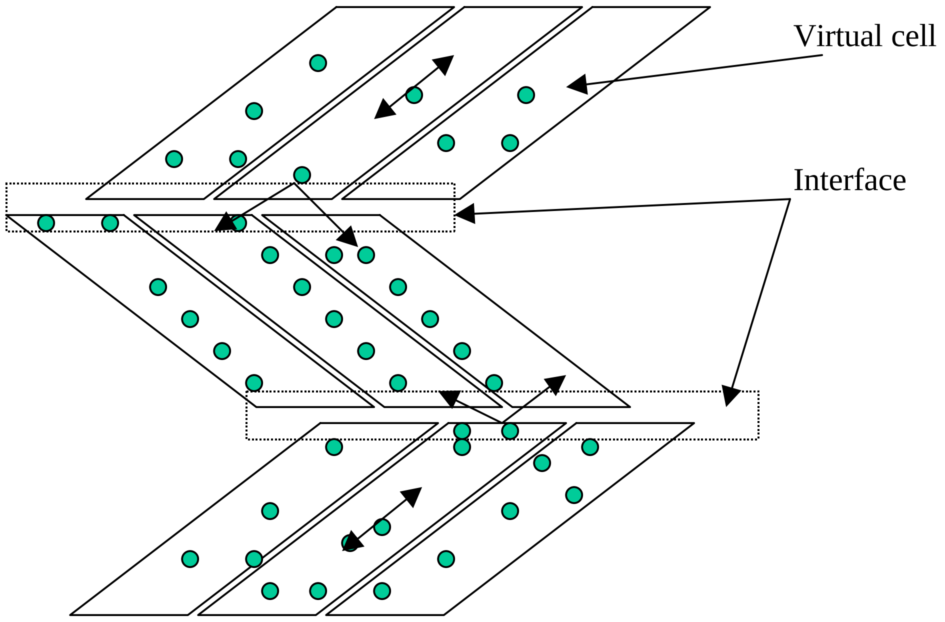


Figure 4. 1 Block diagram of particle movement in packed column jig

The particle moves in the virtual cell of packed column jig according to the momentum equation derived in Chapter 3. While on the interface during two layers of packing, the particles will distribute by a random action. The particle moving out a virtual cell can get into any of the adjacent virtual cells of the next layer. This random action makes the distribution of particles uniform in all the virtual cells at the same level. Then the cells in the same layer can be considered as to have the same separation conditions. In different layers, some conditions may different, such as the upward stream velocity, cell dimensions, …etc.

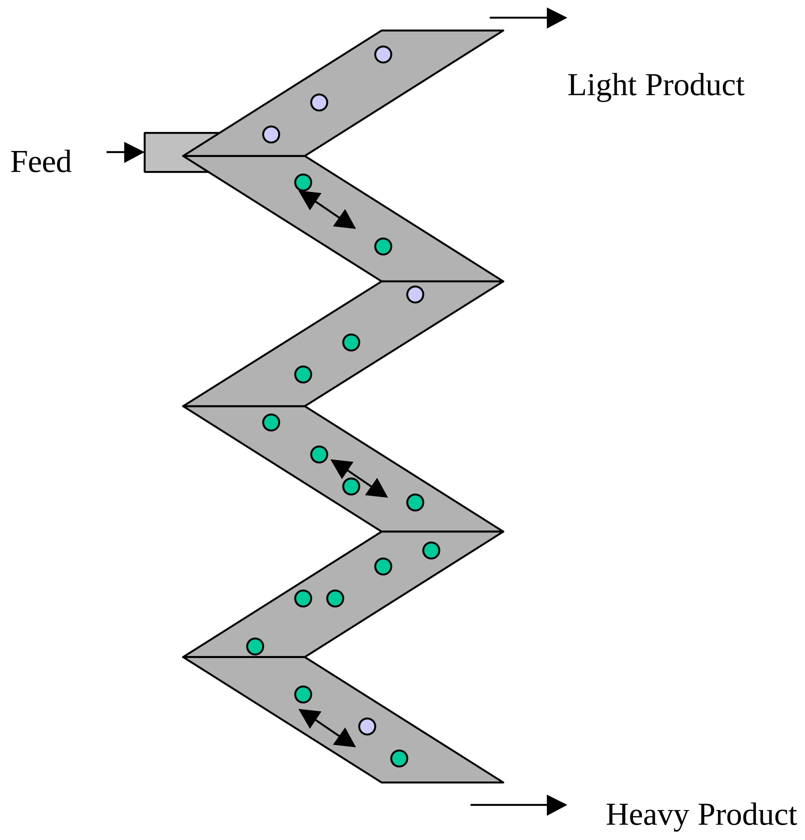


Figure 4. 2 Simplified physical model of packed column jig

After the jigging process in the packed column jig comes to a steady state, the virtual cells at each level are assumed to have the same condition and the same separation behavior. We can simulate the packed column jig by using one cell for each layer of cells.

Then, the simulation of packed column jig is simplified to the particle movement in a series of cells as shown in Figure 4.2.

In this simulation, the simulated particles are randomly generated. The momentum differential equation is set for each particle according to the derived model (3.3-2,3 4). Runga-Kutta method is used for solving the equation. It is a fourth-order method and it works much more rapidly than the modified Euler method.

Assuming we have the following differential equation,

$$\frac{dy}{dx} = f(x, y) \tag{4.1-1}$$

with the initial condition: $y(x_0) = y_0$

The computation formula of Runga-Kutta method for solving the above differential equation is following:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{4.1-2}$$

with :

$$k_1 = hf(x_n, y_n) \tag{4.1-3}$$

$$k_2 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) \tag{4.1-4}$$

$$k_3 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) \tag{4.1-5}$$

$$k_4 = hf(x_n + h, y + k_3) \tag{4.1-6}$$

By repeating to set and solve the equation for each particle, the particle is simulated to move in the packed column jig until it reaches the outputs or the simulation time exceeds the setting value. The simulated particles are counted by particle size, mineral type and the discharge output. After certain number of particles being simulated, the mineral distributions of each size range in concentrate (heavy product) and tailing (light product) are calculated. The total separation result is then calculated based on these distribution data as well as the particle size distributions of both minerals.

4.2 Introduction To The Simulation Program

A computer program developed in this thesis work for simulating the packed column jig includes three main parts as shown in Figure 4.3.

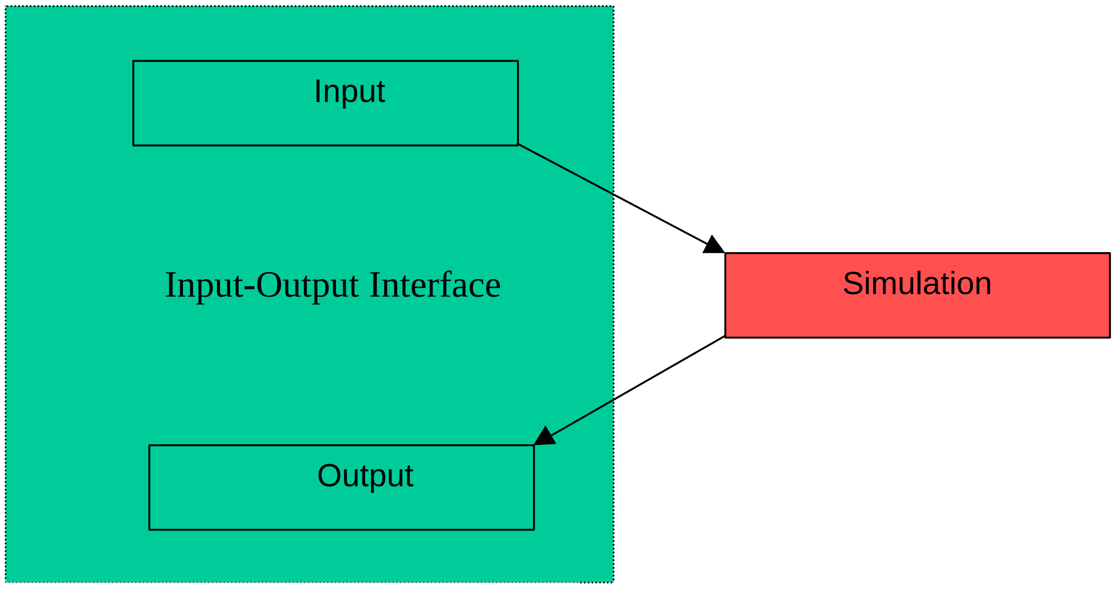


Figure 4. 3 The components of the simulation program

The first part is for input the operation conditions for the packed column jigging. The second part is the core of this simulation. It is developed for simulating particle movement in packed column jig at different conditions. The third part is for output the simulation results. The whole program can simulate the packed column jig performance for different ore, operation conditions as well as different geometry of the packed column jig. It can also be used for a visual simulation of particle movement in cells of packed column jig. For further consideration, it also can be developed for dynamic simulation of the packed column jigging process. The first and third parts are developed by using the Microsoft Visual Basic 5.0. They form a bundle and provide user-friendly interfaces for inputting simulation conditions and outputting the simulation results. The second part of simulation program is coded by using C/C++ to make the whole program more efficient and easy to be improved. The second part can either run separately and interacts with the input and output parts by files operation or integrated with the input/output parts by ActiveX technology in Microsoft Visual Basic environment.

Figure 4.4 shows the interface of the start of the program.

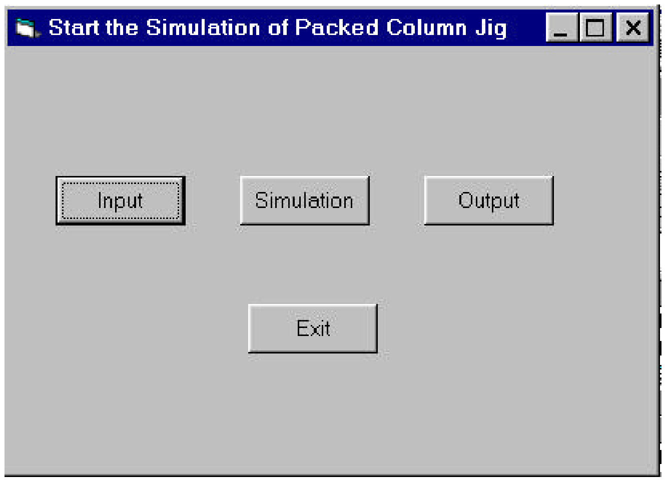


Figure 4. 4 Interface of the simulation start

There are four selections in this interface:

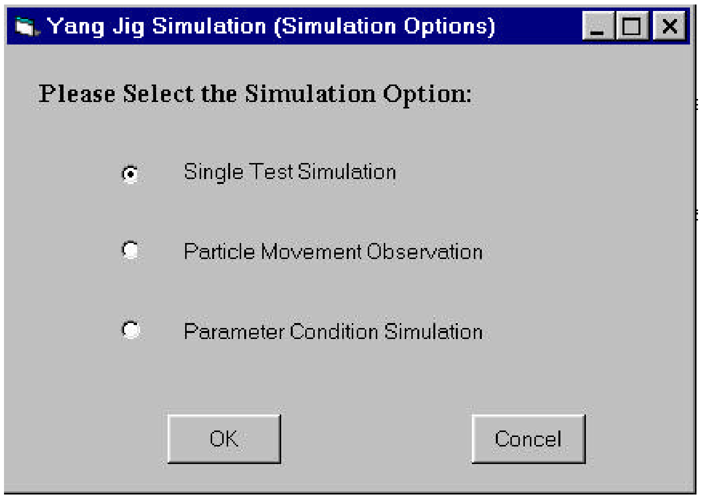
- Go to Input interface to create scenarios of packed column jigging.
- Go to the Simulation for getting the help for running the simulation file.

- Go to the Output to retrieve the simulation results.
- Exit the program.

4.2.1 Input Block of the Simulation Program

The Input file was designed to input different simulation conditions for different simulation purposes.

Figure 4.5 shows the first interface after enter the Input block.


Yang Jig Simulation (Simulation Options)

Please Select the Simulation Option:

[✓ Single Test Simulation]

Particle Movement Observation

Parameter Condition Simulation

Figure 4. 5 Simulation option interface

There are three different selections for going forward.

- Single Test Simulation. It provides you to define the simulation test conditions for just one test.
- Particle Movement Observation. It lets you input the jigging conditions in which you want to observe the particle movements in screen and set the particle size of the observed particles (only two particles, one heavy mineral particle and one light particle, are designed in the realization of the code.).

- Parameter Condition Simulation. It goes to the next interface to select the interested parameter for condition simulation test. There are a number of selections, which are shown in Figure 4.6.

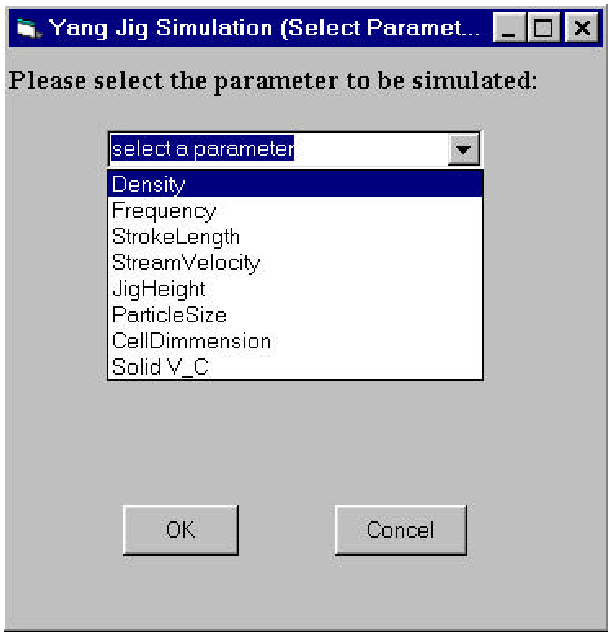


Figure 4. 6 Parameter selection interface

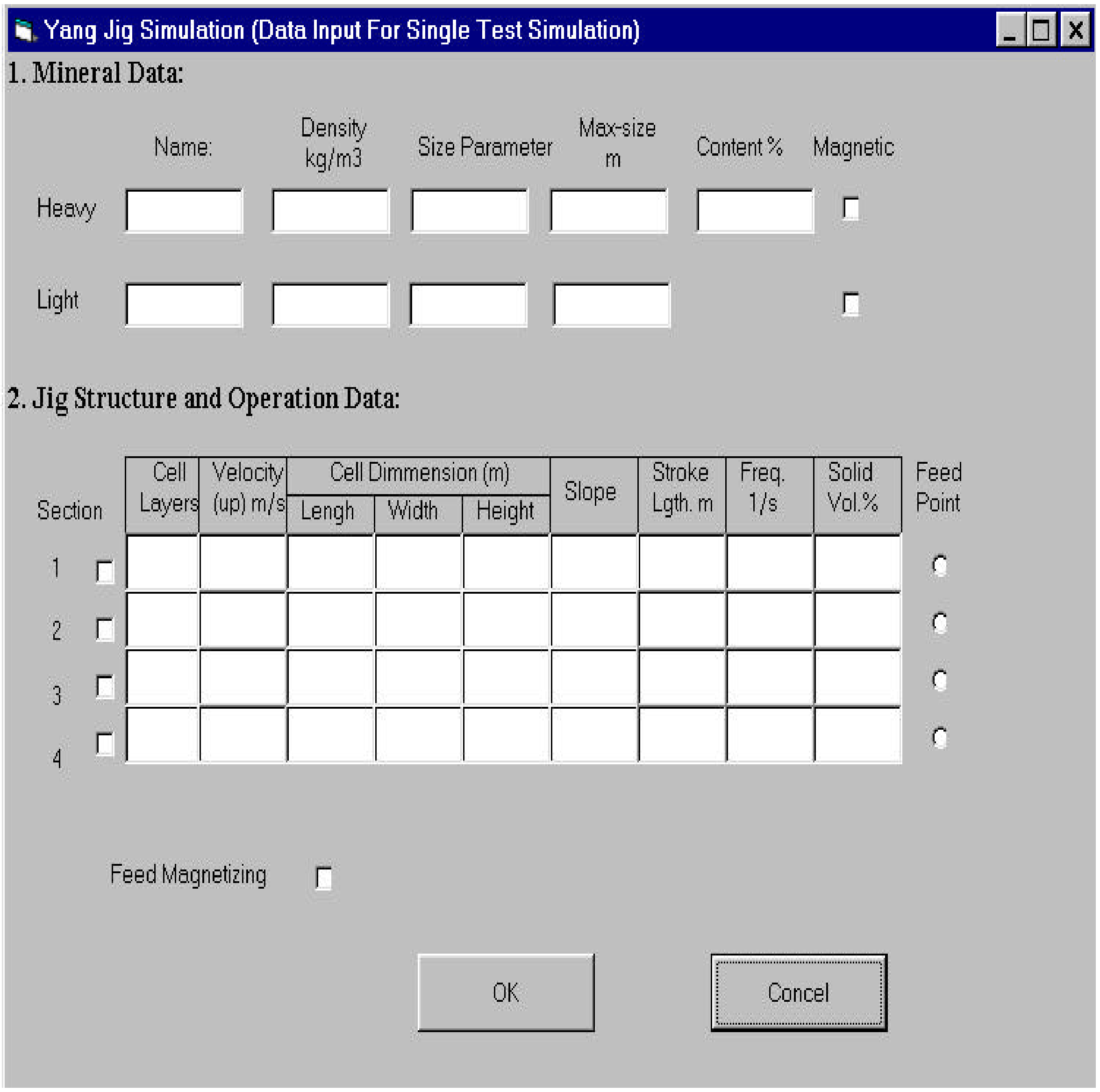
The input block considered the various parameters for condition simulation test which include cell dimensions, mineral density, particle size, stream velocity, pulsating frequency, stroke length, solid packing concentration.

The final input interfaces as an example shown in Figure 4.7 give the opportunities to define every detail for the packed column jigging process, which include the following.

- Mineral properties: mineral name, density, size distribution parameter, maximum particle size, content of heavy mineral and magnetic property.

- Jig geometry and operation conditions: section number, vertical cell number of each section, cell dimensions, stream velocity, pulsating frequency, stroke length, solid packing density, feed point, … etc.

All the input interfaces are easy to be understood and operated. The input data are automatically stored in files for running the core simulation program.



Yang Jig Simulation (Data Input For Single Test Simulation)

1. Mineral Data:

	Name:	Density kg/m3	Size Parameter	Max-size m	Content %	Magnetic
Heavy						☐
Light						☐

2. Jig Structure and Operation Data:

Section		Cell Layers	Velocity (up) m/s	Cell Dimmension (m)			Slope	Stroke Lgth. m	Freq. 1/s	Solid Vol.%	Feed Point
				Lengh	Width	Height					
1	☐										○
2	☐										○
3	☐										○
4	☐										○

Feed Magnetizing ☐

Figure 4. 7 Final interfaces for input of the jigging conditions

The Structure diagram of the input file is shown in Figure 4.8. The detailed code of the input file is listed in the appendix.

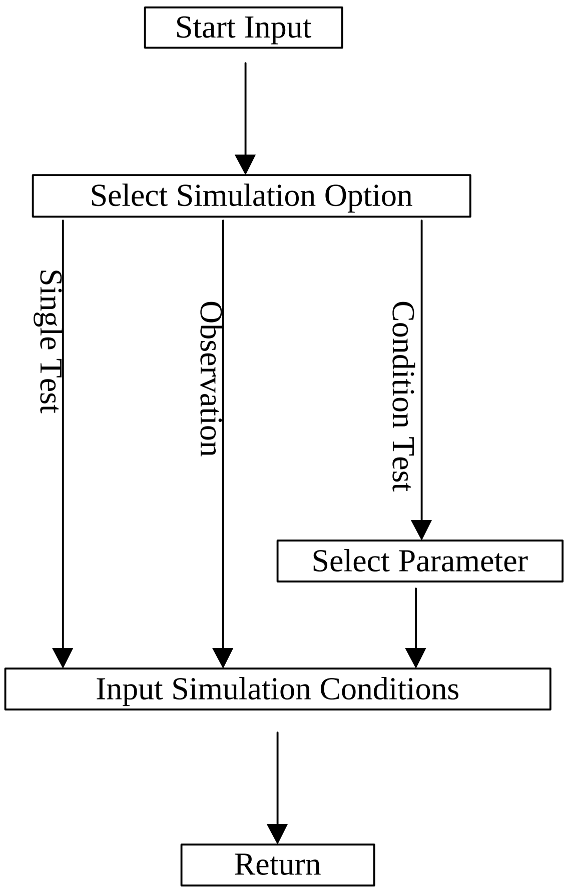


Figure 4. 8 The structure diagram of input file

4.2.2 Simulation Block of the Simulation Program

As mentioned above, this block is coded in C/C++. A number of classes are defined and these makes the simulation program easy to be upgraded provided there is a need in the future.

The detailed code is listed in the appendix. It consists of three files, a Head (.H) file, a CPP file that accomplish the functions of the head file and an example main CPP file for simulating the packed column jig in different conditions by using the defined classes.

The basic structure of the main file is shown in Figure 4.9.

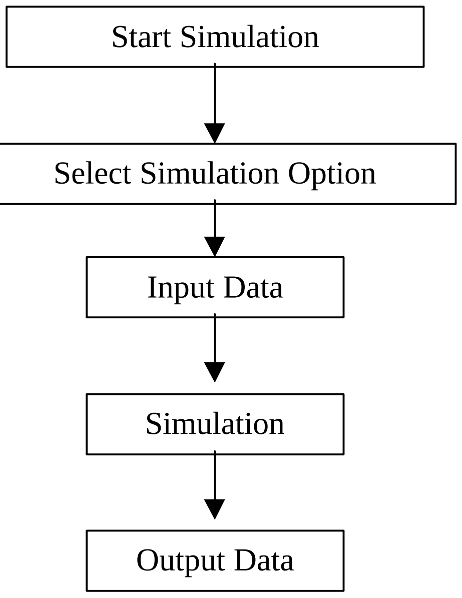


Figure 4. 9 The basic flowchart of the simulation main file

The particle movement simulation process is the core of this block. It includes a series of computations, decisions and transitions. The particle velocity and position are calculated based on the derived equations for each step by using the Runga-Kutta method. If the new position is beyond the cell boundary, its position is modified to the boundary of the cell. If the particle is above or below the present cell, the particle is transferred to the next cell by a random function which is simulating the mixing behavior on the interface of two adjacent cells in an actual packed column jig. These processes will continue until the particle is out of the final cells or the simulation time is larger than the provisional limit. The simulation process includes a number of random functions for generating particle, computing the velocity and acceleration of the stream and the mechanic resistance force. These random functions are favorable to obtain reasonable results. There are some defined functions in each class for showing or displaying the particle simulation message or image on screen. The final simulation result is stored in files, which can be retrieved by the output block of the simulation program. The basic flowchart of the particle movement simulation process is shown in Figure 4.10.

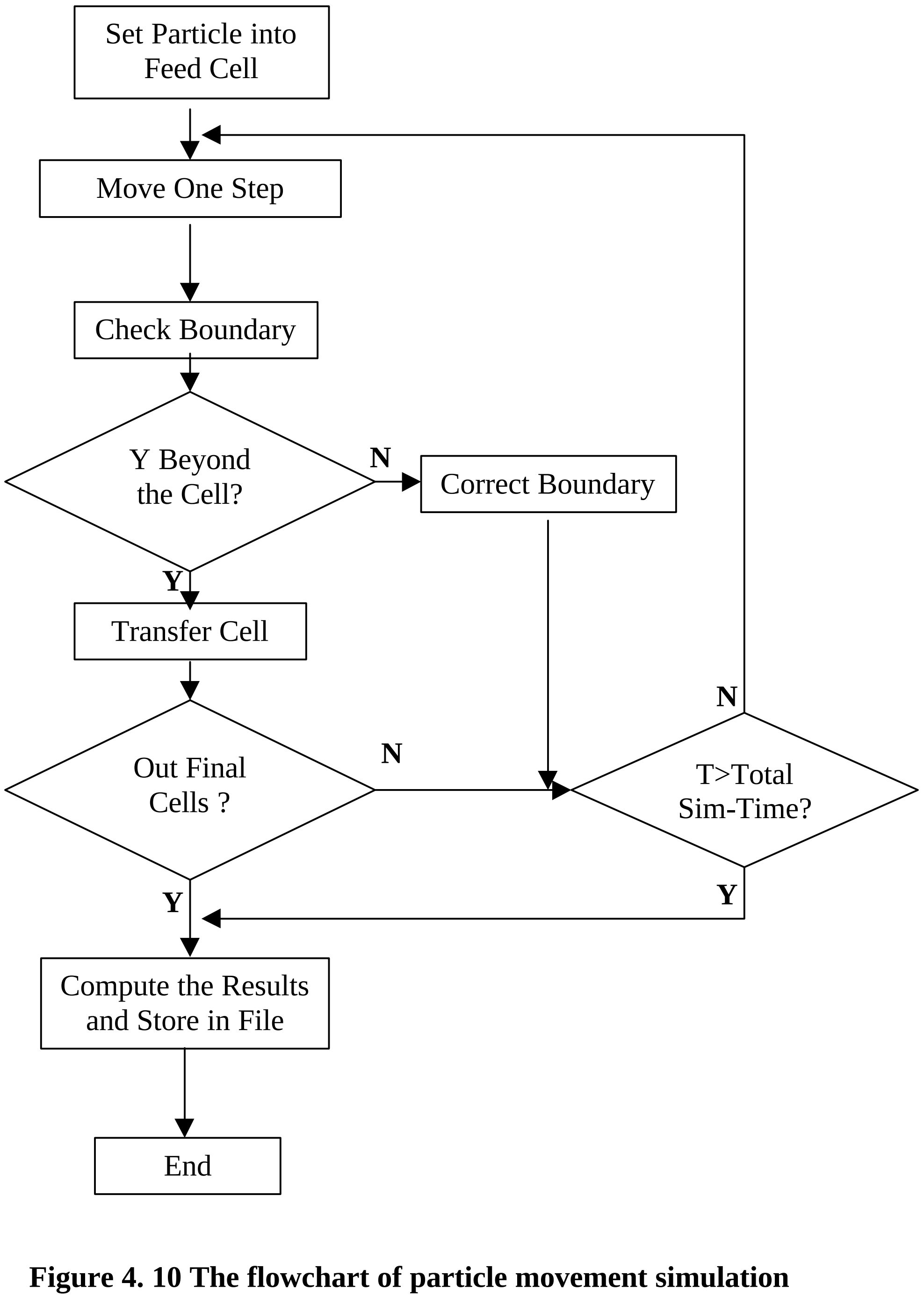


Figure 4. 10 The flowchart of particle movement simulation

4.2.3 Output Block of the Simulation Program

The output block is relatively simple. It gives the selections for which simulation result to be outputted for processing. It uses the ActiveX technology of Microsoft Visual Basic to dynamically make a link to Excel. There are user-defined event for automatically copying the simulation test results to the Excel sheet, the simulation results then can either be tabled or charted by using the functions of Excel. In the design of the simulation program, the single simulation test is detailed to 8 particle size ranges for separately consideration of the yield, grade and recovery of each size range in heavy product. The parameter condition simulation test only gives the total recovery and grade of the heavy product. Figure 4.11 is the interface of Output block and Figure 4.12 is an example of output display.

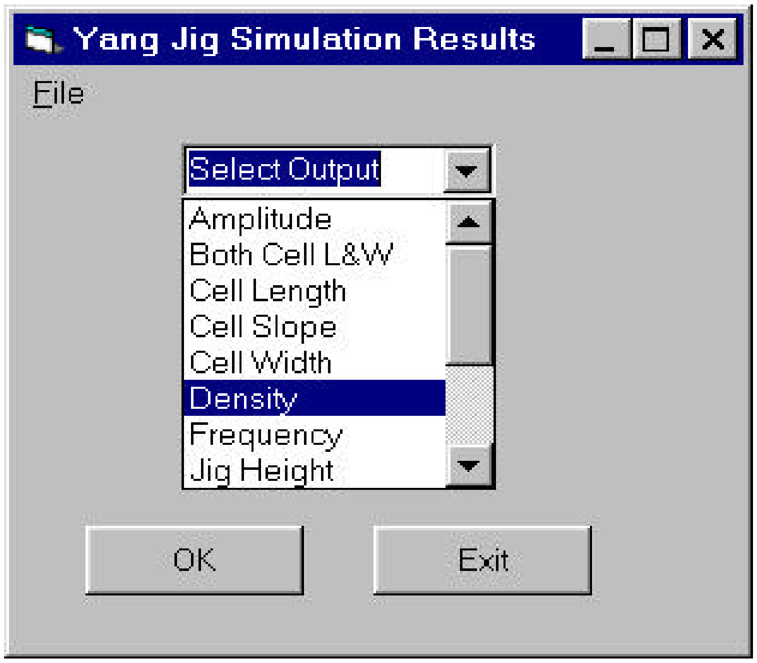
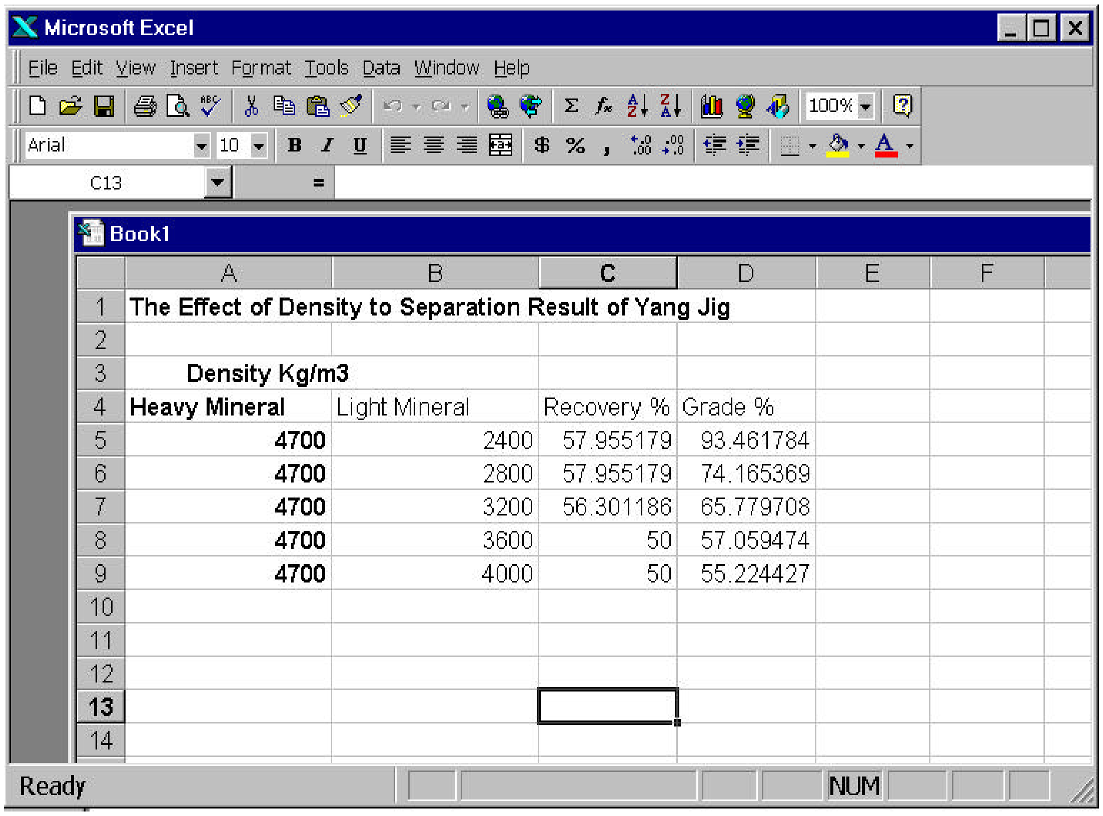


Figure 4. 11 The output interface of the simulation program



	A	B	C	D
1	The Effect of Density to Separation Result of Yang Jig			
2				
3	Density Kg/m3			
4	Heavy Mineral	Light Mineral	Recovery %	Grade %
5	4700	2400	57.955179	93.461784
6	4700	2800	57.955179	74.165369
7	4700	3200	56.301186	65.779708
8	4700	3600	50	57.059474
9	4700	4000	50	55.224427

Figure 4. 12 An example of the output of packed column jig simulation

Chapter 5 Discussion of the Simulation Results

There are a lot of parameters involved in the packed column jigging process. The parameters are considered as possible as the knowledge available in this simulation. In the following, some simulation results are shown and the effects of various parameters are discussed with the simulation results.

5.1 The Separation Performance of Packed Column Jig

In the laboratory, a number of tests of packed column jig were carried out for different ores, and the results showed some promising separation performance. Table 5.1 shows a simulated separation result of packed column jig for a feed with wide size range shown in Table 5.2. The density of heavy mineral is 4700.0 kg/m^3, the density of the light mineral is 2700.0 kg/m^3. The flowsheet is shown in Figure 5.1.

Table 5. 1 An example simulation result of packed column jig

Product	Yield %	Grade %	Recovery %
Concentrate 1(**C1**)	24.39	96.51	47.08
Concentrate 2(**C2**)	11.79	96.58	22.77
Sum of Concentrate	**36.18**	**96.53**	**69.85**
Tailing 1 (**T1**)	25.65	5.86	3.01
Tailing 2 (**T2**)	38.17	35.56	27.14
Sum of Tailing	**63.82**	**23.62**	**30.15**
Total	100.00	50.00	100.00

Table 5. 2 The feed size distribution

Size Range	Yield %	Grade %
-0.200+0.141 mm	15.91	50.00
-0.141+0.100 mm	13.38	50.00
-0.100+0.071 mm	11.25	50.00
-0.071+0.050 mm	9.46	50.00
-0.050+0.036 mm	7.96	50.00
-0.036+0.025 mm	6.69	50.00
-0.025+0.018 mm	5.63	50.00
-0.018 mm	29.73	50.00
Total	100.00	50.00

By using the packed column jig combining with screening, a very good separation results can be attained with the total recovery of heavy mineral about 70% and the grade of heavy product of 96% of heavy mineral.

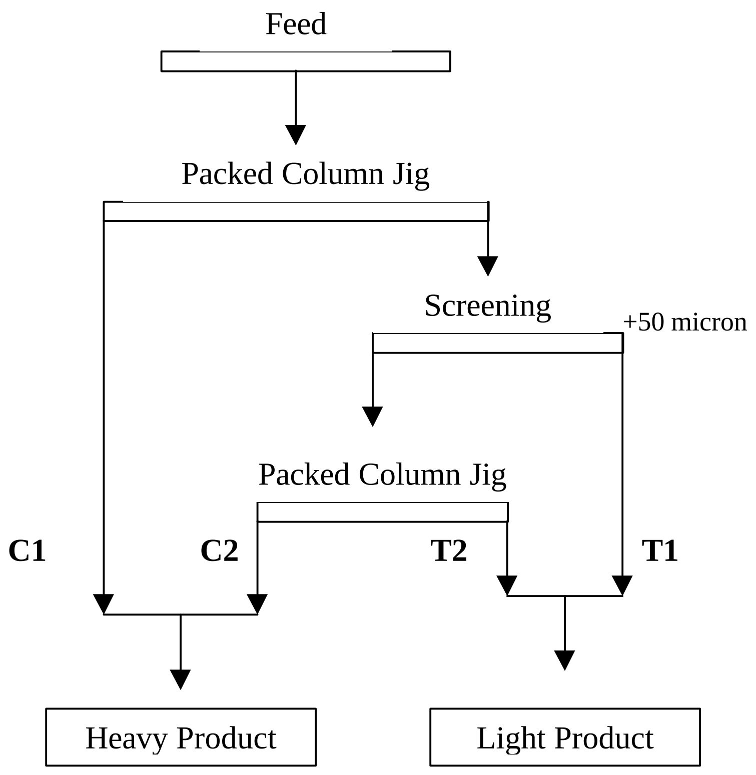


Figure 5. 1 A typical flowsheet of packed column jig for separation

The simulation result is similar to the actual test result of packed column jig. Figure 5.2 shows an actual typical packed column jig test result for a non-magnetic iron ore.

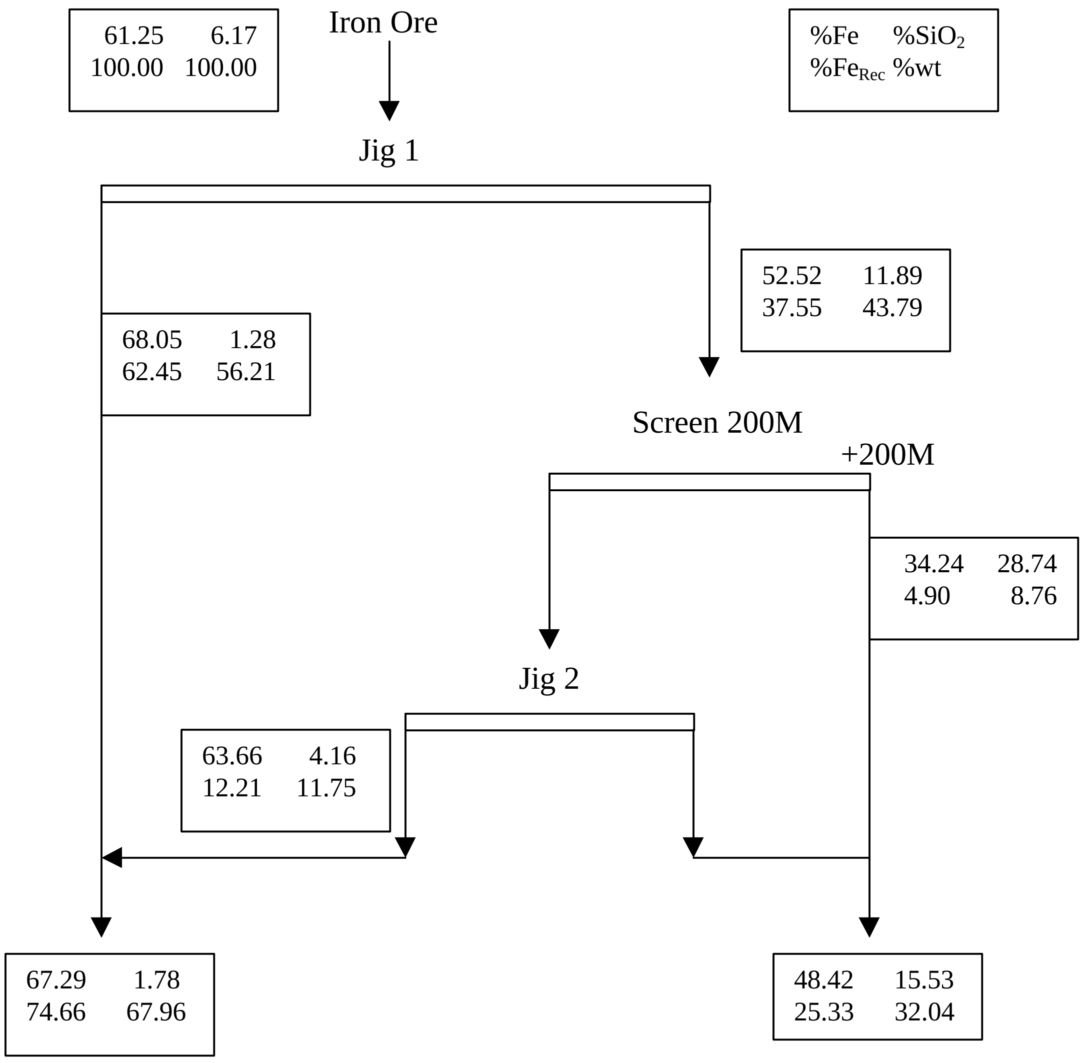


Figure 5. 2 Packed column jig flowsheet for non-magnetic iron ore

For narrow size range feed, only one stage of packed column jig can get very good separation result. Table 5.3 shows a simulation result for a feed with narrow size range.

Table 5. 3 A simulation result of packed column jig for narrow size feed

Size Range (mm)	Feed (%)		Concentrate (%)			Tailing (%)		
	yield	grade	yield	grade	Rec.	yield	grade	Rec.
-0.100 +0.071	50.00	50.00	29.91	83.58	100.00	20.09	0.00	0.00
-0.071 +0.050	25.00	50.00	12.50	100.00	100.00	12.50	0.00	0.00
-0.050 +0.036	12.50	50.00	6.25	100.00	100.00	6.25	0.00	0.00
-0.036 +0.025	6.25	50.00	2.50	100.00	79.87	3.75	16.67	20.13
-0.025 +0.018	3.13	50.00	0.00	100.00	0.00	3.13	50.00	100.00
-0.018 +0.013	1.56	50.00	0.00	100.00	0.00	1.56	50.00	100.00
-0.013 +0.009	0.78	50.00	0.00	100.00	0.00	0.78	50.00	100.00
-0.009	0.78	50.00	0.00	100.00	0.00	0.78	50.00	100.00
Total	**100.0**	**50.00**	**51.16**	**90.40**	**92.49**	**48.84**	**7.70**	**7.51**

The density of the heavy mineral is 4700.0 kg/m^3, the density of the light mineral is 2700 kg/m^3. The main simulation conditions are: stream velocity 0.014 m/s, pulsating frequency 3.0 1/s, pulsating amplitude 0.025 m, cell number in vertical direction 12.

5.2 The Trajectory of Particle in the Packed Column Jig

In the simulation, a visual display of particle movement in the cells of packed column jig can be viewed from the screen. This provides a close observation to the particle movement in the packed column jig. By observing the particle movement in different conditions. The following conclusion can be obtained from the simulation.

- The particle movement in the packed column jig is pulsating which corresponding to the pulsating of the stream.
- The particles that move down to the bottom have a tendency to move down to the bottom of each cell, while the particles that go up to the top have a tendency to move up to the ceiling of each cell. This shows that there exists a stratification process in each cell of packed column jig.
- If two particles can be separated in the packed column jig, they usually can be separated after 1-3 cells. This means that there is no need of too many cells for satisfactory separation, although the more the cell number, the more stable the process could be.

Figure 5.3 shows the pulsating movement of particles. Particle 1 (heavy particle) moves to the bottom of the jig, while particle 2 (light particle) moves to the top of the jig. Figure 5.4 shows images of particle trajectory in the packed column jig

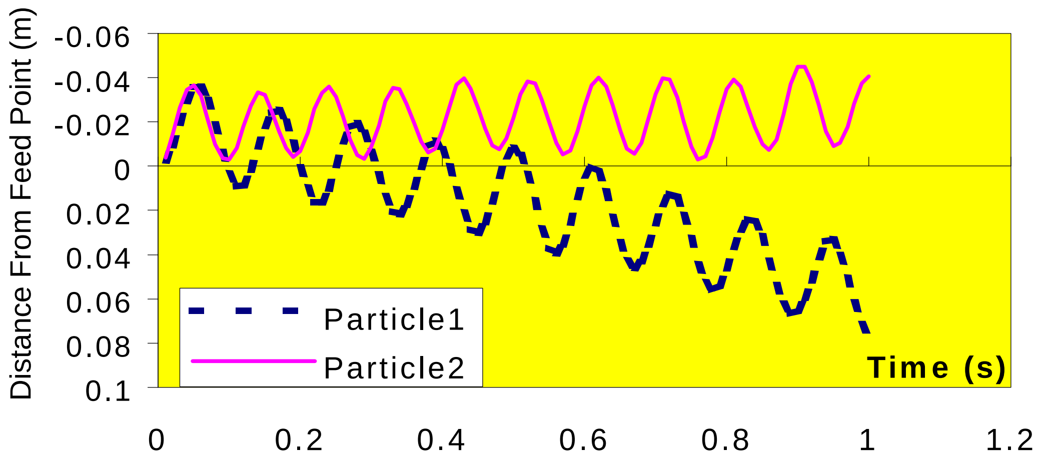


Figure 5. 3 The pulsating movement of particles in the packed column jig

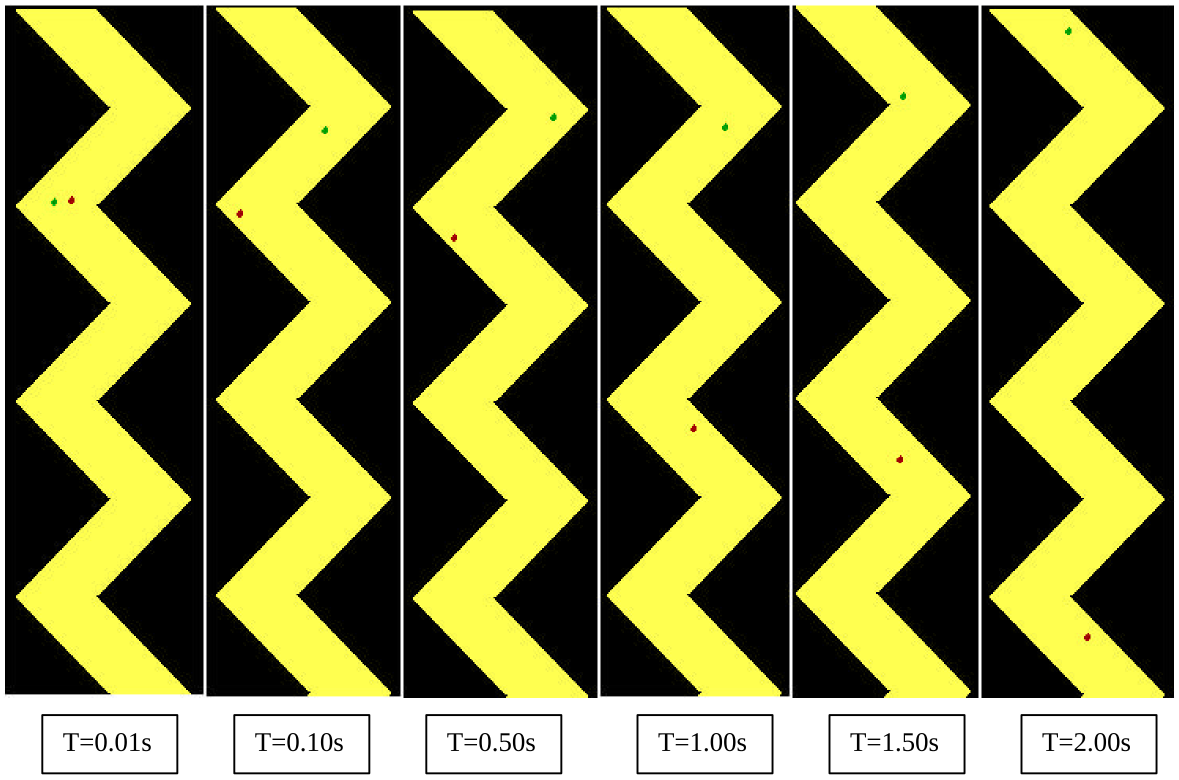


Figure 5. 4 Simulated particle trajectory in the packed column jig

5.3 The Effects of Packing and Its Dimensions on Jig Performance

Figures 5.5 and 5.6 show the effect of packing dimensions: width and length of the cell on the performance of packed column jig from the simulation. With the changing of the cell dimension, there is some effect on the performance of the packed column jig. When the jigging is in a critical condition, which means that the separation is optimized by parameters other than cell dimensions, the effect of the cell length & width becomes significant as shown in Figure 5.5. Otherwise, the effect of the cell dimensions is not significant as shown in Figure 5.6 according the simulation.

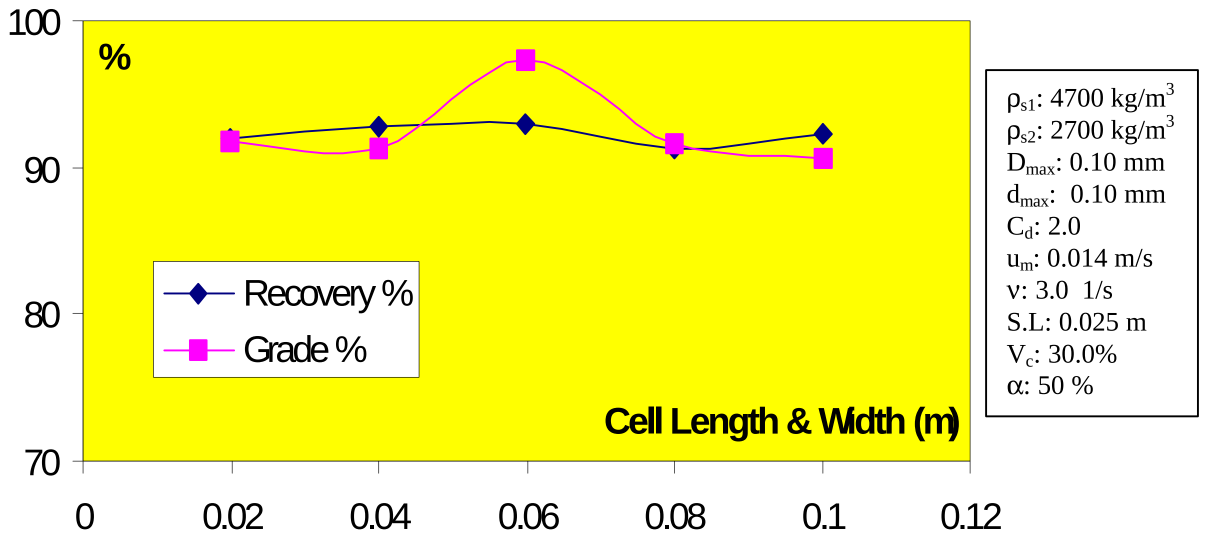


Figure 5. 5 The effect of cell dimension on the packed column jigging (1)

(in critical condition)

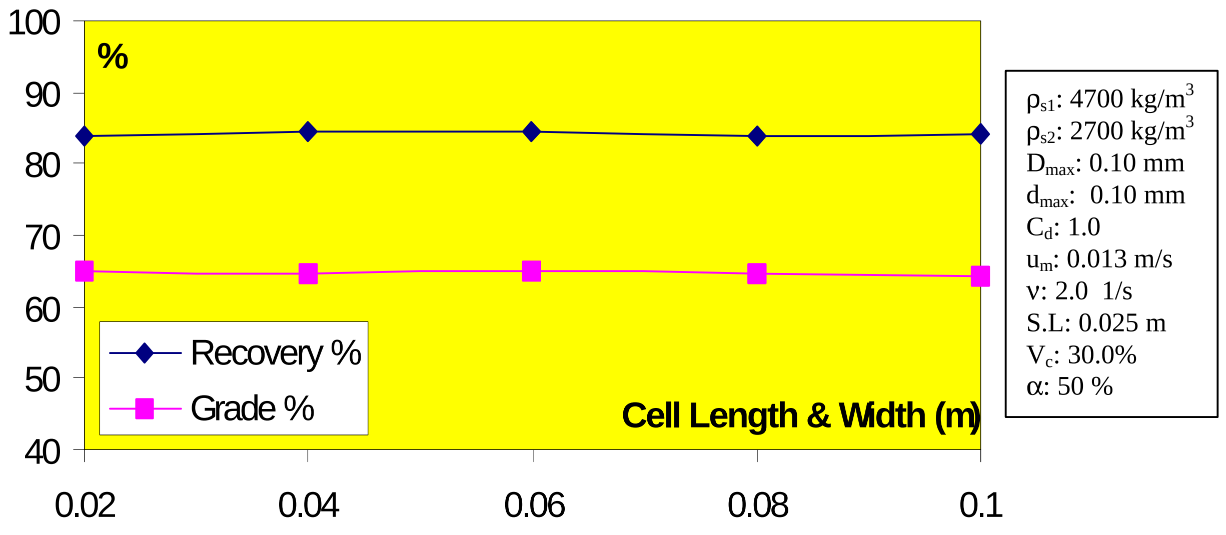


Figure 5. 6 The effect of cell dimension on the packed column jigging (2)

(in non-critical condition)

From the results, it can be concluded that the packing materials do have some effect on the performance of the jig. Small cell dimensions will bring a strong interaction of particles with the wall of cells, which is not good for separation of particle by density, while large cell dimensions will lose the effect of packing. For obtaining better result, cell dimensions should be carefully selected. Because of the limit of the model, the effect of packing is not fully expressed in this simulation. Actually, the packing in the packed column jig does have the following effects.

- The packing makes the pulp stream much more uniform.
- The packing eliminates the large vortex in the packed column jig while this kind of vortex in the conventional jig is always fully developed to increase the mixing of particles and reduce the separation selectivity.
- The packing makes the packed column jig easy to scale-up. The packed column jig can be approximately linearly scaled up.

Figures 5.7 and 5.8 show the effect of cell slope on the separation performance of packed column jig. For the finer particles as shown in Figure 5.7, as the slope increase, the recovery reduces slightly, while the grade of heavy products increases sharply in the beginning and then levels off. For the coarser particles as shown in the Figure5.8, the effect of the slope on the separation shows the similar effect for the finer particles.

As the cell slope decreases, the settling area for the particles in the packed column increases. Then more particles can probably settle down to the bottom. The effect seems to be much significant when slope is around 1.0. According to the simulation results, both higher grade and higher recovery of heavy product are obtained with the cell slope range between 1 to 5.

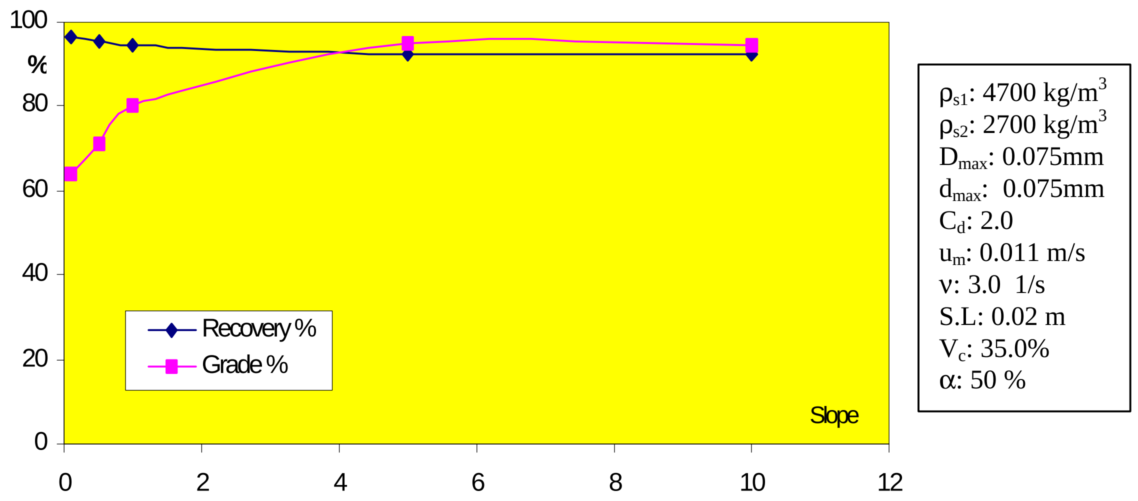


Figure 5. 7 The effect of cell slope on the packed column jigging (1)

(for finer particles)

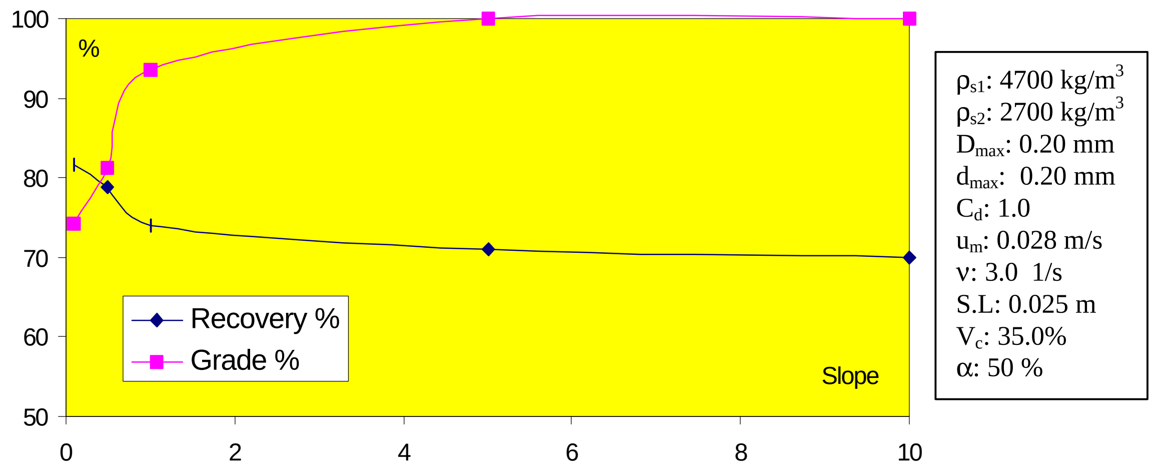


Figure 5. 8 The effect of cell slope on the packed column jigging (2)

(for coarser particles)

5.4 The Effects of Pulsating Frequency and Amplitude on Jig Performance

The pulsating frequency and amplitude are the main operation parameters of the packed column jig. Figure 5.9 shows the effect of pulsating frequency on the jigging performance of the packed column jig. As the pulsating frequency increases, the recovery of heavy mineral first goes up and then goes down, while the grade of the heavy product slightly reduces at first and then increases.

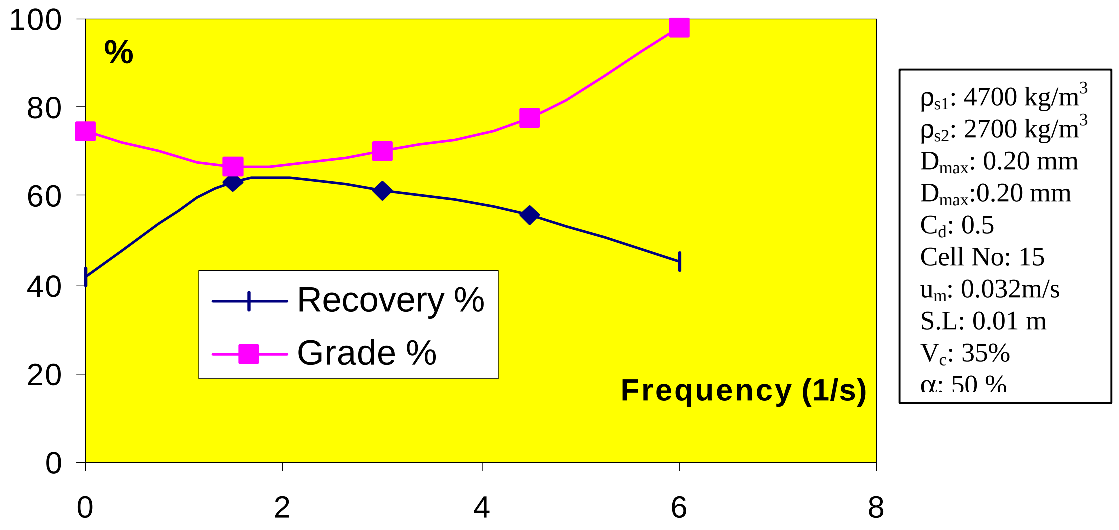


Figure 5. 9 The effect of pulsating frequency on the packed column jigging

The pulsating stroke length affects the separation with the similar mechanism as the frequency does, which is shown in Figure 5.10.

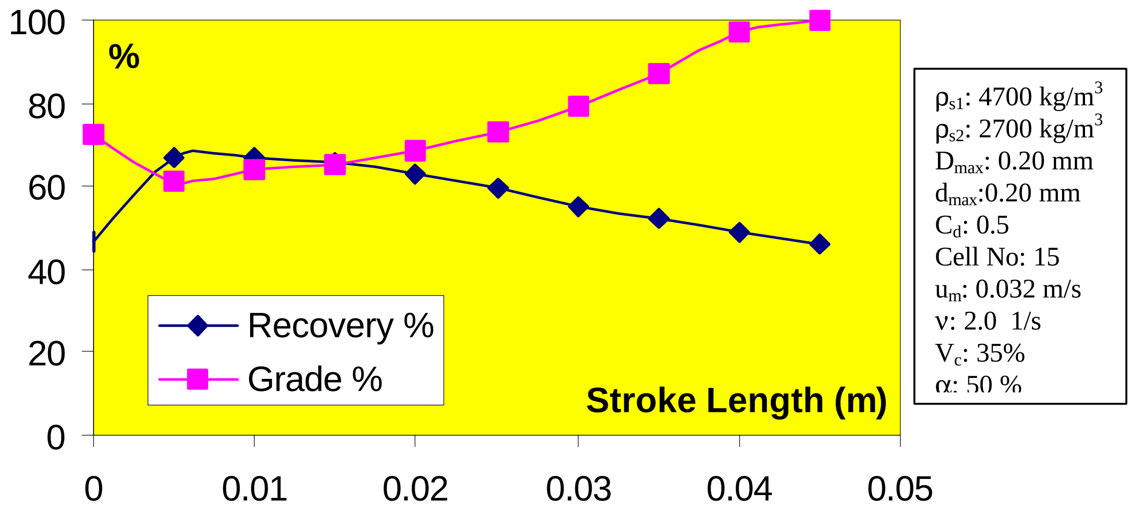


Figure 5. 10 The effect of pulsating amplitude on the packed column jigging

It is interesting that there exist critical values for both the frequency and the stroke length where the grade of heavy product is even worse than without pulsating. In the beginning to increase the pulsating frequency and amplitude, the particles in the packed column jig start to suspend and disperse. Although the heavy particles become easier to sink to the bottom, the larger light particles becomes easier to sink too. Perhaps this is the reason for the increase of recovery and the decrease of grade for the heavy product. As the pulsating frequency or amplitude further increases, the particles become well suspended and dispersed, even the largest light particle as well as a number of middle-sized heavy particles go to the top. Then the grade of heavy product increases and the recovery gradually decreases. It seems that the pulsating frequency and stroke length greatly affect the packed column jigging performance. A higher frequency and higher stroke length favors the higher grade of heavy product. But it also needs to consider the trade-off with the decrease of recovery and the increase of energy consumption.

From the above figures, it is clear that the selection of the pulsating frequency and amplitude (stroke length) is very important for obtaining good separation results.

5.5 The Effects of the Stream Velocity on Packed Column Jig Performance

The stream velocity is another important factor to affect the separation in packed column jig. It determines the particle size to be suspended. It also has a close relationship with the solid concentration inside the jig. From the microscopic view, it determines the amplitude of drag force and affects the random mechanic resistance force.

Figures 5.11 and 5.12 show the effect of stream velocity on the separation performance of packed column jig for different feed sizes. With the increase of up-stream velocity, the grade of heavy product increases, while the recovery of heavy product decreases. For the coarse feed, the stream velocity range for obtaining a reasonable separation result is relatively large and the adjustment of the stream velocity is easy to make. For the finer feed below 30 microns, this stream velocity range becomes much narrower and it becomes difficult to control the stream velocity in a narrow range.

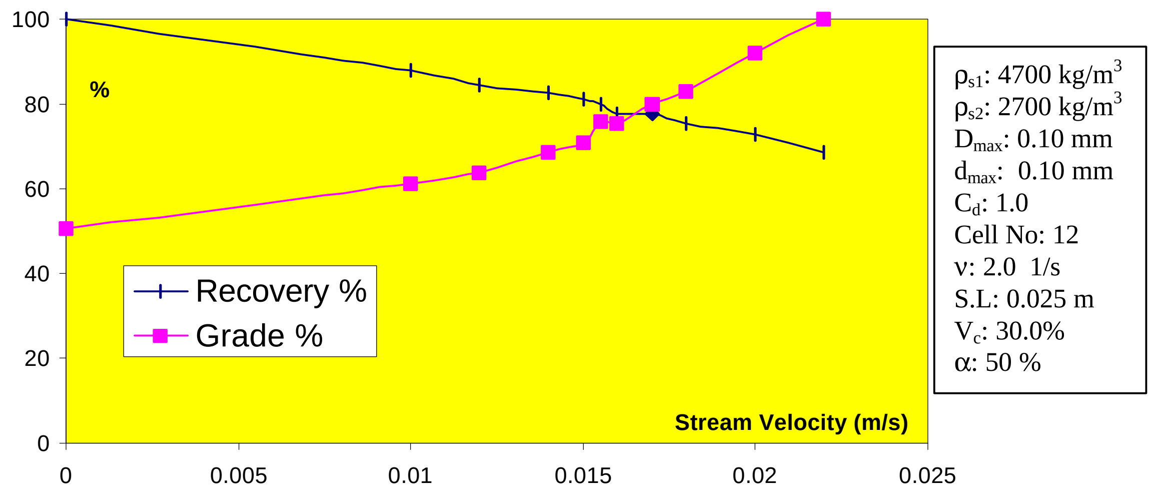


Figure 5. 11 The effect of stream velocity on the packed column jigging (1) (D_{max}=0.1mm)

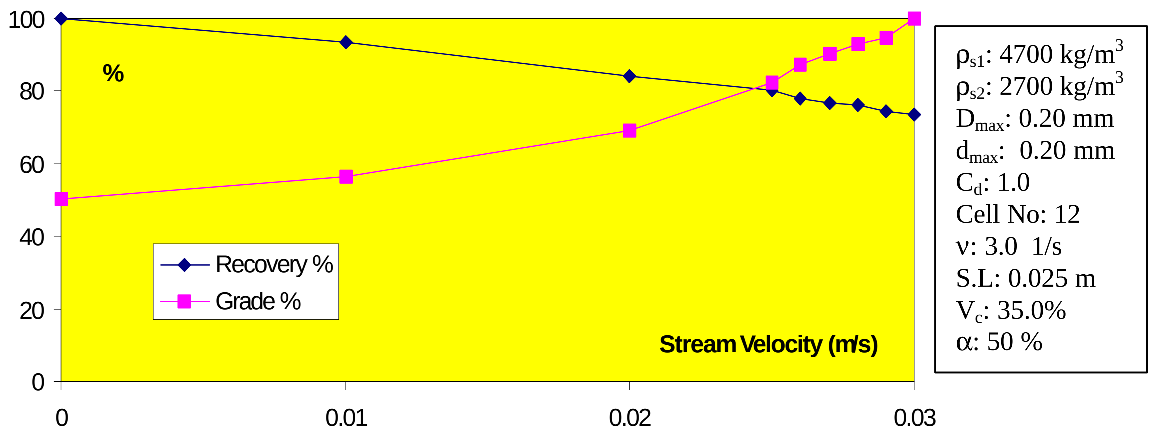


Figure 5. 12 The effect of stream velocity on the packed column jigging (2) (D_{max}=0.2mm)

5.6 The Effects of the Jig Height on Jig Performance

In the simulation, the effect of jig height can be simulated either by increasing the vertical cell's number or by increasing the height of each virtual cell. Figure 5.13 shows the effect of jig height by changing the height of each cell to the separation performance of packed column jig. As the jig height increases, the grade of heavy product increases slightly while the recovery decreases slightly.

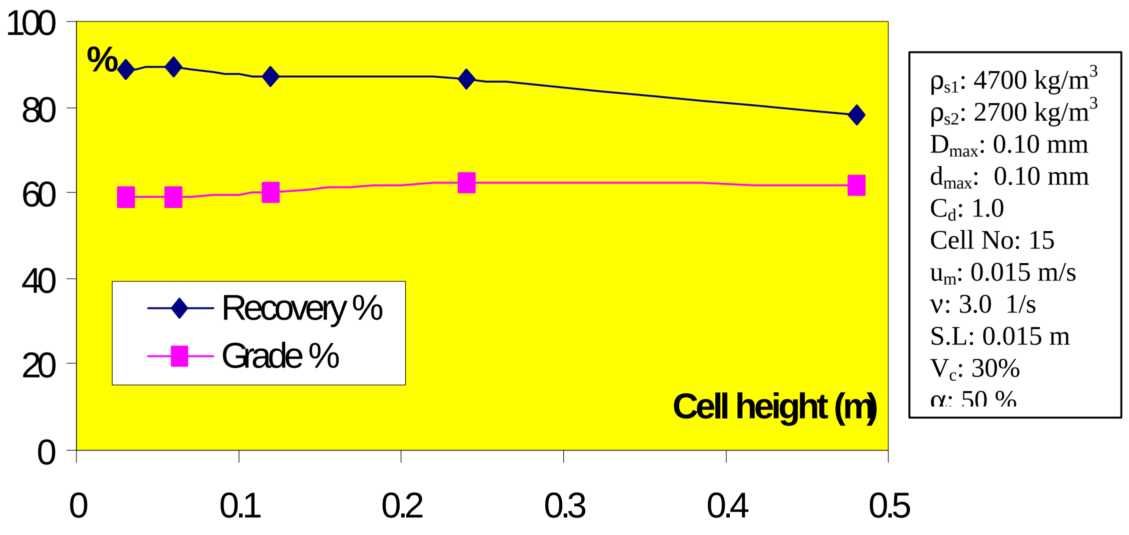
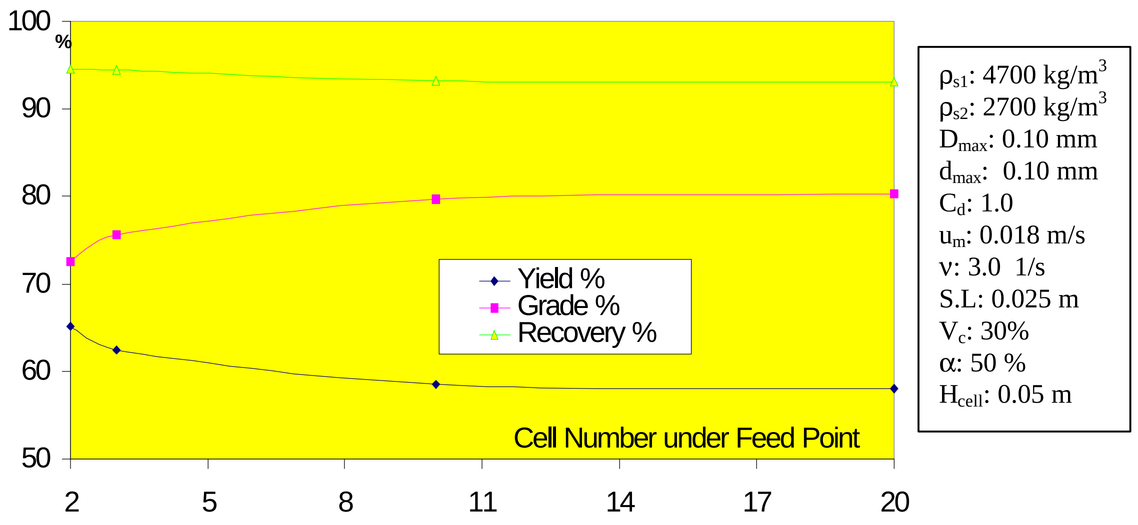


Figure 5. 13 The effect of cell height on the packed column jigging

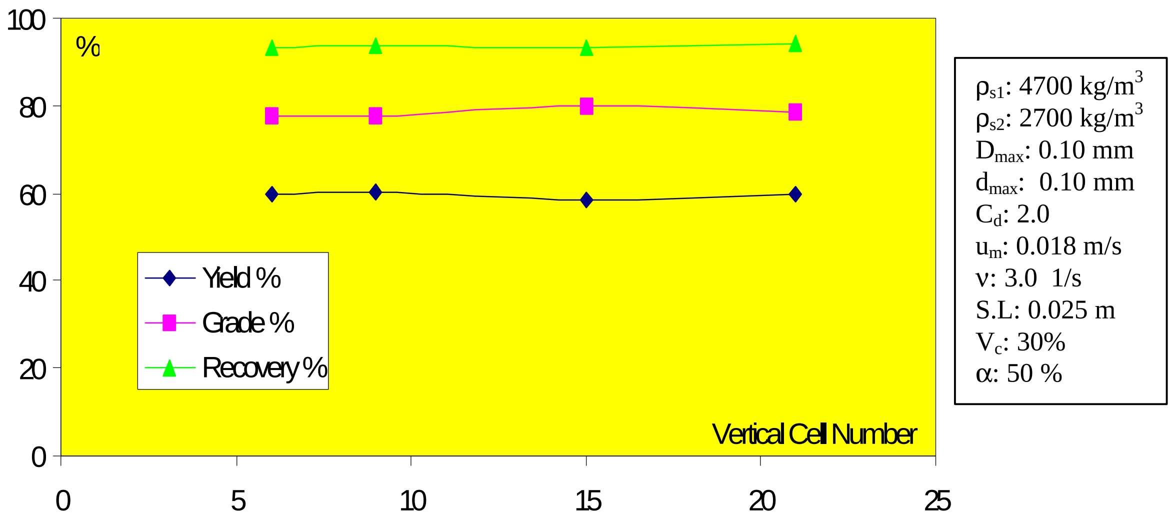
It is commonly accepted that the higher the jig, the better the separation performance of the packed column jig should be. When the jig height above the feed point keeps unchanged, the effect of jig height below the feed point on the performance of packed column jig is shown in Figure 5.14.

As the jig height below the feed point increases, the grade of the heavy mineral in the heavy product goes up slightly, while the recovery reduces slightly. But this phenomenon is not so significant when the jig height below feed point exceeds a certain height. According to the simulation results, the effect of jig height on the jig performance is quite small when the ratio of cell number up to down the feed point keeps the same as shown in Figure 5.15. This conclusion is drawn by the assumption that the stream in each cell is the same. Thus, the actual scale-up of the packed column should consider the vertical cell number for making the stream distribute uniform.



ρ_{s1}: 4700 kg/m^3
ρ_{s2}: 2700 kg/m^3
D_{max}: 0.10 mm
d_{max}: 0.10 mm
C_d: 1.0
u_m: 0.018 m/s
ν: 3.0 1/s
S.L: 0.025 m
V_c: 30%
α: 50 %
H_{cell}: 0.05 m

Figure 5. 14 The effect of cell number on the packed column jigging (1)
(Keep the cell number above the feed point unchanged)



ρ_{s1}: 4700 kg/m^3
ρ_{s2}: 2700 kg/m^3
D_{max}: 0.10 mm
d_{max}: 0.10 mm
C_d: 2.0
u_m: 0.018 m/s
ν: 3.0 1/s
S.L: 0.025 m
V_c: 30%
α: 50 %

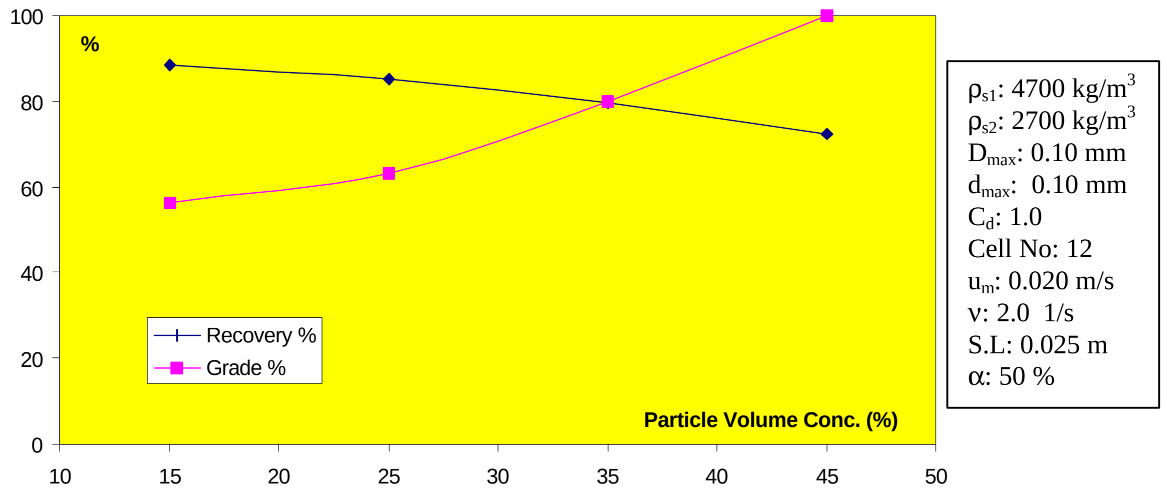
Figure 5. 15 The effect of cell number on the packed column jigging (2)

(Keep cell ratio unchanged)

It seems that there is a limit for improving the separation performance only by increasing the jig height. Making the height of the jig too high may course inconvenience in operation and increase in energy consumption as well as investment.

5.7 The Effects of Particle Packing Density on Jig Performance

The particle packing density in the packed column jig seems to have an important role for the separation of particles. Figure 5.16 shows this effect.



ρ_{s1}: 4700 kg/m^3
ρ_{s2}: 2700 kg/m^3
D_{max}: 0.10 mm
d_{max}: 0.10 mm
C_d: 1.0
Cell No: 12
u_m: 0.020 m/s
ν: 2.0 1/s
S.L: 0.025 m
α: 50 %

Figure 5. 16 The effect of particle packing density on the packed column jigging

From the simulation results shown in Figure5.16, the particle packing density affects the jig performance markedly. A higher particle packing density in the packed column jig makes a higher pulp density in each cell. The higher the pulp density as long as the particles are still in suspension and dispersion, the better the separation of particles according to the particle density difference. In the practical operation, it is critical to keep a relative high particle packing concentrate to give good separation results. There are some ways in practice that may help to obtain a higher particle packing density:

- Preparing feed to narrow size range.
- Keeping a high solid concentration in the feed.
- Using a screen in the bottom section of the packed column jig as in the conventional jig did.

5.8 The Effects of Magnetic Force on Jig Performance

When the heavy mineral is magnetic, we can use the magnetic field to improve the performance of the packed column jig. In this case, the magnetic flocculation and gravitational separation happen simultaneously in the jig. There are two main advantages comparing with sole magnetic separation or gravitational separation. The first is that it can reject the majority of interlocked particles that are difficult to be rejected by magnetic separation. The second is that it can recover more heavy minerals than magnetic separation or gravitational separation because it makes magnetic flocs and is also able to recover accompanied non-magnetic heavy particles. There were a number of very good results in actual test of magnetic ores by using the packed column jig with magnetic flocculation. In this simulation, the magnetic flocculation is simply simulated by exerting a magnetic force to the magnetic particles. Table 5.4 compares the difference of separation performance of packed column jig between those with magnetic flocculation and without magnetic flocculation. The heavy mineral is magnetic with density of 4700 kg/m^3 and the light mineral is non-magnetic with density of 2700 kg/m^3. The maximum particle size is 0.10 mm with a size distribution coefficient 1.0.

Table 5. 4 Simulation results of packed column jigging on a magnetic ore

Size Range	Feed (%)		Concentrate (%) (without magnetizing)			Concentrate (%) (magnetizing)		
(mm)	yield	grade	yield	grade	Rec.	yield	grade	Rec.
-0.100 +0.071	29.29	50.00	17.50	83.68	100.00	15.64	93.64	100.00
-0.071 +0.050	20.71	50.00	10.36	100.00	100.00	10.36	100.00	100.00
-0.050 +0.036	14.64	50.00	7.32	100.00	100.00	7.32	100.00	100.00
-0.036 +0.025	10.36	50.00	3.98	100.00	76.90	5.18	100.00	100.00
-0.025 +0.018	7.32	50.00	0.00	100.00	0.00	3.66	100.00	100.00
-0.018 +0.013	5.18	50.00	0.00	100.00	0.00	2.59	100.00	100.00
-0.013 +0.009	3.66	50.00	0.00	100.00	0.00	1.09	100.00	59.52
-0.009	8.84	50.00	0.00	100.00	0.00	0.00	100.00	0.00
Total	**100.0**	**50.00**	**39.16**	**92.71**	**72.61**	**45.83**	**97.83**	**89.68**

From the simulation results, it is clear that by using the magnetic field, the recovery and grade of magnetic heavy particle are improved. The separation results can be further improved by regrinding. By regrinding, the magnetic particles always can form certain size flocs, while the amount of interlocked particles will reduce and the average particle size of the light mineral will decrease, which make the light mineral particles easy to be rejected from the top of the packed column jig. Actual tests of packed column jig for magnetic taconite ores have obtained a number of very good results. Perhaps, this is the most potential area in which the packed column jig will play an important role.

5.9 The Effects of Feed Properties on Jig Performance

5.9.1 Density Difference

The density difference between two kinds of mineral particles is the base for separating the particles by gravitational separation. The density difference has a key effect on the separation in packed column jig, as can be seen from Figure 5.17.

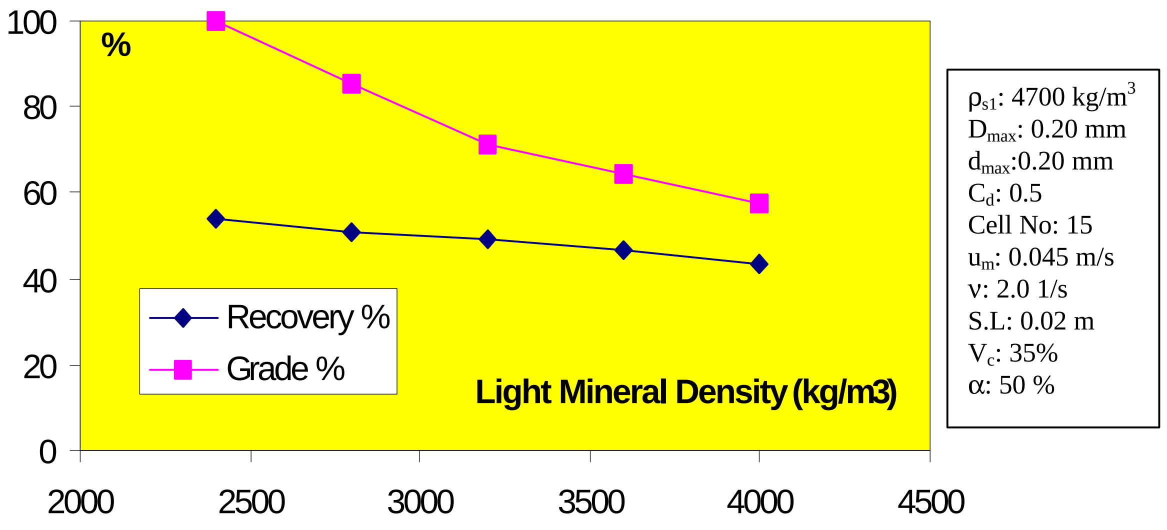


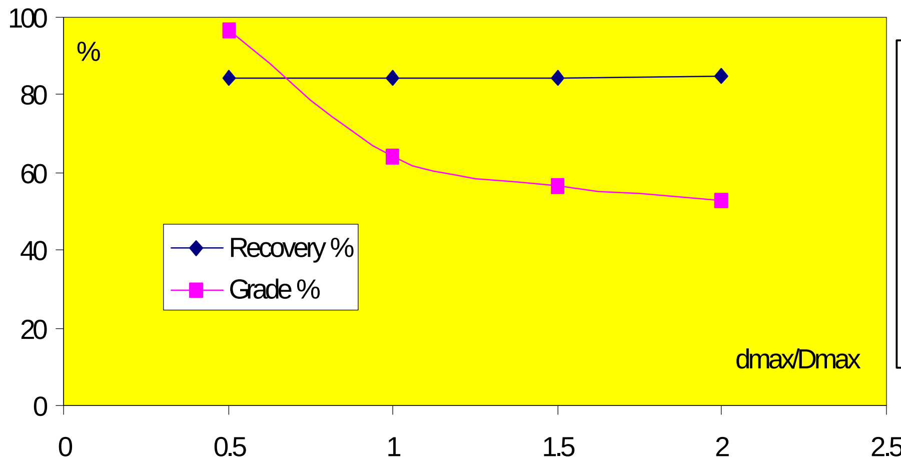
Figure 5. 17 The effect of density difference on the packed column jigging

When the density difference between two kinds of minerals is larger, the separation is easy to be attained. When the density difference is smaller, the separation becomes more difficult.

The feasibility of separation in packed column jig can be determined by the same coefficient defined in gravity separation theory in the case of non-magnetic ore. When two minerals of magnetic and non-magnetic are separated with the aid of magnetic flocculation, the feasibility can be significantly improved.

5.9.2 Particle Size and Distribution

In gravitational separation, particle size and distribution are the very important factors to affect the separation. With suitable size and distribution, the separation may become easier which is already shown in Table 5.1 and Table 5.3. Usually the narrower the feed size range, the better the separation. Fig 5.18 shows the effect of relative size between light and heavy minerals on the separation performance of packed column jig.



ρ_{s1}: 4700 kg/m^3
D_{max}: 0.10 mm
C_d: 1.0
Cell No: 12
u_m: 0.018 m/s
ν: 2.0 1/s
S.L: 0.025 m
V_c: 30%
α: 50 %

Figure 5. 18 The effect of particle size ratio on the packed column jigging

It can be seen from the Figure 5.18 that as the particle size of heavy mineral in the feed increases relatively to that of light mineral, the separation becomes better. When the particle size of heavy mineral in the feed is larger than that of the light mineral, the particle size will bring positive effect for the separation of minerals. If the particle size of light mineral is larger than that of the heavy mineral, a negative effect will bring to the separation. Sometimes, we should have a pre-concentrate with the whole feed size range, and then by regrinding the pre-concentrate to make the particle size distribution narrower to improve the efficiency of the cleaning separation. Also, preparation of the feed to different size ranges and processing separately is usually a good approach to give better separation.

5.10 The Work Needed for Improving the Simulation

The simulation has given a view for the packed column jig and shown the effects of different parameters on the performance. Some interpreted conclusions from the simulation are quite easy to understand, but some results are still needed to be verified by further research. For the time constraint, this simulation gave just an approximate expression of the actual packed column jigging process. For a more accurate description of the packed column jig, more work is needed in the following areas:

- The variance of pulp density in cells along the vertical direction of the packed column jig. In this simulation, the pulp density is assumed to be the same for every cell. Actually, there should be some distribution along the jig height. A dynamic simulation with a feedback of continuous correction of the density assumption of each cell along the height direction was tried. It was dropped finally because it took a long time to become equilibrium and needed a lot of time to refine the code.
- The feed-rate and feed solid concentration. These two parameters are not considered directly in this simulation.   They are important and are best to be directly involved in the simulation. Further knowledge is needed about the packed column jig micro-dynamic processes to combine these two factors for the simulation. This only can be done with a feedback-correction route that needs more work to refine the code

and a much higher-speed computer is needed to carry out the simulation in a reasonable time frame.

- The packing effects. The packing material has a number of effects in the jigging process. Some effects are already known, while others are not. How to accurately express these factors, such as the effect of reducing the vortex, and the effect for even distribution of stream, …etc is still need to be determined.

- The pulsating curve. By changing the shape of pulsating curve, the particle movement in the packed column jig can be much different. The separation may improve by the modification of this pulsating curve. Because of the difficulty in describing an arbitrary form of pulsating curve in a simple model, only the harmonic pulsation has been considered in this simulation.

- Capacity of the packed column jig. In this simulation, the capacity of the packed column jig was not considered, although the capacity could be estimated by counting the average time of particles passing through the packed column jig. Actually, the capacity is an important economical factor for the application of packed column jig.

Chapter 6 Conclusions

1. Packed column jig is a new invention toward processing of fine particles. It showed promising test results in processing magnetic ore, coal as well as some other ores.
2. The simulation of this thesis work gives a deeper and extensive view for the packed column jigging process. The simulation was conducted with a microscopic model considering various forces acting on individual solid particles. It can simulate the packed column jigging in different geometry, feed and operation conditions, and is able to give a visual simulation of particles moving in the packing cells. The simulation provides a convenient tool for the further improvement of the packed column jig technology.
3. The pulsating frequency, amplitude and stream up velocity are the most important operation parameters for packed column jigging. Generally, increasing the pulsating frequency, amplitude and the stream-up velocity will increase the grade of heavy product while the recovery of heavy mineral will decrease.
4. The effect of packing dimension is significant when the jigging is in optimized critical conditions. This means that the packing affects the jigging performance when higher efficiency is needed in jigging.
5. The solid particle packing density is also an important factor in packed column jigging process. The higher the solid packing concentrate, the better the separation could be as long as the particles are still free of moving.
6. The density difference between heavy mineral and light mineral in the feed is the base for the separation of the particles in gravitational separation. The larger the density difference, the easier the separation should be.

7. The particle size distribution also affects the separation in the jig. The narrower the particle size range of the feed, the better the separation.
8. Magnetic flocculation is much helpful for the separation in the packed column jig. Proper magnetic flocculation combining with the dispersing action in the packed column jig makes the separation much better.
9. The jig height is not a critical factor to affect the performance of packed column jig.

References

Bagnold, R. A., 1954, "Experiments on a gravity-free dispersion of large solid spheres in a Newtonian fluid under shear," Proceedings of the Royal society, A., No. 1160, Vol. 225, pp. 49.

Beck, A.J.G., Holtham, P.N., 1993, "Computer simulation of particle stratification in a two-dimensional batch jig," Minerals Engineering, Vol. 6, No. 5, pp. 523-536.

Breuer, H. and Jungmann, A. 1986, "New possibilities of jigging separation for fine and ultrafine fractions," Aufbereitungs-Technik/Mineral Processing, Vol. 27, No. 7, pp. 649-658.

Campbell, T. P., 1991, "Centrifugal jig pulsing system," US patent 4,998,986.

Davies, J. T., 1972, *Turbulence Phenomena*, Academic Press, New York and London, pp.28-29

De Jong, T. P. R. and Dalmijn, W. L. 1997, "Improving jigging results of non-ferrous car scrap by application of a intermediate layer," International Journal of Mineral Processing, Vol. 49, No.1-2, pp. 59-72.

De Jong, T. P. R., Witteveen, H. J. and Dalmijn, W. L., 1996, "Penetration velocities in a homogeneous jig bed," International Journal of Mineral Processing, Vol. 46, No. 3-4, pp. 277-291.

Eichholz, G., Moskala, R., Schneider-Kuehn, U. and Klefisch, D., 1998, "Compensator pulsator jig - a further development of the membrane pulsator jig," Aufbereitungs-Technik/Mineral Processing, Vol. 39, No. 3, pp.124-129.

Fellensiek, E. 1986, "Improving the actual separation efficiency of jigs," Aufbereitungs-Technik/Mineral Processing, Vol.27, No.12, pp. 649-658.

Gray, A.H., 1997, "Inline Pressure Jig - an exciting, low cost technology with significant operational benefits in gravity separation of minerals," Technical Proceedings Resourcing the 21st Century Proceedings of the 1997 AusIMM Annual Conference, Ballarat, Aust., pp. 259-265.

Holland-Batt, A. B., 1998, "Gravity separation: a revitalized technology," Mining Engineering, Vol. 50, No. 9, pp. 43-47

Laser, U., and Wagener, W., 1994, "Dry jigging in a centrifugal field," Aufbereitungs-Technik/Mineral Processing, Vol. 35, No. 5, pp. 225-232.

Li, Xianguo, Abbott, John, 1993, "Simulation of the jigging process," Coal Preparation, Vol. 13, No. 3-4, pp. 197-207.

Lin, I. J., Krush-Bram, M. and Rosenhouse, G., 1998, "Benefication of minerals by magnetic jigging, Part 3. The bed effects and the multi-frequency magnetic jig," International Journal of Mineral Processing, Vol. 55, No. 1, pp. 61-72

Lin, I. J., Krush-Bram, M. and Rosenhouse, G. 1997, "Benefication of minerals by magnetic jigging, Part 1. Theoretical aspects," International Journal of Mineral Processing, Vol. 50, No. 3, pp.143-159.

Lin, I. J., Krush-Bram, M. and Rosenhouse, G. 1997, "Benefication of minerals by magnetic jigging, Part 2. Identification of the parameters and verification of the mathematical model for the theoretical analysis of the mineral particles motion in the magnetic jig," International Journal of Mineral Processing, Vol. 54, No. 1, pp. 29-44.

Lyman, G. J., 1992, "Review of jigging principles and control," Coal Preparation, Vol. 11, No. 3-4, pp.145-165.

Mackeley, M. R., Smith, K. B. and Wise, N. P., 1993, "The Mixing and Separation of Particle Suspension Using Oscillatory Flow in Baffled Tubes," Trans IChemE, Vol. 71, Part A, pp. 649-656.

Meloy, T. P., Bilgesu, I. and Yang, D.C., 1997, "Packed Column Jig – A Model," submitted for presentation, XX IMPC in Aachen.

Mesters, K. and Kurkowski, H., 1997, "Density separation of recycling building materials by means of jig technology," Aufbereitungs-Technik/Mineral Processing, Vol. 38, No. 10, pp. 536-542.

Mishra, B. K. and Mehtotra, S. P., 1998, "Modeling of particle stratification in jigs by the discrete element method," Minerals Engineering, Vol. 11, No. 6, pp. 511-522.

Richardson, J. F. and Zaki, W. N. 1954, "Sedimentation and fluidization," Trans. Instn. Chem. Eng., Vol.32, pp. 35-53.

Rong, R. X., Lyman and Geoffrey J., 1992, "Effect of jigging time and air cycle on bed stratification in a pilot scale Baum jig," Fuel, Vol. 71, No. 1, pp. 115-123.

Schonert, K. Gerstenberg, R. and Zimmermann, W., 1987, "Jigging with a density-sensitive filter layer," Preprints for Presentation as the SME Annual Meeting, Denver, CO, USA, pp. 87-127.

Schubert, H., 1994, "Review of the fundamentals of wet jigging," Aufbereitungs-Technik/Mineral Processing, Vol. 35, No. 7, pp. 337-347.

Tavares, L.M., King, R. P., 1995, "Useful model for the calculation of the performance of batch and continuous jigs," Coal Preparation, Vol.15, Vol.3-4, pp. 99-128.

Tucker, P, 1995, "Modeling the Kelsey centrifugal jig," Minerals Engineering, Vol. 8, No.3, pp. 333-336.

Yang, D. C., 1996, "Device and Process for Gravitational Separation of Solid Particles," US Patent No. 5,507,393, April 16.

Yang, D. C. and Meloy, T. P., 1997, "Gravity Separation of Minus 500 Mesh particles," submitted for presentation, 1997 SME.

Appendices

Appendix A: The Visual Basic Code For Input And Output

```
'Input form for the condition simulation tests of cell dimensions

Dim sel As Integer

Private Sub Combo2_Click()
If Combo2.Text = "Width" Then
labc(0).Caption = "Length"
labc(1).Caption = "Slope"
labc(3).Caption = "Width"
sel = 2
Open "a:minl32.in" For Output As #2
Open "a:cell32.in" For Output As #3
Open "a:grad32.in" For Output As #4
Open "a:condn32.in" For Output As #5

ElseIf Combo2.Text = "Length" Then
labc(0).Caption = "Width"
labc(1).Caption = "Slope"
labc(3).Caption = "Length"
sel = 1
Open "a:minl31.in" For Output As #2
Open "a:cell31.in" For Output As #3
Open "a:grad31.in" For Output As #4
Open "a:condn31.in" For Output As #5

ElseIf Combo2.Text = "Slope" Then
labc(0).Caption = "Length"
labc(1).Caption = "Width"
labc(3).Caption = "Slope"
sel = 3
Open "a:minl33.in" For Output As #2
Open "a:cell33.in" For Output As #3
Open "a:grad33.in" For Output As #4
Open "a:condn33.in" For Output As #5

Else
labc(1).Caption = "Slope"
labc(3).Caption = "Both L&W"
sel = 4
Open "a:minl34.in" For Output As #2
Open "a:cell34.in" For Output As #3
Open "a:grad34.in" For Output As #4
Open "a:condn34.in" For Output As #5
End If
End Sub

Private Sub concel_Click()
celldimmension.Hide
selection.Show
End Sub

Private Sub Form_Load()
Combo2.AddItem "Width"
```

```
Combo2.AddItem "Length"
Combo2.AddItem "Slope"
Combo2.AddItem "Width & Length"
End Sub

Private Sub ok_Click()
Dim counter As Integer
Dim i As Integer
Dim j As Integer
Dim m As Integer
Dim n As Integer
Dim Sign As Integer
Dim IsFeedCell As Integer
Dim TotalVerticalCellNumber As Integer
Dim SectionNumber As Integer

Open "a:option.dat" For Output As #1
Print #1, 3, sel
Print #5, feedmag.Value; pointno.Text
For i = 0 To 1

Print #2, MineralName(i).Text; Spc(5); density(i).Text; _
      Spc(5); MaxSize(i).Text; Spc(5); sizep(i).Text; Spc(5); magcheck(i).Value
Next i

i = 0
TotalVerticalCellNumber = 0
Do While Check1(i).Value = 1
TotalVerticalCellNumber = TotalVerticalCellNumber + _
    vertical(i).Text
i = i + 1
SectionNumber = i
Loop

Sign = 1
For counter = 0 To SectionNumber - 1
n = SectionNumber - 1 - counter

IsFeedCell = Option1(n).Value
If IsFeedCell <> 0 Then IsFeedCell = 1

For j = 0 To vertical(n).Text - 1
m = TotalVerticalCellNumber - j

If sel = 4 Then
Print #3, m; Sign * cwidth(n).Text; cheight(n).Text; _
    water(n).Text; freq(n).Text; stroke(n).Text; _
    vol(n).Text; IsFeedCell

ElseIf sel = 3 Then
Print #3, m; clength(n).Text; cwidth(n).Text; cheight(n).Text; _
    water(n).Text; freq(n).Text; stroke(n).Text; _
    vol(n).Text; IsFeedCell

ElseIf sel = 2 Then
Print #3, m; clength(n).Text; Sign * cwidth(n).Text; cheight(n).Text; _
```

```
    water(n).Text; freq(n).Text; stroke(n).Text; _
    vol(n).Text; IsFeedCell

ElseIf sel = 1 Then
Print #3, m; clength(n).Text; Sign * cwidth(n).Text; cheight(n).Text; _
    water(n).Text; freq(n).Text; stroke(n).Text; _
    vol(n).Text; IsFeedCell

End If
IsFeedCell = False
Sign = Sign * (-1)
Next j
TotalVerticalCellNumber = TotalVerticalCellNumber - _
    vertical(n).Text
Next counter

Print #4, ratio(0).Text

For i = 0 To pointno.Text - 1
Print #5, lowerlimit.Text + _
    (upperlimit.Text - lowerlimit.Text) * i / (pointno.Text - 1)
Next i

Close #1
Close #2
Close #3
Close #4
Close #5
celldimmension.Hide
start.Show
End Sub



‘Input form for density condition simulation tests

Private Sub concel_Click()
densityform.Hide
selection.Show
End Sub

Private Sub density_Click()
If density.Text = "Heavy Change" Then
labels(0).Caption = "Light"
labels(1).Caption = "Heavy"
ElseIf density.Text = "Light Change" Then
labels(0).Caption = "Heavy"
labels(1).Caption = "Light"
End If
denshow.Caption = "Density"
End Sub

Private Sub Form_Load()
density.AddItem "Heavy Change"
density.AddItem "Light Change"
End Sub
```

```
Private Sub ok_Click()
Dim counter As Integer
Dim i As Integer
Dim j As Integer
Dim m As Integer
Dim n As Integer
Dim Sign As Integer

Dim temp1 As Double
Dim IsFeedCell As Integer
Dim TotalVerticalCellNumber As Integer
Dim SectionNumber As Integer

Open "a:option.dat" For Output As #1
Open "a:minl4.in" For Output As #2
Open "a:cell4.in" For Output As #3
Open "a:grad4.in" For Output As #4
Open "a:condn4.in" For Output As #5
Print #1, 4
Print #5, feedmag.Value; pointno.Text
If density.Text = "Heavy Change" Then
temp1 = (upperlimit.Text - lowerlimit.Text) / (pointno.Text - 1)
For i = 1 To pointno.Text
Print #2, "Heavy"; Spc(5); lowerlimit.Text + (i - 1) * temp1;
Print #2, Spc(5); MaxSize(0).Text; Spc(5); sizep(0).Text; Spc(5); magcheck(0).Value
Print #2, "Light"; Spc(5); denfixed.Text;
Print #2, Spc(5); MaxSize(1).Text; Spc(5); sizep(1).Text; Spc(5); magcheck(1).Value
Next i
ElseIf density.Text = "Light Change" Then
temp1 = (upperlimit.Text - lowerlimit.Text) / (pointno.Text - 1)
For i = 1 To pointno.Text
Print #2, "Heavy"; Spc(5); denfixed.Text;
Print #2, Spc(5); MaxSize(0).Text; Spc(5); sizep(0).Text; Spc(5); magcheck(0).Value
Print #2, "Light"; Spc(5); lowerlimit.Text + (i - 1) * temp1;
Print #2, Spc(5); MaxSize(1).Text; Spc(5); sizep(1).Text; Spc(5); magcheck(1).Value
Next i
End If

i = 0
TotalVerticalCellNumber = 0
Do While Check1(i).Value = 1
TotalVerticalCellNumber = TotalVerticalCellNumber + _
    vertical(i).Text
i = i + 1
SectionNumber = i
Loop

Sign = 1
For counter = 0 To SectionNumber - 1
n = SectionNumber - 1 - counter

IsFeedCell = Option1(n).Value
If IsFeedCell <> 0 Then IsFeedCell = 1
```

```
For j = 0 To vertical(n).Text - 1
m = TotalVerticalCellNumber - j

Print #3, m; clength(n).Text; cwidth(n).Text; cheight(n).Text; _
    Sign * slope(n).Text; water(n).Text; freq(n).Text; stroke(n).Text; _
    vol(n).Text; IsFeedCell

Sign = Sign * (-1)
IsFeedCell = False

Next j
TotalVerticalCellNumber = TotalVerticalCellNumber - _
    vertical(n).Text

Next counter

Print #4, ratio(0).Text

Close #1
Close #2
Close #3
Close #4
Close #5
densityform.Hide
start.Show
End Sub


‘Input form for pulsating frequency condition simulation tests

Private Sub concel_Click()
frequencyform.Hide
selection.Show
End Sub

Private Sub ok_Click()
Dim counter As Integer
Dim i As Integer
Dim j As Integer
Dim m As Integer
Dim n As Integer
Dim Sign As Integer
Dim IsFeedCell As Integer
Dim TotalVerticalCellNumber As Integer
Dim SectionNumber As Integer

Open "a:option.dat" For Output As #1
Open "a:minl5.in" For Output As #2
Open "a:cell5.in" For Output As #3
Open "a:grad5.in" For Output As #4
Open "a:condn5.in" For Output As #5

Print #1, 5
Print #5, feedmag.Value; pointno.Text
For i = 0 To 1
```

```
Print #2, MineralName(i).Text; Spc(5); density(i).Text; _
     Spc(5); MaxSize(i).Text; Spc(5); sizep(i).Text; Spc(5); magcheck(i).Value
Next i

i = 0
TotalVerticalCellNumber = 0
Do While Check1(i).Value = 1
TotalVerticalCellNumber = TotalVerticalCellNumber + _
   vertical(i).Text
i = i + 1
SectionNumber = i
Loop

Sign = 1
For counter = 0 To SectionNumber - 1
n = SectionNumber - 1 - counter

IsFeedCell = Option1(n).Value
If IsFeedCell <> 0 Then IsFeedCell = 1

For j = 0 To vertical(n).Text - 1
m = TotalVerticalCellNumber - j

Print #3, m; clength(n).Text; cwidth(n).Text; cheight(n).Text; _
   Sign * slope(n).Text; water(n).Text; stroke(n).Text; _
   vol(n).Text; IsFeedCell

Sign = Sign * (-1)
IsFeedCell = False

Next j

TotalVerticalCellNumber = TotalVerticalCellNumber - _
   vertical(n).Text

Next counter

Print #4, ratio(0).Text

For i = 0 To pointno.Text - 1
Print #5, lowerlimit.Text + _
   (upperlimit.Text - lowerlimit.Text) * i / (pointno.Text - 1)
Next i

Close #1
Close #2
Close #3
Close #4
Close #5
frequencyform.Hide
start.Show
End Sub


‘Input form for jig height condition simulation tests
```

```
Private Sub concel_Click()
jigheight.Hide
selection.Show
End Sub



Private Sub ok_Click()
Dim counter As Integer
Dim i As Integer
Dim j As Integer
Dim m As Integer
Dim n As Integer
Dim Sign As Integer

Dim IsFeedCell As Integer
Dim TotalVerticalCellNumber As Integer
Dim SectionNumber As Integer

Open "a:option.dat" For Output As #1
Open "a:minl6.in" For Output As #2
Open "a:cell6.in" For Output As #3
Open "a:grad6.in" For Output As #4
Open "a:condn6.in" For Output As #5

Print #1, 6
Print #5, feedmag.Value; pointno.Text
For i = 0 To 1

Print #2, MineralName(i).Text; Spc(5); density(i).Text; _
      Spc(5); MaxSize(i).Text; Spc(5); sizep(i).Text; Spc(5); magcheck(i).Value
Next i

i = 0
TotalVerticalCellNumber = 0
Do While Check1(i).Value = 1
TotalVerticalCellNumber = TotalVerticalCellNumber + _
    vertical(i).Text
i = i + 1
SectionNumber = i
Loop

Sign = 1
For counter = 0 To SectionNumber - 1
n = SectionNumber - 1 - counter

IsFeedCell = Option1(n).Value
If IsFeedCell <> 0 Then IsFeedCell = 1

For j = 0 To vertical(n).Text - 1
m = TotalVerticalCellNumber - j

Print #3, m; clength(n).Text; cwidth(n).Text; _
    Sign * slope(n).Text; water(n).Text; freq(n).Text; stroke(n).Text; _
    vol(n).Text; IsFeedCell
```

```
Sign = Sign * (-1)
IsFeedCell = False

Next j

TotalVerticalCellNumber = TotalVerticalCellNumber - _
   vertical(n).Text

Next counter

Print #4, ratio(0).Text

For i = 0 To pointno.Text - 1
Print #5, lowerlimit.Text + _
   (upperlimit.Text - lowerlimit.Text) * i / (pointno.Text - 1)
Next i

Close #1
Close #2
Close #3
Close #4
Close #5
jigheight.Hide
start.Show
End Sub


'Input form for particle observation

Private Sub concel_Click()
particle.Hide
selection.Show
End Sub

Private Sub ok_Click()
Dim counter As Integer
Dim i As Integer
Dim j As Integer
Dim m As Integer
Dim n As Integer
Dim Sign As Integer
Dim IsFeedCell As Integer
Dim TotalVerticalCellNumber As Integer
Dim SectionNumber As Integer

Open "a:option.dat" For Output As #1
Open "a:minl2.in" For Output As #2
Open "a:cell2.in" For Output As #3
Open "a:grad2.in" For Output As #4
Open "a:condn2.in" For Output As #5

Print #1, 2
Print #5, feedmag.Value
For i = 0 To 1

Print #2, MineralName(i).Text; Spc(5); density(i).Text; _
```

```
      Spc(5); MaxSize(i).Text; Spc(5); sizep(i).Text; Spc(5); magcheck(i).Value
Next i

i = 0
TotalVerticalCellNumber = 0
Do While Check1(i).Value = 1
TotalVerticalCellNumber = TotalVerticalCellNumber + _
    vertical(i).Text
i = i + 1
SectionNumber = i
Loop

Sign = 1
For counter = 0 To SectionNumber - 1
n = SectionNumber - 1 - counter
IsFeedCell = Option1(n).Value
If IsFeedCell <> 0 Then IsFeedCell = 1

For j = 0 To vertical(n).Text - 1
m = TotalVerticalCellNumber - j

Print #3, m; clength(n).Text; cwidth(n).Text; cheight(n).Text; _
    Sign * slope(n).Text; water(n).Text; freq(n).Text; stroke(n).Text; _
    vol(n).Text; IsFeedCell

Sign = Sign * (-1)
IsFeedCell = False

Next j
TotalVerticalCellNumber = TotalVerticalCellNumber - _
    vertical(n).Text

Next counter

Print #4, ratio(0).Text

For i = 0 To 1
Print #5, particlesize(i).Text

Next i

Close #1
Close #2
Close #3
Close #4
Close #5
particle.Hide
start.Show
End Sub



‘Input form for particle size condition simulation

Private Sub concel_Click()
particlesize.Hide
```

```
selection.Show
End Sub

Private Sub psize_Click()
If psize.Text = "Heavy Change" Then
labels(0).Caption = "Light"
labels(1).Caption = "Heavy"
ElseIf psize.Text = "Light Change" Then
labels(0).Caption = "Heavy"
labels(1).Caption = "Light"
End If
End Sub

Private Sub Form_Load()
psize.AddItem "Heavy Change"
psize.AddItem "Light Change"
End Sub

Private Sub ok_Click()
Dim counter As Integer
Dim i As Integer
Dim j As Integer
Dim m As Integer
Dim n As Integer
Dim Sign As Integer
Dim temp1 As Double
Dim IsFeedCell As Integer
Dim TotalVerticalCellNumber As Integer
Dim SectionNumber As Integer

Open "a:option.dat" For Output As #1
Open "a:minl7.in" For Output As #2
Open "a:cell7.in" For Output As #3
Open "a:grad7.in" For Output As #4
Open "a:condn7.in" For Output As #5
Print #1, 7
Print #5, feedmag.Value; pointno.Text
If psize.Text = "Heavy Change" Then
temp1 = (upperlimit.Text - lowerlimit.Text) / (pointno.Text - 1)
For i = 1 To pointno.Text
Print #2, "Heavy"; Spc(5); den(0).Text; Spc(5); lowerlimit.Text + (i - 1) * temp1;
Print #2, Spc(5); sizep(0).Text; Spc(5); magcheck(0).Value
Print #2, "Light"; Spc(5); den(1).Text;
Print #2, Spc(5); onefixed.Text; Spc(5); sizep(1).Text; Spc(5); magcheck(1).Value
Next i
ElseIf psize.Text = "Light Change" Then
temp1 = (upperlimit.Text - lowerlimit.Text) / (pointno.Text - 1)
For i = 1 To pointno.Text
Print #2, "Heavy"; Spc(5); den(0).Text;
Print #2, Spc(5); onefixed.Text; Spc(5); sizep(0).Text; Spc(5); magcheck(0).Value
Print #2, "Light"; Spc(5); den(1).Text; Spc(5); lowerlimit.Text + (i - 1) * temp1;
Print #2, Spc(5); sizep(1).Text; Spc(5); magcheck(1).Value
Next i
End If

i = 0
```

```
TotalVerticalCellNumber = 0
Do While Check1(i).Value = 1
TotalVerticalCellNumber = TotalVerticalCellNumber + _
    vertical(i).Text
i = i + 1
SectionNumber = i
Loop

Sign = 1
For counter = 0 To SectionNumber - 1
n = SectionNumber - 1 - counter


IsFeedCell = Option1(n).Value
If IsFeedCell <> 0 Then IsFeedCell = 1

For j = 0 To vertical(n).Text - 1
m = TotalVerticalCellNumber - j

Print #3, m; clength(n).Text; cwidth(n).Text; cheight(n).Text; _
    Sign * slope(n).Text; water(n).Text; freq(n).Text; stroke(n).Text; _
    vol(n).Text; IsFeedCell
Sign = Sign * (-1)
IsFeedCell = False

Next j
TotalVerticalCellNumber = TotalVerticalCellNumber - _
    vertical(n).Text

Next counter

Print #4, ratio(0).Text

Close #1
Close #2
Close #3
Close #4

particlesize.Hide
start.Show
End Sub


'Input form for stream velocity condition simulation

Private Sub concel_Click()
streamvelocity.Hide
selection.Show
End Sub



Private Sub ok_Click()
Dim counter As Integer
Dim i As Integer
Dim j As Integer
Dim m As Integer
Dim n As Integer
Dim Sign As Integer
Dim IsFeedCell As Integer
Dim TotalVerticalCellNumber As Integer
Dim SectionNumber As Integer

Open "a:option.dat" For Output As #1
Open "a:minl8.in" For Output As #2
Open "a:cell8.in" For Output As #3
Open "a:grad8.in" For Output As #4
Open "a:condn8.in" For Output As #5

Print #1, 8
Print #5, feedmag.Value; pointno.Text
For i = 0 To 1

Print #2, MineralName(i).Text; Spc(5); density(i).Text; _
      Spc(5); MaxSize(i).Text; Spc(5); sizep(i).Text; Spc(5); magcheck(i).Value
Next i

i = 0
TotalVerticalCellNumber = 0
Do While Check1(i).Value = 1
TotalVerticalCellNumber = TotalVerticalCellNumber + _
    vertical(i).Text
i = i + 1
SectionNumber = i
Loop

Sign = 1
For counter = 0 To SectionNumber - 1
n = SectionNumber - 1 - counter

IsFeedCell = Option1(n).Value
If IsFeedCell <> 0 Then IsFeedCell = 1

For j = 0 To vertical(n).Text - 1
m = TotalVerticalCellNumber - j

Print #3, m; clength(n).Text; cwidth(n).Text; cheight(n).Text; _
    Sign * slope(n).Text; freq(n).Text; stroke(n).Text; _
    vol(n).Text; IsFeedCell

Sign = Sign * (-1)
IsFeedCell = False
Next j

TotalVerticalCellNumber = TotalVerticalCellNumber - _
    vertical(n).Text

Next counter

Print #4, ratio(0).Text

For i = 0 To pointno.Text - 1
```

```
Print #5, lowerlimit.Text + _
   (upperlimit.Text - lowerlimit.Text) * i / (pointno.Text - 1)
Next i

Close #1
Close #2
Close #3
Close #4
Close #5
streamvelocity.Hide
start.Show
End Sub


‘Input form for stroke length condition simulation

Private Sub concel_Click()
strokeform.Hide
selection.Show
End Sub


Private Sub ok_Click()
Dim counter As Integer
Dim i As Integer
Dim j As Integer
Dim m As Integer
Dim n As Integer
Dim Sign As Integer
Dim IsFeedCell As Integer
Dim TotalVerticalCellNumber As Integer
Dim SectionNumber As Integer

Open "a:option.dat" For Output As #1
Open "a:minl9.in" For Output As #2
Open "a:cell9.in" For Output As #3
Open "a:grad9.in" For Output As #4
Open "a:condn9.in" For Output As #5

Print #1, 9
Print #5, feedmag.Value; pointno.Text
For i = 0 To 1

Print #2, MineralName(i).Text; Spc(5); density(i).Text; _
     Spc(5); MaxSize(i).Text; Spc(5); sizep(i).Text; Spc(5); magcheck(i).Value
Next i

i = 0
TotalVerticalCellNumber = 0
Do While Check1(i).Value = 1
TotalVerticalCellNumber = TotalVerticalCellNumber + _
   vertical(i).Text
i = i + 1
SectionNumber = i
Loop
```

```
Sign = 1
For counter = 0 To SectionNumber - 1
n = SectionNumber - 1 - counter

IsFeedCell = Option1(n).Value
If IsFeedCell <> 0 Then IsFeedCell = 1

For j = 0 To vertical(n).Text - 1
m = TotalVerticalCellNumber - j

Print #3, m; clength(n).Text; cwidth(n).Text; cheight(n).Text; _
   Sign * slope(n).Text; water(n).Text; freq(n).Text; _
   vol(n).Text; IsFeedCell

Sign = Sign * (-1)
IsFeedCell = False
Next j
TotalVerticalCellNumber = TotalVerticalCellNumber - _
   vertical(n).Text
Next counter

Print #4, ratio(0).Text

For i = 0 To pointno.Text - 1
Print #5, lowerlimit.Text + _
   (upperlimit.Text - lowerlimit.Text) * i / (pointno.Text - 1)
Next i

Close #1
Close #2
Close #3
Close #4
Close #5
strokeform.Hide
start.Show
End Sub


‘Input form for single simulation test

Private Sub concel_Click()
unittest.Hide
selection.Show
End Sub


Private Sub ok_Click()
Dim counter As Integer
Dim i As Integer
Dim j As Integer
Dim m As Integer
Dim n As Integer

Dim IsFeedCell As Integer
Dim TotalVerticalCellNumber As Integer
Dim SectionNumber As Integer
Dim Sign As Integer
Open "a:option.dat" For Output As #1
Open "a:minl1.in" For Output As #2
Open "a:cell1.in" For Output As #3
Open "a:grad1.in" For Output As #4
Open "a:condn1.in" For Output As #5
Print #1, 1
Print #5, feedmag.Value

For i = 0 To 1

Print #2, MineralName(i).Text; Spc(5); density(i).Text; _
      Spc(5); MaxSize(i).Text; Spc(5); sizep(i).Text; Spc(5); magcheck(i).Value
Next i

i = 0
TotalVerticalCellNumber = 0
Do While Check1(i).Value = 1
TotalVerticalCellNumber = TotalVerticalCellNumber + _
   vertical(i).Text
i = i + 1
SectionNumber = i
Loop

Sign = 1
For counter = 0 To SectionNumber - 1
n = SectionNumber - 1 - counter

IsFeedCell = Option1(n).Value
If IsFeedCell <> 0 Then IsFeedCell = 1

For j = 0 To vertical(n).Text - 1
m = TotalVerticalCellNumber - j
Print #3, m; clength(n).Text; cwidth(n).Text; cheight(n).Text; _
   Sign * slope(n).Text; water(n).Text; freq(n).Text; stroke(n).Text; _
   vol(n).Text; IsFeedCell
Sign = Sign * (-1)
IsFeedCell = False
Next j
TotalVerticalCellNumber = TotalVerticalCellNumber - _
   vertical(n).Text
Next counter

Print #4, ratio(0).Text
Close #1
Close #2
Close #3
Close #4
Close #5
unittest.Hide
start.Show
End Sub



‘Input form for condition simulation of solid packing concentrate
```

```
Private Sub concel_Click()
volume.Hide
selection.Show
End Sub


Private Sub ok_Click()
Dim counter As Integer
Dim i As Integer
Dim j As Integer
Dim m As Integer
Dim n As Integer
Dim Sign As Integer
Dim IsFeedCell As Integer
Dim TotalVerticalCellNumber As Integer
Dim SectionNumber As Integer

Open "a:option.dat" For Output As #1
Open "a:minl10.in" For Output As #2
Open "a:cell10.in" For Output As #3
Open "a:grad10.in" For Output As #4
Open "a:condn10.in" For Output As #5

Print #1, 10
Print #5, feedmag.Value; pointno.Text
For i = 0 To 1

Print #2, MineralName(i).Text; Spc(5); density(i).Text; _
      Spc(5); MaxSize(i).Text; Spc(5); sizep(i).Text; Spc(5); magcheck(i).Value
Next i

i = 0
TotalVerticalCellNumber = 0
Do While Check1(i).Value = 1
TotalVerticalCellNumber = TotalVerticalCellNumber + _
    vertical(i).Text
i = i + 1
SectionNumber = i
Loop

Sign = 1
For counter = 0 To SectionNumber - 1
n = SectionNumber - 1 - counter

IsFeedCell = Option1(n).Value
If IsFeedCell <> 0 Then IsFeedCell = 1

For j = 0 To vertical(n).Text - 1
m = TotalVerticalCellNumber - j

Print #3, m; clength(n).Text; cwidth(n).Text; cheight(n).Text; _
    Sign * slope(n).Text; water(n).Text; freq(n).Text; stroke(n).Text; _
    IsFeedCell

Sign = Sign * (-1)
```

```
IsFeedCell = False

Next j

TotalVerticalCellNumber = TotalVerticalCellNumber - _
    vertical(n).Text

Next counter

Print #4, ratio(0).Text

For i = 0 To pointno.Text - 1
Print #5, lowerlimit.Text + _
    (upperlimit.Text - lowerlimit.Text) * i / (pointno.Text - 1)
Next i

Close #1
Close #2
Close #3
Close #4
Close #5
volume.Hide
start.Show
End Sub



‘Output form for simulation results

Option Explicit

' Variables to represent Excel objects.
Dim ExcelApp As Excel.Application
Dim ExcelWorkbook As Excel.Workbook
Dim ExcelSheet As Excel.Worksheet


Private Sub exits_Click()
output.Hide
start.Show
End Sub

Private Sub Form_Load()
Combo1.AddItem "Single Test"
Combo1.AddItem "Density"
Combo1.AddItem "Frequency"
Combo1.AddItem "Amplitude"
Combo1.AddItem "Stream Velocity"
Combo1.AddItem "Jig Height"
Combo1.AddItem "Particle Size"
Combo1.AddItem "Cell Length"
Combo1.AddItem "Cell Width"
Combo1.AddItem "Cell Slope"
Combo1.AddItem "Both Cell L&W"
Combo1.AddItem "Volume Conc"
Combo1.AddItem "Trajectory"
```

```
End Sub

Private Sub NmuA_Click() ‘ Pulsating amplitude simulation results
Dim var1 As Double
Dim var2 As Double
Dim var3 As Double

Dim i As Integer
Dim pointno As Integer
Set ExcelApp = CreateObject("Excel.application")
Set ExcelWorkbook = ExcelApp.Workbooks.Add
Set ExcelSheet = ExcelWorkbook.Worksheets(1)
ExcelApp.Visible = True


  ' Apply the boldface style to titles and labels.
  ExcelSheet.Range("$A$1:$G$3,$A$3:$A$13").Font.Bold = True

  ' Adjust the width of columns
  ExcelSheet.Columns("A").ColumnWidth = 15.29
  ExcelSheet.Columns("B").ColumnWidth = 10#
  ExcelSheet.Columns("C").ColumnWidth = 10#

  ExcelSheet.Cells(3, 1).Value = "Amplitude"
  ExcelSheet.Cells(3, 2).Value = "Recovery %"
  ExcelSheet.Cells(3, 3).Value = "Grade %"
  ' Copy the data to the sheet.
  Open "a:result9.dat" For Input As #1
  ExcelSheet.Cells(1, 1).Value = "The Effect of Pulsating Amplitude to Separation Result of Yang Jig"
  Input #1, pointno

  For i = 4 To pointno + 3
  Input #1, var1, var2, var3

  ExcelSheet.Cells(i, 1).Value = var1
  ExcelSheet.Cells(i, 2).Value = var2
  ExcelSheet.Cells(i, 3).Value = var3

  Next i
  Close #1
End Sub



Private Sub NmuD_Click() ‘Density simulation results
Dim var1 As Double
Dim var2 As Double
Dim var3 As Double
Dim var4 As Double
Dim i As Integer
Dim pointno As Integer
Set ExcelApp = CreateObject("Excel.application")
Set ExcelWorkbook = ExcelApp.Workbooks.Add
Set ExcelSheet = ExcelWorkbook.Worksheets(1)
ExcelApp.Visible = True
```

```
 ' Apply the boldface style to titles and labels.
  ExcelSheet.Range("$A$1:$G$3,$A$3:$A$13").Font.Bold = True

  ' Adjust the width of columns
  ExcelSheet.Columns("A").ColumnWidth = 15.29
  ExcelSheet.Columns("B").ColumnWidth = 15.29
  ExcelSheet.Columns("C").ColumnWidth = 10#
  ExcelSheet.Columns("D").ColumnWidth = 10#
  ExcelSheet.Cells(3, 1).Value = "        Density Kg/m3"
  ExcelSheet.Cells(4, 1).Value = "Heavy Mineral"
  ExcelSheet.Cells(4, 2).Value = "Light Mineral"
  ExcelSheet.Cells(4, 3).Value = "Recovery %"
  ExcelSheet.Cells(4, 4).Value = "Grade %"
  ' Copy the data to the sheet.
  Open "a:result4.dat" For Input As #1
  ExcelSheet.Cells(1, 1).Value = "The Effect of Density to Separation Result of Yang Jig"
  Input #1, pointno

  For i = 5 To pointno + 4
  Input #1, var1, var2, var3, var4

  ExcelSheet.Cells(i, 1).Value = var1
  ExcelSheet.Cells(i, 2).Value = var2
  ExcelSheet.Cells(i, 3).Value = var3
  ExcelSheet.Cells(i, 4).Value = var4
  Next i
  Close #1
  End Sub

Private Sub NmuExit_Click()
output.Hide
start.Show
End Sub



Private Sub NmuF_Click() ‘ Pulsating frequency simulation result
Dim var1 As Double
Dim var2 As Double
Dim var3 As Double

Dim i As Integer
Dim pointno As Integer
Set ExcelApp = CreateObject("Excel.application")
Set ExcelWorkbook = ExcelApp.Workbooks.Add
Set ExcelSheet = ExcelWorkbook.Worksheets(1)
ExcelApp.Visible = True


  ' Apply the boldface style to titles and labels.
  ExcelSheet.Range("$A$1:$G$3,$A$3:$A$13").Font.Bold = True

  ' Adjust the width of columns
  ExcelSheet.Columns("A").ColumnWidth = 15.29
  ExcelSheet.Columns("B").ColumnWidth = 12#
  ExcelSheet.Columns("C").ColumnWidth = 12#

  ExcelSheet.Cells(3, 1).Value = "Frequency"
  ExcelSheet.Cells(3, 2).Value = "Recovery %"
  ExcelSheet.Cells(3, 3).Value = "Grade %"

  ' Copy the data to the sheet.
  Open "a:result5.dat" For Input As #1
  ExcelSheet.Cells(1, 1).Value = "The Effect of Pulsating Frequency to Separation Result of Yang Jig"
  Input #1, pointno

  For i = 4 To pointno + 3
  Input #1, var1, var2, var3

  ExcelSheet.Cells(i, 1).Value = var1
  ExcelSheet.Cells(i, 2).Value = var2
  ExcelSheet.Cells(i, 3).Value = var3

  Next i
  Close #1
End Sub

Private Sub NmuJ_Click() ‘Jig height simulation result
Dim var1 As Double
Dim var2 As Double
Dim var3 As Double

Dim i As Integer
Dim pointno As Integer
Set ExcelApp = CreateObject("Excel.application")
Set ExcelWorkbook = ExcelApp.Workbooks.Add
Set ExcelSheet = ExcelWorkbook.Worksheets(1)
ExcelApp.Visible = True


  ' Apply the boldface style to titles and labels.
  ExcelSheet.Range("$A$1:$G$3,$A$3:$A$13").Font.Bold = True

  ' Adjust the width of columns
  ExcelSheet.Columns("A").ColumnWidth = 15.29
  ExcelSheet.Columns("B").ColumnWidth = 12#
  ExcelSheet.Columns("C").ColumnWidth = 12#

  ExcelSheet.Cells(3, 1).Value = "Jig Height"
  ExcelSheet.Cells(3, 2).Value = "Recovery %"
  ExcelSheet.Cells(3, 3).Value = "Grade %"

  ' Copy the data to the sheet.
  Open "a:result6.dat" For Input As #1
  ExcelSheet.Cells(1, 1).Value = "The Effect of Jig Height to Separation Result of Yang Jig"
  Input #1, pointno

  For i = 4 To pointno + 3
  Input #1, var1, var2, var3
  ExcelSheet.Cells(i, 1).Value = var1
  ExcelSheet.Cells(i, 2).Value = var2
  ExcelSheet.Cells(i, 3).Value = var3

  Next i
  Close #1
End Sub

Private Sub NmuP_Click() ‘Particle size simulation result
Dim var1 As Double
Dim var2 As Double
Dim var3 As Double
Dim var4 As Double
Dim i As Integer
Dim pointno As Integer
Set ExcelApp = CreateObject("Excel.application")
Set ExcelWorkbook = ExcelApp.Workbooks.Add
Set ExcelSheet = ExcelWorkbook.Worksheets(1)
ExcelApp.Visible = True


  ' Apply the boldface style to titles and labels.
  ExcelSheet.Range("$A$1:$G$3,$A$3:$A$13").Font.Bold = True

  ' Adjust the width of columns
  ExcelSheet.Columns("A").ColumnWidth = 15.29
  ExcelSheet.Columns("B").ColumnWidth = 15.29
  ExcelSheet.Columns("C").ColumnWidth = 12#
  ExcelSheet.Columns("D").ColumnWidth = 12#
  ExcelSheet.Cells(3, 1).Value = "Max. Particle Size (mm)"
  ExcelSheet.Cells(4, 1).Value = "Heavy Mineral"
  ExcelSheet.Cells(4, 2).Value = "Light Mineral"
  ExcelSheet.Cells(4, 3).Value = "Recovery %"
  ExcelSheet.Cells(4, 4).Value = "Grade %"
  ' Copy the data to the sheet.
  Open "a:result7.dat" For Input As #1
  ExcelSheet.Cells(1, 1).Value = "The Effect of Particle Size to Separation Result of Yang Jig"
  Input #1, pointno

  For i = 5 To pointno + 4
  Input #1, var1, var2, var3, var4

  ExcelSheet.Cells(i, 1).Value = var1 * 1000
  ExcelSheet.Cells(i, 2).Value = var2 * 1000
  ExcelSheet.Cells(i, 3).Value = var3
  ExcelSheet.Cells(i, 4).Value = var4
  Next i
  Close #1
End Sub

Private Sub NmuS_Click() ‘Single test simulation result
Dim i As Integer

Dim var1 As Variant
Dim var2 As Double
Dim var3 As Double
Dim var4 As Double


Set ExcelApp = CreateObject("Excel.application")
Set ExcelWorkbook = ExcelApp.Workbooks.Add
Set ExcelSheet = ExcelWorkbook.Worksheets(1)
ExcelApp.Visible = True


  ' Apply the boldface style to titles and labels.
  ExcelSheet.Range("$A$1:$G$3,$A$3:$A$13").Font.Bold = True

  ' Adjust the width of columns
  ExcelSheet.Columns("A").ColumnWidth = 15.29
  ExcelSheet.Columns("B").ColumnWidth = 10#
  ExcelSheet.Columns("C").ColumnWidth = 10#
  ExcelSheet.Columns("D").ColumnWidth = 10#
  ExcelSheet.Cells(3, 1).Value = "Size Range"
  ExcelSheet.Cells(3, 2).Value = "Yield %"
  ExcelSheet.Cells(3, 3).Value = "Grade %"
  ExcelSheet.Cells(3, 4).Value = "Recovery %"
  ' Copy the data to the sheet.
  Open "a:result1.dat" For Input As #1
  ExcelSheet.Cells(1, 2).Value = "A Simulation Test Result of Yang Jig"

  For i = 4 To 12
  Input #1, var1, var2, var3, var4

  ExcelSheet.Cells(i, 1).Value = var1
  ExcelSheet.Cells(i, 2).Value = var2
  ExcelSheet.Cells(i, 3).Value = var3
  ExcelSheet.Cells(i, 4).Value = var4
  Next i

  Close #1

End Sub

Private Sub NmuT_Click() ‘Particle trajectory result
Dim var1 As Double
Dim var2 As Double
Dim var3 As Double

Dim i As Integer
Dim pointno As Integer
Set ExcelApp = CreateObject("Excel.application")
Set ExcelWorkbook = ExcelApp.Workbooks.Add
Set ExcelSheet = ExcelWorkbook.Worksheets(1)
ExcelApp.Visible = True


  ' Apply the boldface style to titles and labels.
  ExcelSheet.Range("$A$1:$G$3").Font.Bold = True

  ' Adjust the width of columns
  ExcelSheet.Columns("A").ColumnWidth = 12
  ExcelSheet.Columns("B").ColumnWidth = 12#
  ExcelSheet.Columns("C").ColumnWidth = 12#

  ExcelSheet.Cells(3, 2).Value = "    Distance from feedpoint"
  ExcelSheet.Cells(4, 1).Value = "Time"
  ExcelSheet.Cells(4, 2).Value = "Particle1"
  ExcelSheet.Cells(4, 3).Value = "Particle2"

  ' Copy the data to the sheet.
  Open "a:result2.dat" For Input As #1
  ExcelSheet.Cells(1, 1).Value = "       The Particle Trajectory in Yang Jig"
  i = 5
  Input #1, var1, var2, var3
  Do
  ExcelSheet.Cells(i, 1).Value = var1
  ExcelSheet.Cells(i, 2).Value = var2
  ExcelSheet.Cells(i, 3).Value = var3
  Input #1, var1, var2, var3
  i = i + 1
  Loop While var1 <> 999
  Close #1
End Sub

Private Sub NmuV_Click() 'Stream velocity simulation results
Dim var1 As Double
Dim var2 As Double
Dim var3 As Double
Dim i As Integer
Dim pointno As Integer
Set ExcelApp = CreateObject("Excel.application")
Set ExcelWorkbook = ExcelApp.Workbooks.Add
Set ExcelSheet = ExcelWorkbook.Worksheets(1)
ExcelApp.Visible = True
'DrawExcelChart

  ' Apply the boldface style to titles and labels.
  ExcelSheet.Range("$A$1:$G$3,$A$3:$A$13").Font.Bold = True

  ' Adjust the width of columns
  ExcelSheet.Columns("A").ColumnWidth = 15.29
  ExcelSheet.Columns("B").ColumnWidth = 12#
  ExcelSheet.Columns("C").ColumnWidth = 12#

  ExcelSheet.Cells(3, 1).Value = "Stream Velocity"
  ExcelSheet.Cells(3, 2).Value = "Recovery %"
  ExcelSheet.Cells(3, 3).Value = "Grade %"

  ' Copy the chart title and the units to the sheet.
  Open "a:result8.dat" For Input As #1
  ExcelSheet.Cells(1, 1).Value = "The Effect of Stream Velocity to Separation Result of Yang Jig"
  Input #1, pointno

  For i = 4 To pointno + 3
  Input #1, var1, var2, var3

  ExcelSheet.Cells(i, 1).Value = var1
  ExcelSheet.Cells(i, 2).Value = var2
  ExcelSheet.Cells(i, 3).Value = var3

  Next i
  Close #1
End Sub

Private Sub NmuVc_Click() ‘Particle packing concentrate simulation results
Dim var1 As Double
Dim var2 As Double
Dim var3 As Double
Dim i As Integer
Dim pointno As Integer
Set ExcelApp = CreateObject("Excel.application")
Set ExcelWorkbook = ExcelApp.Workbooks.Add
Set ExcelSheet = ExcelWorkbook.Worksheets(1)
ExcelApp.Visible = True

  ' Apply the boldface style to titles and labels.
  ExcelSheet.Range("$A$1:$G$3,$A$3:$A$13").Font.Bold = True

  ' Adjust the width of columns
  ExcelSheet.Columns("A").ColumnWidth = 15.29
  ExcelSheet.Columns("B").ColumnWidth = 12#
  ExcelSheet.Columns("C").ColumnWidth = 12#

  ExcelSheet.Cells(3, 1).Value = "Volume Conc. %"
  ExcelSheet.Cells(3, 2).Value = "Recovery %"
  ExcelSheet.Cells(3, 3).Value = "Grade %"

  ' Copy the chart title and the units to the sheet.
  Open "a:result10.dat" For Input As #1
  ExcelSheet.Cells(1, 1).Value = "The Effect of Particle Volume Concentrate to Separation Result of Yang
Jig"
  Input #1, pointno

  For i = 4 To pointno + 3
  Input #1, var1, var2, var3

  ExcelSheet.Cells(i, 1).Value = var1
  ExcelSheet.Cells(i, 2).Value = var2
  ExcelSheet.Cells(i, 3).Value = var3

  Next i
  Close #1
End Sub

Private Sub oks_Click() ‘Selections of output
If Combo1.Text = "Single Test" Then
NmuS_Click
ElseIf Combo1.Text = "Density" Then NmuD_Click
ElseIf Combo1.Text = "Frequency" Then NmuF_Click
ElseIf Combo1.Text = "Volume Conc" Then NmuVc_Click
ElseIf Combo1.Text = "Both Cell L&W" Then subnum4_Click
ElseIf Combo1.Text = "Amplitude" Then NmuA_Click
ElseIf Combo1.Text = "Stream Velocity" Then NmuV_Click
ElseIf Combo1.Text = "Jig Height" Then NmuJ_Click
ElseIf Combo1.Text = "Particle Size" Then NmuP_Click
ElseIf Combo1.Text = "Cell Length" Then subnum1_Click
ElseIf Combo1.Text = "Cell Width" Then subnum2_Click
ElseIf Combo1.Text = "Cell Slope" Then subnum3_Click
ElseIf Combo1.Text = "Trajectory" Then NmuT_Click
End If
End Sub


Private Sub subnum1_Click() ‘ Cell length simulation results
Dim var1 As Double
Dim var2 As Double
Dim var3 As Double

Dim i As Integer
Dim pointno As Integer

Set ExcelApp = CreateObject("Excel.application")
Set ExcelWorkbook = ExcelApp.Workbooks.Add
Set ExcelSheet = ExcelWorkbook.Worksheets(1)
ExcelApp.Visible = True


  ' Apply the boldface style to titles and labels.
  ExcelSheet.Range("$A$1:$G$3,$A$3:$A$13").Font.Bold = True

  ' Adjust the width of columns
  ExcelSheet.Columns("A").ColumnWidth = 15.29
  ExcelSheet.Columns("B").ColumnWidth = 12#
  ExcelSheet.Columns("C").ColumnWidth = 12#


  ExcelSheet.Cells(3, 1).Value = "Length"
  ExcelSheet.Cells(3, 2).Value = "Recovery %"
  ExcelSheet.Cells(3, 3).Value = "Grade %"

  ' Copy the data to the sheet.
  Open "a:result31.dat" For Input As #1
  ExcelSheet.Cells(1, 1).Value = "The Effect of Cell Length to Separation Result of Yang Jig"
  Input #1, pointno

  For i = 4 To pointno + 3
  Input #1, var1, var2, var3

  ExcelSheet.Cells(i, 1).Value = var1
  ExcelSheet.Cells(i, 2).Value = var2
  ExcelSheet.Cells(i, 3).Value = var3

  Next i
  Close #1
End Sub

Private Sub subnum2_Click() ‘ Cell width simulation results
Dim var1 As Double
Dim var2 As Double
Dim var3 As Double

Dim i As Integer
Dim pointno As Integer

Set ExcelApp = CreateObject("Excel.application")
Set ExcelWorkbook = ExcelApp.Workbooks.Add
Set ExcelSheet = ExcelWorkbook.Worksheets(1)
ExcelApp.Visible = True


  ' Apply the boldface style to titles and labels.
  ExcelSheet.Range("$A$1:$G$3,$A$3:$A$13").Font.Bold = True

  ' Adjust the width of columns
  ExcelSheet.Columns("A").ColumnWidth = 12
  ExcelSheet.Columns("B").ColumnWidth = 12#
  ExcelSheet.Columns("C").ColumnWidth = 12#

  ExcelSheet.Cells(3, 1).Value = "Width"
  ExcelSheet.Cells(3, 2).Value = "Recovery %"
  ExcelSheet.Cells(3, 3).Value = "Grade %"

  ' Copy the data to the sheet.
  Open "a:result32.dat" For Input As #1
  ExcelSheet.Cells(1, 1).Value = "The Effect of Cell Width to Separation Result of Yang Jig"
  Input #1, pointno

  For i = 4 To pointno + 3
  Input #1, var1, var2, var3

  ExcelSheet.Cells(i, 1).Value = var1
  ExcelSheet.Cells(i, 2).Value = var2
  ExcelSheet.Cells(i, 3).Value = var3

  Next i
  Close #1
End Sub

Private Sub subnum3_Click() ‘Cell slope simulation results
Dim var1 As Double
Dim var2 As Double
Dim var3 As Double

Dim i As Integer
Dim pointno As Integer

Set ExcelApp = CreateObject("Excel.application")
Set ExcelWorkbook = ExcelApp.Workbooks.Add
Set ExcelSheet = ExcelWorkbook.Worksheets(1)
ExcelApp.Visible = True


  ' Apply the boldface style to titles and labels.
  ExcelSheet.Range("$A$1:$G$3,$A$3:$A$13").Font.Bold = True
  ' Adjust the width of columns
  ExcelSheet.Columns("A").ColumnWidth = 12
  ExcelSheet.Columns("B").ColumnWidth = 12#
  ExcelSheet.Columns("C").ColumnWidth = 12#


  ExcelSheet.Cells(3, 1).Value = "Slope"
  ExcelSheet.Cells(3, 2).Value = "Recovery %"
  ExcelSheet.Cells(3, 3).Value = "Grade %"

  ' Copy the data to the sheet.
  Open "a:result33.dat" For Input As #1
  ExcelSheet.Cells(1, 1).Value = "The Effect of Cell Slope to Separation Result of Yang Jig"
  Input #1, pointno

  For i = 4 To pointno + 3
  Input #1, var1, var2, var3

  ExcelSheet.Cells(i, 1).Value = var1
  ExcelSheet.Cells(i, 2).Value = var2
  ExcelSheet.Cells(i, 3).Value = var3

  Next i
  Close #1
End Sub

Private Sub subnum4_Click() ‘ Cell Length & Width simulation results
Dim var1 As Double
Dim var2 As Double
Dim var3 As Double

Dim i As Integer
Dim pointno As Integer

Set ExcelApp = CreateObject("Excel.application")
Set ExcelWorkbook = ExcelApp.Workbooks.Add
Set ExcelSheet = ExcelWorkbook.Worksheets(1)
ExcelApp.Visible = True


  ' Apply the boldface style to titles and labels.
  ExcelSheet.Range("$A$1:$G$3,$A$3:$A$13").Font.Bold = True

  ' Adjust the width of columns
  ExcelSheet.Columns("A").ColumnWidth = 12
  ExcelSheet.Columns("B").ColumnWidth = 12#
  ExcelSheet.Columns("C").ColumnWidth = 12#


  ExcelSheet.Cells(3, 1).Value = "L&W  m"
  ExcelSheet.Cells(3, 2).Value = "Recovery %"
  ExcelSheet.Cells(3, 3).Value = "Grade %"

  ' Copy the data to the sheet.
  Open "a:result34.dat" For Input As #1
  ExcelSheet.Cells(1, 1).Value = "The Effect of Cell Length & Width to Separation Result of Yang Jig"
  Input #1, pointno

  For i = 4 To pointno + 3
  Input #1, var1, var2, var3

  ExcelSheet.Cells(i, 1).Value = var1
  ExcelSheet.Cells(i, 2).Value = var2
  ExcelSheet.Cells(i, 3).Value = var3

  Next i
  Close #1
End Sub
```

Appendix B: C/C++ Code For Simulation Of Packed Column Jig

1. Head File

```
/*************************************************************/
/*     The head file for Packed Column Jig Simulation        */
/*                                                           */
/*             File name: jig_m.h                            */
/*             Designed by Qiang Dai                         */
/*             Advisor: Dr. David C. Yang                    */
/*             Date: Jan. 1999                               */
/*                                                           */
/************************************************************/
#include<conio.h>
#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <graphics.h>

enum boolean {false=0,true};

class Particle;

class Mineral  //define the Mineral class
{
    private:

    int mineral_ID;
    double density;
    double max_size;
    double dis_const;
    boolean is_mag;

    public:

    Mineral(); //constructor

      void set(int ID,double dnst,double msz,double tp,boolean ism);

      int get_mineral_ID(){return mineral_ID;};

      boolean get_is_mag(){return is_mag;};

      double get_density(){return density;};

      double get_dis_const(){return dis_const;};

      double get_max_size(){return max_size;};

      double get_avg_size();

      void show();
};
```

```
class Vcell   //define the class to express the cell in Yang Jig
{
    private:

    int cell_ID;
    double length;
    double width;
    double height;
    double slope;
    double velocity;
    double frequency;
    double amplitude;
    double V_conc;
    boolean is_feed_cell;

    public:

    Vcell(); // constructor

    void set(int no,double lngth,double wdth,double hght,double slp,
               double vlcty,double freq, double ampl,double vc,boolean ifc);

    double get_V_conc(){return V_conc;};

    int get_cell_ID(){return cell_ID;};

    void get_a(double x,double y,double z,double time,double* ax,
               double* ay,double* az);

    void get_U(double x,double y,double z,double time,double* ux,
               double* uy,double* uz);

    double get_shear(double x, double y, double z,double time);

    double get_H(){return height;};

    double Calc_Vm();

    double get_length(){return length;};

    double get_width(){return width;};

    double get_slope(){return slope;};

    boolean Is_F_cell(){return is_feed_cell;};

    void show();

    void draw(int x0, int y0);

};


class Particle   //define the Particle class
{
    private:

    Mineral* Pm;
    Vcell* Pv;
    double size;
    double x;
    double y;
    double z;
    double Vx;
    double Vy;
    double Vz;
    double time;

    public:

    Particle();//constructor

    void set(Mineral*pm,Vcell*pv, double sz,double x0,double y0,
           double z0, double vx, double vy, double vz,double t);

    double get_time(){return time;};

    double get_size(){return size;};

    double get_y(){return y;};

    Mineral* get_M(){return Pm;};

    void transfer(Vcell* pv, int* signal);

    void Move(double avg_density,double avgsize, boolean mag);

    int check_boundary();

    Vcell* get_cell(){return Pv;};

    void Get_Position(double* px,double* py, double* pz)
        {   *px=x;  *py=y;  *pz=z;  };

    void Get_V(double* pvx,double* pvy, double* pvz)
            {   *pvx=Vx; *pvy=Vy; *pvz=Vz;      };

    void Show();

    void erase(int x0, int y0);

    void draw(int x0, int y0,int color);

};



struct Jig_Node  // The Node for storing the virtual jig cell
{
    Vcell cell;
    Jig_Node * next; // pointer pointed to the next Jig_Node
    Jig_Node * prev; // pointer pointed to the previous Jig_Node
    Jig_Node(){ next=NULL;prev=NULL;};//constructor
      void set(Vcell cell1){ cell=cell1;next=NULL;prev=NULL;};
};


struct Jig_Node_List
{
    Jig_Node* head;

    Jig_Node_List(){ head=NULL;};//constructor

    ~Jig_Node_List();            //destructor

    void add_Cell( Jig_Node* cell);

    void remove_all();
};


class JigSystem     //define the class for Yang Jig
{
      private:

      Mineral* Heavy;
      Mineral* Light;
      Jig_Node_List* Head;
      unsigned cell_No;
      Jig_Node* pfc; //pointer to the Jig_Node containing the feed cell
      double feed_grad;
      boolean mag_feed;

      public:

      JigSystem() { Heavy=NULL; Light=NULL;Head=NULL;
                    cell_No=0;pfc=NULL;feed_grad=50.0;mag_feed=false;};
      void set(Mineral* ph, Mineral* pl,Jig_Node_List* JNlist, unsigned
      cellno,
              Jig_Node* pf,double grad,boolean mag);
      Vcell get_feedcell(){return pfc->cell; };
      void simulate( FILE *pp,int part_no);

      void multi_sim(FILE *pp,int part_no);
      void Observer( FILE* pp,Particle* part1, Particle* part2,
                        int x0, int y0, double TT);

      Jig_Node_List* get_Head(){return Head;};

      void show();

      void draw(int x0, int y0);
 };
```

2. CPP File

```
/**************************************************************/
/*     The cpp file for Packed Column Jig Simulation          */
/*                                                            */
/*             File name: jig_m.cpp                           */
/*             Designed by Qiang Dai                          */
/*             Advisor: Dr. David C. Yang                     */
/*             Date: Jan. 1999                                */
/*                                                            */
/*************************************************************/

#include "jig_m.h"
const double K=10.0; //enlargement times of particle
const double MK=0.10;//Mechanic force constant
const double MS=0.006;//coefficient for Bagnold force
const double NO_STEP=10.0;
const double XK=0.2;//constant when draw cell
const double VISCOCITY=0.00112;
const double TIMES=1200;//enlarge times for display
const double PI=3.1416;
const double N=7.0;//constant for the velocity profile
const double STEP=20.0; // The time step coefficient
const double Total_time=120.0; //the total simulation time
const double A_MAG=9.8;//the acceleration from magnetic force
const double TT=0.01; //the period when computing stream acceleration


Mineral::Mineral()
{
    mineral_ID=1;
    density=0.0;
    max_size=0.0;
    dis_const=3.0;
    is_mag=false;
}

//function for setting a mineral
void Mineral::set(int ID,double dnst,double msz,double tp,boolean ism)
{
    mineral_ID=ID;
    density=dnst;
    max_size=msz;
    dis_const=tp;
    is_mag=ism;
}

//function for getting the average size of a mineral
double Mineral::get_avg_size()
{
      return(dis_const*max_size/(dis_const+1.0));
}
```

```
//function to show the mineral
void Mineral::show()
{
      printf("\nShow the message of Mineral %d:",mineral_ID);
      printf("\nDensity=%8.2f kg/m3", density);
      printf("\nThe maximum size=%8.5fmm", max_size*1000.0);
      printf("\nThe size distribution fits Gaudin-Schulman Equation");
      printf("\nWith distribution constant equal %8.5f.",dis_const);
      if(is_mag) printf("\nThe mineral is magnetic.");
      else printf("\nThe mineral is non-magnetic.");
}

Vcell::Vcell()
{
    cell_ID=1;
    length=0.0;
    width=0.0;
    height=0.0;
    slope=1.0;
    velocity=0.0;
    frequency=0.0;
    amplitude=0.0;
    V_conc=0.0;
    is_feed_cell=false;
};

//function for setting a cell
void Vcell::set(int no,double lngth,double wdth,double hght,double slp,
               double vlcty,double freq, double ampl,double vc,boolean ifc)
{
    cell_ID=no;
    length=lngth;
    width=wdth;
    height=hght;
    slope=slp;
    velocity=vlcty;
    frequency=freq;
    amplitude=ampl;
    V_conc=vc;
    is_feed_cell=ifc;
}

//function for getting the stream acceleration in cell
void Vcell::get_a(double x,double y,double z,double time,double* ax,
                  double* ay,double* az)
{
      double uc,u,uu,R,r,temp,temp1,temp2,kk;
      kk=(N+1.0)*(2.0*N+1.0)/(2.0*N*N);
      uc=-kk*(velocity+
            2.0*PI*frequency*amplitude*sin(2.0*PI*frequency*time));
      R=0.5*sqrt(length*length+width*width);
      temp=(x-length/2.0-y/slope)*slope/sqrt(1.0+slope*slope);
      r=sqrt(temp*temp+(z-0.5*width)*(z-0.5*width));
      temp1=fabs(1.0-r/R);
      temp2=1.0/N;
      temp1=pow(temp1,temp2);
      uu=0.12*temp1*uc;
    temp=sqrt(1+slope*slope);
    if (slope>0)
    {
            *ax=-temp1/temp*kk*4.0*PI*PI*frequency*frequency*amplitude*
                  cos(2.0*PI*frequency*time)+((float)random(101)-
                  50.0)/50.0*uu/TT;
            *ay=-slope*temp1/temp*kk*4.0*PI*PI*frequency*frequency*amplitude*
                  cos(2.0*PI*frequency*time)+((float)random(101)-
                  50.0)/50.0*uu/TT;
            *az=((float)random(101)-50.0)/50.0*uu/TT;
     }
    else
     {
            *ax=temp1/temp*kk*4.0*PI*PI*frequency*frequency*amplitude*
                  cos(2.0*PI*frequency*time)+((float)random(101)-
                  50.0)/50.0*uu/TT;
            *ay=slope*temp1/temp*kk*4.0*PI*PI*frequency*frequency*amplitude*
                  cos(2.0*PI*frequency*time)+((float)random(101)-
                  50.0)/50.0*uu/TT;
            *az=((float)random(101)-50.0)/50.0*uu/TT;

      };
}


//function for getting the stream velocity in cell
void Vcell::get_U(double x, double y, double z,
                          double time, double* ux, double* uy, double* uz)
{

      double uc,u, uu,R,r,temp,temp1,temp2,temp3;
      uc=-(N+1.0)*(2.0*N+1.0)/(2.0*N*N)*(velocity+
      2.0*PI*frequency*amplitude*sin(2.0*PI*frequency*time));
      R=0.5*sqrt(length*length+width*width);
      temp=sqrt(1.0+slope*slope);
      temp3=(x-length/2.0-y/slope)*slope/temp;
      r=sqrt(temp3*temp3+(z-0.5*width)*(z-0.5*width));
      temp1=fabs(1.0-r/R);
      temp2=1.0/N;
      u=uc*pow(temp1,temp2);
      uu=0.12*u;//0.04*u*3.0

      if (slope>0.0)
      {
            *ux=u/temp+uu*((float)random(101)-50.0)/50.0;
            *uy=u*slope/temp+uu*((float)random(101)-50.0)/50.0;
            *uz=uu*((float)random(101)-50.0)/50.0;
      }
      else
      {
            *ux=-u/temp+uu*((float)random(101)-50.0)/50.0;
            *uy=-u*slope/temp+uu*((float)random(101)-50.0)/50.0;
            *uz=uu*((float)random(101)-50.0)/50.0;
      };
}
```

```
//function for getting the stream shear strength in cell
double Vcell::get_shear(double x, double y, double z, double time)
{
      double uc,shear,R,r,temp,temp1,temp2,temp3,temp4;
      uc=-(N+1.0)*(2.0*N+1.0)/(2.0*N*N)*(velocity+
      2.0*PI*frequency*amplitude*sin(2.0*PI*frequency*time));
      R=0.5*sqrt(length*length+width*width);
      temp=sqrt(1.0+slope*slope);
      temp4=x-length/2.0-y/slope;
      temp3=temp4*slope/temp;
      r=sqrt(temp3*temp3+(z-0.5*width)*(z-0.5*width));
      temp1=fabs(1.0-r/R);
      temp2=1.0/N-1.0;
      shear=-uc/N/R*pow(temp1,temp2);
      return fabs(shear);
}


//function for returning the average pulsing velocity
double Vcell::Calc_Vm(){return 2*PI*frequency*amplitude;}


//function to show a cell
void Vcell::show()
{
      printf("\n Show the cell message of Jig:\n");
      printf("\nThe cell_ID of the cell is %d",cell_ID);
      if(is_feed_cell) printf("   This is the feed cell!");
      printf("\nlength=%8.5f, width=%8.5f, height=%8.5f, slope=%8.5f",
                  length,width,height,slope);
      printf("\nvelocity=%8.5f, frequency=%8.5f, amplitude=%8.5f",
                  velocity,frequency,amplitude);
      printf("\nThe supposed volume concentrate is %8.5f",V_conc);
}


//function to draw the cell in screen
void Vcell::draw(int x0, int y0)
{
         int x1,y1,x2,y2,x3,y3,x4,y4;
         x1=x0-(int)(length*XK*TIMES);
         y1=y0;
         x2=x0+(int)(length*(1+XK)*TIMES);
         y2=y1;
         x3=x0+(int)(length*(1+XK)*TIMES+height*TIMES/slope);
         y3=y0+(int)(TIMES*height);
         x4=x0+(int)((height/slope-length*XK)*TIMES);
         y4=y3;
         int Vcell_dem[] ={x1,y1,x2,y2,x3,y3,x4,y4};
         setcolor(BLACK);                      // line color
         setlinestyle(SOLID_LINE, 0, THICK_WIDTH); // line width
         setfillstyle(SOLID_FILL, YELLOW);   // set fill color
         fillpoly(4, Vcell_dem);                // draw Vcell
```

```
}



Particle::Particle()
{
    Pm=NULL;
    Pv=NULL;
    size=0.0;
    x=0.0;
    y=0.0;
    z=0.0;
    Vx=0.0;
    Vy=0.0;
    Vz=0.0;
    time=0.0;
}

void Particle::set(Mineral* pm,Vcell* pv,double sz,double x0,double y0,
                           double z0,double vx, double vy, double vz,double
t)
{
    Pm=pm;
    Pv=pv;
    size=sz;
    x=x0;
    y=y0;
    z=z0;
    Vx=vx;
    Vy=vy;
    Vz=vz;
    time=t;
}

//function for particle to transfer from one cell to another
void Particle::transfer(Vcell* pv, int* signal)
{

      if (*signal==-1) //go down
      {
            y=y-Pv->get_H();
            if(y>pv->get_H()) y=pv->get_H();
            x=y/pv->get_slope()+size/2.0+(pv->get_length()-
            size)*random(101)/100.0;
            z=size/2.0+(pv->get_width()-size)*random(101)/100.0;
            Pv=pv;
            Pv->get_U(x,y,z,time,&Vx,&Vy,&Vz);
       }

      else if(*signal==1)   //go up
      {
            y=pv->get_H()+y;
            if(y<0) y=0.0;
            x=y/pv->get_slope()+size/2.0+
              (pv->get_length()-size)*random(101)/100.0;
            z=size/2.0+(pv->get_width()-size)*random(101)/100.0;
            Pv=pv;
            Pv->get_U(x,y,z,time,&Vx,&Vy,&Vz);
        };
   return;
}

//function of for checking the boundary of the particle
int Particle::check_boundary()
{
      if(y>Pv->get_H()) return -1;
      if (y<0) return 1;
      else
      {
            double xleft, xright,zfront,zback;

            xright=Pv->get_length()+y/(Pv->get_slope())-size/2.0;
            xleft=xright-Pv->get_length()+size/2.0;
            zfront=Pv->get_width()-size/2.0;
            zback=size/2.0;
            if(x>xright) x=xright;
            if(x<xleft) x=xleft;
            if(z>zfront) z=zfront;
            if(z<zback) z=zback;

            return 0;
      };
}


//function for particle to move one step in cell
void Particle::Move(double avg_density,double avgsize, boolean mag)
{
      int i;
      double step,vx,vy,vz,k[4][3],ux1,uy1,uz1,ax,ay,az,Ax,Ay,Az,C,
               dp,ds,dt,vc,temp,temp11,temp12,temp13,grv,
               shear_x,shear_y,shear_z;

      step=size*STEP;
      vc=Pv->get_V_conc()/100.0;
      C=1.0/(pow(0.65/vc,0.33)-1.0);//linear solid concentrate

      shear_x=pow(Pv->get_shear(x-size/2.0,y,z,time),2)-
                  pow(Pv->get_shear(x+size/2.0,y,z,time),2);
      shear_y=pow(Pv->get_shear(x,y-size/2.0,z,time),2)-
                  pow(Pv->get_shear(x,y+size/2.0,z,time),2);
      shear_z=pow(Pv->get_shear(x,y,z-size/2.0,time),2)-
                  pow(Pv->get_shear(x,y,z+size/2.0,time),2);

      ds=Pm->get_density();
      dp=1000.0+(avg_density-1000.0)*vc;
      dt=ds+0.05*dp;
      temp=MK*avg_density*vc*Pv->Calc_Vm()/size/dt;
      temp11=MS*avg_density*pow(C*avgsize,2)*shear_x/size/dt;
      temp12=MS*avg_density*pow(C*avgsize,2)*shear_y/size/dt;
      temp13=MS*avg_density*pow(C*avgsize,2)*shear_z/size/dt;

      Pv->get_U(x,y,z,time,&ux1,&uy1,&uz1);
      Pv->get_a(x,y,z,time,&ax,&ay,&az);
      grv=(ds-dp)*9.8/dt;

      vx=Vx;
      vy=Vy;
      vz=Vz;

//computing the acceleration of the particle in x,y,z directions
      Ax=(ux1-vx)*fabs(ux1-vx)*dp/(3.0*size*dt)+dp*ax/dt+
            temp*((float)random(101)-50.0)/50.0*vx+temp11;
      Az=(uz1-vz)*fabs(uz1-vz)*dp/(3.0*size*dt)+dp*az/dt+
            temp*((float)random(101)-50.0)/50.0*vz+temp13;
      Ay=grv+(uy1-vy)*fabs(uy1-vy)*dp/(3.0*size*dt)+dp*ay/dt+
            ((float)random(101)-50.0)/50.0*vy*temp+temp12;
      if (Pm->get_is_mag()&&mag)
            Ay=Ay+A_MAG;

      k[0][0]=Ax*step;
      k[0][1]=Ay*step;
      k[0][2]=Az*step;

      Pv->get_U(x,y,z,time+step/2.0,&ux1,&uy1,&uz1);

      vx=Vx+k[0][0]/2.0;
      vy=Vy+k[0][1]/2.0;
      vz=Vz+k[0][2]/2.0;

      Ax=(ux1-vx)*fabs(ux1-vx)*dp/(3.0*size*dt)+dp*ax/dt+
            temp*((float)random(101)-50.0)/50.0*vx+temp11;
      Az=(uz1-vz)*fabs(uz1-vz)*dp/(3.0*size*dt)+dp*az/dt+
            temp*((float)random(101)-50.0)/50.0*vz+temp13;
      Ay=grv+(uy1-vy)*fabs(uy1-vy)*dp/(3.0*size*dt)+
            dp*ay/dt+((float)random(101)-50.0)/50.0*vy*temp+temp12;
      if (Pm->get_is_mag()&&mag)
            Ay=Ay+A_MAG;

      k[1][0]=Ax*step;
      k[1][1]=Ay*step;
      k[1][2]=Az*step;

      vx=Vx+k[1][0]/2.0;
      vy=Vy+k[1][1]/2.0;
      vz=Vz+k[1][2]/2.0;

      Ax=(ux1-vx)*fabs(ux1-vx)*dp/(3.0*size*dt)+dp*ax/dt+
            temp*((float)random(101)-50.0)/50.0*vx+temp11;
      Az=(uz1-vz)*fabs(uz1-vz)*dp/(3.0*size*dt)+dp*az/dt+
            temp*((float)random(101)-50.0)/50.0*vz+temp13;
      Ay=grv+(uy1-vy)*fabs(uy1-vy)*dp/(3.0*size*dt)+
         dp*ay/dt+((float)random(101)-50.0)/50.0*vy*temp+temp12;
      if (Pm->get_is_mag()&&mag)
            Ay=Ay+A_MAG;

      k[2][0]=Ax*step;
      k[2][1]=Ay*step;
      k[2][2]=Az*step;

      Pv->get_U(x,y,z,time+step,&ux1,&uy1,&uz1);
      vx=Vx+k[2][0]/2.0;
      vy=Vy+k[2][1]/2.0;
      vz=Vz+k[2][2]/2.0;

      Ax=(ux1-vx)*fabs(ux1-vx)*dp/(3.0*size*dt)+dp*ax/dt+
            temp*((float)random(101)-50.0)/50.0*vx+temp11;
      Az=(uz1-vz)*fabs(uz1-vz)*dp/(3.0*size*dt)+dp*az/dt+
            temp*((float)random(101)-50.0)/50.0*vz+temp13;
      Ay=grv+(uy1-vy)*fabs(uy1-vy)*dp/(3.0*size*dt)+
         dp*ay/dt+((float)random(101)-50.0)/50.0*vy*temp+temp12;
      if (Pm->get_is_mag()&&mag)
         Ay=Ay+A_MAG;

      k[3][0]=Ax*step;
      k[3][1]=Ay*step;
      k[3][2]=Az*step;

      x=x+Vx*step/2.0;
      y=y+Vy*step/2.0;
      z=z+Vz*step/2.0;

//computing the velocity after one step movement
      Vx=Vx+(k[0][0]+2.0*k[1][0]+2.0*k[2][0]+k[3][0])/6.0;
      Vy=Vy+(k[0][1]+2.0*k[1][1]+2.0*k[2][1]+k[3][1])/6.0;
      Vz=Vz+(k[0][2]+2.0*k[1][2]+2.0*k[2][2]+k[3][2])/6.0;

//computing the new position after the movement
      x=x+Vx*step/2.0;
      y=y+Vy*step/2.0;
      z=z+Vz*step/2.0;
      time=time+step;

}


//function for showing the message of a particle
void Particle::Show()
{
      printf("\nThe message of the particle is as following:\n");
      printf("\nThe mineral_ID of the particle is: %d",
                   Pm->get_mineral_ID());
      printf("\nThe density of the particle is : %8.2f",Pm->get_density());
      printf("\nThe particle size is: %8.2f micron", size*1000000.0);
      printf("\nThe particle is in cell %d, with x=%8.5f, y=%8.5f, z=%8.5f",
                  Pv->get_cell_ID(),x,y,z);
      printf("\nThe velocity of the particle is Vx=%8.5f,
             Vy=%8.5f, Vz=%8.5f",Vx,Vy,Vz);
      printf("\nThe simulation time is %8.5f seconds",time);
}

//function for ereasing a particle in screen
void Particle::erase(int x0, int y0)
{
       int X,Y,R;
       X=x0+(int)(TIMES*x);
       Y=y0+(int)(TIMES*y);
       R=(int)(K*TIMES*size);
       if(R<1) R=1;
       setcolor(YELLOW);
       circle(X,Y,R);
       setfillstyle(SOLID_FILL, YELLOW);
       floodfill(X,Y,YELLOW);
}

//function for drawing a particle in screen
void Particle::draw(int x0,int y0,int color)
{
       int X,Y,R;
       X=x0+(int)(TIMES*x);
       Y=y0+(int)(TIMES*y);
       R=(int)(K*TIMES*size);
       if(R<1) R=1;
       setcolor(color);
       circle(X,Y,R);
       setfillstyle(SOLID_FILL, color);
       floodfill(X, Y, color);
}


//destructor of Jig_Node_List
Jig_Node_List::~Jig_Node_List()
{
      if(head==NULL) return;
      else if(head->next!=NULL)
      while(head->next!=NULL)
      {
            Jig_Node * temp;
            temp=head;
            head=head->next;
            temp->next->prev=head;
            delete temp;
      };

            delete head;
}


//function for removing all objects in a Jig_Node_List
void Jig_Node_List::remove_all()
{
      if (head==NULL) return;
      else if(head->next!=NULL)
      while(head->next!=NULL)
      {
            Jig_Node * temp;
            temp=head;
            head=head->next;
            temp->next->prev=head;
            delete temp;
      };

      if(head->next==NULL)
      {
            Jig_Node * temp;
            temp=head;
            head=NULL;
            delete temp;
      };

}


//function for adding a Jig_Node to Jig_Node_List
void Jig_Node_List::add_Cell( Jig_Node* cell)
{
    if(head==NULL)
    {
        head=cell;
        return;
      };

    if(head->cell.get_cell_ID()>cell->cell.get_cell_ID())
      {
            Jig_Node* temp;
            temp=head;
            head=cell;
            temp->prev=head;
            head->next=temp;
            return;
      };


      Jig_Node* temp;
      temp=head;

      while(temp->next!=NULL&&temp->cell.get_cell_ID()<
                  cell->cell.get_cell_ID())
            temp=temp->next;

      if(temp->next==NULL && temp->cell.get_cell_ID()<
                  cell->cell.get_cell_ID())
      {
            temp->next=cell;
            cell->prev=temp;
            return;
      }

      else
      {
            cell->next=temp;
            cell->prev=temp->prev;
            temp->prev->next=cell;
            temp->prev=cell;
      };

}

//function for setting a Jig System
void JigSystem::set(Mineral* ph, Mineral* pl, Jig_Node_List* JNlist,
                  unsigned cellno, Jig_Node* pf,double grad,boolean mag)
{
```

```
            Heavy=ph;
            Light=pl;
            Head=JNlist;
            cell_No=cellno;
            pfc=pf;
            feed_grad=grad;
            mag_feed=mag;
}

//function for single test simulation
void JigSystem::simulate( FILE* pp,int part_no)
{
      int ii,signal,cellnum,fcellnum;
      double size_range[7],sum_c_h[8],sum_c_l[8],sum_t_h[8],sum_t_l[8],
             H_dis[8],L_dis[8],ave_grade, dsize,total_rec_h=0,total_rec_l=0,
             size_ratio_h[8],size_ratio_l[8],H_rec[8],H_grad[8],
             x1,x2,x3,x4,x5,x6,avg_density,time,avgsize,tempd1,tempd2,tempd3,
             tempd4,tempd5,lmax,hmax,sum_f_l,T_ratio,part_v;
      Particle * particle=new Particle;
      Mineral* ppm=new Mineral;
      Jig_Node* temp;
      Vcell* cell=new Vcell;
      avgsize=Heavy->get_avg_size()*feed_grad/100.0+
                  Light->get_avg_size()*(1-feed_grad/100.0);
      for(int j=0; j<7;j++)
         size_range[j]=Heavy->get_max_size()/pow(sqrt(2.0),j+1);
      lmax=Light->get_max_size();
      hmax=Heavy->get_max_size();
      tempd1=-0.5*Heavy->get_dis_const();
      tempd2=-0.5*Light->get_dis_const();
      tempd3=pow((hmax/lmax),Light->get_dis_const());
      sum_f_l=0.0;
      for(j=0;j<7;j++)
      {
            size_ratio_h[j]=pow(2,tempd1*j)-pow(2,tempd1*(j+1));
            if(lmax<size_range[j]) size_ratio_l[j]=0.0;
            else
            {
                  size_ratio_l[j]=1.0-tempd3*pow(2,tempd2*(j+1))-sum_f_l;
                  sum_f_l+=size_ratio_l[j];
            };
      };
      size_ratio_h[7]=pow(2,tempd1*7);
      size_ratio_l[7]=1.0-sum_f_l;
      for (j=0; j<=7;j++)
      {
            sum_c_h[j]=0.0;
            sum_c_l[j]=0.0;
            sum_t_h[j]=0.0;
            sum_t_l[j]=0.0;
      };

      x2=0;
      avg_density=Light->get_density()*Heavy->get_density()/
                        (Light->get_density()*(1-feed_grad/100.0)+
                        Heavy->get_density()*feed_grad/100.0);
      for( int i=1; i<=part_no;i++)
```

```
{
      time=0.0;
      x1=pfc->cell.get_length()*(float)random(101)/100.0;
      x3=pfc->cell.get_width()*(float)random(101)/100.0;
      temp=pfc;
      pfc->cell.get_U(x1,x2,x3,time,&x4,&x5,&x6);
      cell=&(pfc->cell);

      if(random(11)>5)
      {
            dsize=Light->get_max_size()*(float)random(1001)/1000.0;
            if (dsize<0.000005) dsize=0.000005;
            ppm=Light;
      }

      else
      {
            dsize=Heavy->get_max_size()*(float)random(1001)/1000.0;
            if (dsize<0.000005) dsize=0.000005;
            ppm=Heavy;
      };

      particle->set(ppm,cell,dsize,x1,x2,x3,x4,x5,x6,time);

      printf("\n%d\n",i);

      do
      {
            //move one step
            particle->Move(avg_density,avgsize,mag_feed);
            //check if beyond the cell
            signal=particle->check_boundary();
            time=particle->get_time();

            if(signal!=0)
            {
                  if (signal==-1)
                  {
                        if(temp->next==NULL) break;
                        else
                        temp=temp->next;
                        particle-> transfer(&(temp->cell), &signal);
                  }

                  else //(signal==1)
                  {
                        if(temp->prev==NULL) break;
                        else
                        temp=temp->prev;
                        particle->transfer(&(temp->cell), &signal);
                  };
            };

      } while (time<= Total_time);//end of do loop

      particle->Show();
      if(signal==0)
```

```
{
      cellnum=(particle->get_cell())->get_cell_ID();
      fcellnum=pfc->cell.get_cell_ID();
      if(cellnum<fcellnum)
      T_ratio=1.0-0.5*cellnum/fcellnum;
      else
      T_ratio=0.5*((cell_No-cellnum+1.0)/(cell_No-fcellnum+1.0));

      ii=0;
      while(dsize<size_range[ii]) ii++;
      part_v=PI*pow(dsize,3)/6.0;
      if(ppm==Heavy)
      {
            sum_t_h[ii]+=T_ratio*part_v*Heavy->get_density();
            sum_c_h[ii]+=(1.0-T_ratio)*
                              part_v*Heavy->get_density();
      }
      else
      {
            sum_t_l[ii]+=T_ratio*part_v*Heavy->get_density();
            sum_c_l[ii]+=(1.0-T_ratio)*part_v*
                              Heavy->get_density();
      };

};

if(signal==-1)
{
      if(ppm==Heavy)
      {
            ii=0;
            while(dsize<size_range[ii]) ii++;
            sum_c_h[ii]+=PI*pow(dsize,3)*
                              Heavy->get_density()/6.0;
      }
      else
      {
            ii=0;
            while(dsize<size_range[ii]) ii++;
            sum_c_l[ii]+=PI*pow(dsize,3)*
                              Light->get_density()/6.0;
      };
};

if(signal==1)
{
      if(ppm==Heavy)
      {
            ii=0;
            while(dsize<size_range[ii]) ii++;
            sum_t_h[ii]+=PI*pow(dsize,3)*
                              Heavy->get_density()/6.0;
      }
      else
      {
            ii=0;
            while(dsize<size_range[ii]) ii++;
```

```
                        sum_t_l[ii]+=PI*pow(dsize,3)*
                                        Light->get_density()/6.0;
                    };
                };
        };//end of for loop

        for(ii=0; ii<8;ii++)
        {
                if(sum_c_h[ii]+sum_t_h[ii]==0)  H_dis[ii]= 0.0;
                else H_dis[ii]=sum_c_h[ii]/(sum_c_h[ii]+sum_t_h[ii]);
                if(sum_c_l[ii]+sum_t_l[ii]==0)  L_dis[ii]= 1.0;
                else L_dis[ii]=sum_c_l[ii]/(sum_c_l[ii]+sum_t_l[ii]);
                H_rec[ii]=size_ratio_h[ii]*H_dis[ii];
                tempd4=size_ratio_h[ii]*feed_grad*H_dis[ii];
                tempd5=size_ratio_l[ii]*(100.0-feed_grad)*L_dis[ii];
                if (tempd4+tempd5==0.0) H_grad[ii]=100.0;
                else H_grad[ii]=tempd4*100.0/(tempd4+tempd5);
                total_rec_h+=size_ratio_h[ii]*H_dis[ii];
                total_rec_l+=size_ratio_l[ii]*L_dis[ii];

                fprintf(pp,"Size%d,%lf,%lf,%lf\n",
                        ii+1,tempd4+tempd5,H_grad[ii],H_dis[ii]*100.0);
        };
        ave_grade=total_rec_h*feed_grad/
                        (total_rec_h*feed_grad+total_rec_l*(100.0-feed_grad));

        fprintf(pp,"Total,%lf,%lf,%lf\n",

        total_rec_h*feed_grad/ave_grade,100.0*ave_grade,100.0*total_rec_h);

}


//function for parameter condition test simulation
void JigSystem::multi_sim(FILE *pp,int part_no)
{
        int ii,signal,cellnum,fcellnum;
        double size_range[7],sum_c_h[8],sum_c_l[8],sum_t_h[8],sum_t_l[8],
               H_dis[8],L_dis[8],ave_grade, dsize,total_rec_h=0,total_rec_l=0,
               size_ratio_h[8],size_ratio_l[8],avgsize,
               x1,x2,x3,x4,x5,x6,avg_density,time,T_ratio,part_v,
               tempd1, tempd2,tempd3,lmax,hmax,sum_f_l;
        Particle * particle=new Particle;
        Mineral* ppm=new Mineral;
        Jig_Node* temp;
        Vcell* cell=new Vcell;
        avgsize=Heavy->get_avg_size()*feed_grad/100.0+
                    Light->get_avg_size()*(1-feed_grad/100.0);
        for(int j=0; j<7;j++)
        size_range[j]=Heavy->get_max_size()/pow(sqrt(2.0),j+1);
        lmax=Light->get_max_size();
        hmax=Heavy->get_max_size();
        tempd1=-0.5*Heavy->get_dis_const();
        tempd2=-0.5*Light->get_dis_const();
        tempd3=pow((hmax/lmax),Light->get_dis_const());
        sum_f_l=0.0;
        for(j=0;j<7;j++)
```

```
{
      size_ratio_h[j]=pow(2,tempd1*j)-pow(2,tempd1*(j+1));
      if(lmax<size_range[j]) size_ratio_l[j]=0.0;
      else
      {
            size_ratio_l[j]=1.0-tempd3*pow(2,tempd2*(j+1))-sum_f_l;
            sum_f_l+=size_ratio_l[j];
      };
};
size_ratio_h[7]=pow(2,tempd1*7);
size_ratio_l[7]=1.0-sum_f_l;
for (j=0; j<=7;j++)
{
      sum_c_h[j]=0.0;
      sum_c_l[j]=0.0;
      sum_t_h[j]=0.0;
      sum_t_l[j]=0.0;
};

x2=0;
avg_density=Light->get_density()*Heavy->get_density()/
                  (Light->get_density()*(1-feed_grad/100.0)+
                  Heavy->get_density()*feed_grad/100.0);

for( int i=1; i<=part_no;i++)
{
      time=0.0;
      x1=pfc->cell.get_length()*(float)random(101)/100.0;
      x3=pfc->cell.get_width()*(float)random(101)/100.0;
      pfc->cell.get_U(x1,x2,x3,time,&x4,&x5,&x6);
      temp=pfc;
      cell=&(pfc->cell);

      if(random(11)>5)
      {
            dsize=Light->get_max_size()*(float)random(1001)/1000.0;
            if (dsize<0.000005) dsize=0.000005;
            ppm=Light;
      }

      else
      {
            dsize=Heavy->get_max_size()*(float)random(1001)/1000.0;
            if (dsize<0.000005) dsize=0.000005;
            ppm=Heavy;
      };

      particle->set(ppm,cell,dsize,x1,x2,x3,x4,x5,x6,time);

      printf("\n%d\n",i);

      do
      {
            //move one step
            particle->Move(avg_density,avgsize,mag_feed);
            //check if beyond the cell
            signal=particle->check_boundary();
            time=particle->get_time();

            if(signal!=0)
            {
                  if (signal==-1)
                  {
                        if(temp->next==NULL) break;
                        else
                        temp=temp->next;
                        particle-> transfer(&(temp->cell), &signal);
                  }

                  else //(signal==1)
                  {
                        if(temp->prev==NULL) break;
                        else
                        temp=temp->prev;
                        particle->transfer(&(temp->cell), &signal);
                  };

            };

      } while (time<= Total_time);

      particle->Show();
      if(signal==0)
      {
            cellnum=(particle->get_cell())->get_cell_ID();
            fcellnum=pfc->cell.get_cell_ID();
            if(cellnum<fcellnum)
            T_ratio=1.0-0.5*cellnum/fcellnum;
            else
            T_ratio=0.5*((cell_No-cellnum+1.0)/(cell_No-fcellnum+1.0));
            ii=0;
            while(dsize<size_range[ii]) ii++;
            part_v=PI*pow(dsize,3)/6.0;
            if(ppm==Heavy)
            {
                  sum_t_h[ii]+=T_ratio*part_v*Heavy->get_density();
                  sum_c_h[ii]+=(1.0-T_ratio)*part_v*
                                    Heavy->get_density();
            }
            else
            {
                  sum_t_l[ii]+=T_ratio*part_v*Heavy->get_density();
                  sum_c_l[ii]+=(1.0-T_ratio)*part_v*
                                    Heavy->get_density();
            };

      };

      if(signal==-1)
      {
            if(ppm==Heavy)
            {
                  ii=0;
                  while(dsize<size_range[ii]) ii++;
```

```
                        sum_c_h[ii]+=PI*pow(dsize,3)*
                                          Heavy->get_density()/6.0;
                  }
                  else
                  {
                        ii=0;
                        while(dsize<size_range[ii]) ii++;
                        sum_c_l[ii]+=PI*pow(dsize,3)*
                                          Light->get_density()/6.0;
                  };
            };

            if(signal==1)
            {
                  if(ppm==Heavy)
                  {
                        ii=0;
                        while(dsize<size_range[ii]) ii++;
                        sum_t_h[ii]+=PI*pow(dsize,3)*
                                          Heavy->get_density()/6.0;
                  }
                  else
                  {
                        ii=0;
                        while(dsize<size_range[ii]) ii++;
                        sum_t_l[ii]+=PI*pow(dsize,3)*
                                          Light->get_density()/6.0;
                  };
            };
      };//end of for loop

      for(ii=0; ii<8;ii++)
      {
            if(sum_c_h[ii]+sum_t_h[ii]==0)  H_dis[ii]= 0.0;
            else H_dis[ii]=sum_c_h[ii]/(sum_c_h[ii]+sum_t_h[ii]);
            if(sum_c_l[ii]+sum_t_l[ii]==0)  L_dis[ii]= 1.0;
            else L_dis[ii]=sum_c_l[ii]/(sum_c_l[ii]+sum_t_l[ii]);
            total_rec_h+=size_ratio_h[ii]*H_dis[ii];
            total_rec_l+=size_ratio_l[ii]*L_dis[ii];
      };
      ave_grade=total_rec_h*feed_grad/
            (total_rec_h*feed_grad+total_rec_l*(100.0-feed_grad));

      fprintf(pp,"%lf\t%lf\n",100.0*total_rec_h,100.0*ave_grade);

}


//function for observation particle movement
void JigSystem::Observer( FILE* pp,Particle* part1, Particle* part2,
                                    int x0, int y0,double TT)
{
      int signal1,signal2,x1,y1,x2,y2,count;
      double avg_density,time1,time2,y_h,y_l,avgsize;
      Particle oldp1,oldp2;
      Jig_Node* temp1, *temp2;
```

```
avgsize=Heavy->get_avg_size()*feed_grad/100.0+
            Light->get_avg_size()*(1-feed_grad/100.0);
time1=0.0;
time2=0.0;
temp1=pfc;
temp2=pfc;
avg_density=Light->get_density()*Heavy->get_density()/
                  (Light->get_density()*(1-feed_grad/100.0)+
                  Heavy->get_density()*feed_grad/100.0);
x1=x0;
y1=y0;
x2=x0;
y2=y0;

for (count=1; count<=(int)(1000*TT); count++)
{
      do
      {
            oldp1=*part1;
            oldp1.erase(x1,y1);
            //move one step
            part1->Move(avg_density,avgsize,mag_feed);
            //check if beyond the cell
            signal1=part1->check_boundary();
            time1=part1->get_time();
            if(signal1!=0)
            {
                  if (signal1==-1)
                  {
                        if(temp1->next==NULL)
                        {
                              getchar();
                              getchar();
                              return;
                        }
                        else
                        x1+=(int)(TIMES*temp1->cell.get_H()/
                                    temp1->cell.get_slope());
                        y1+=(int)(TIMES*temp1->cell.get_H());
                        temp1=temp1->next;
                        part1-> transfer(&(temp1->cell), &signal1);
            };

            if(signal1==1)
            {
                  if(temp1->prev==NULL)
                  {
                        getchar();
                        getchar();
                        return;
                  }
                  else
                  temp1=temp1->prev;
                  x1-=(int)(TIMES*temp1->cell.get_H()/
                              temp1->cell.get_slope());
                  y1-=(int)(TIMES*temp1->cell.get_H());
                  part1->transfer(&(temp1->cell), &signal1);
                  };

            };
            //draw the first particle
            part1->draw(x1,y1,RED);

      } while (time1<= 0.001*count);

      do
      {
            oldp2=*part2;
            oldp2.erase(x2,y2);
            //move one step
            part2->Move(avg_density,avgsize,mag_feed);
            //check if beyond the cell
            signal2=part2->check_boundary();
            time2=part2->get_time();

            if(signal2!=0)
            {
                  if (signal2==-1)
                  {
                        if(temp2->next==NULL)
                        {
                              getchar();
                              getchar();
                              return;
                        }
                        else
                        x2+=(int)(TIMES*temp2->cell.get_H()/
                                    temp2->cell.get_slope());
                        y2+=(int)(TIMES*temp2->cell.get_H());
                        temp2=temp2->next;
                        part2-> transfer(&(temp2->cell), &signal2);
                  };

                  if(signal2==1)
                  {
                        if(temp2->prev==NULL)
                        {
                              getchar();
                              getchar();
                              return;
                        }
                        else
                        temp2=temp2->prev;
                        x2-=(int)(TIMES*temp2->cell.get_H()/
                              temp2->cell.get_slope());
                        y2-=(int)(TIMES*temp2->cell.get_H());
                        part2->transfer(&(temp2->cell), &signal2);
                  };

            };
            //draw the second particle
            part2->draw(x2,y2,GREEN);
```

```
            } while (time2<= 0.001*count);

            y_h=(double)(y1-y0)/TIMES+part1->get_y();
            y_l=(double)(y2-y0)/TIMES+part2->get_y();

            if(fmod(count,10)==0)
            fprintf(pp,"%lf\t%lf\t%lf\n",0.001*count,y_h,y_l);
   };
       getchar();getchar();
}


//function for showing the Jig System
void JigSystem::show()
{
      Jig_Node* temp;
      printf("\nThe Jig System is composed of following:");
      printf("\nHeavy Mineral:");
      Heavy->show();
      printf("\nLight Mineral:");
      Light->show();
      printf("\nThe feed grade of heavy mineral is %8.4f percent",feed_grad);
      if(mag_feed) printf("\nWith feedline magnetizing");
      printf("\nThere are %d cells in the vertical direction. They
      are:",cell_No);
      temp=Head->head;
      temp->cell.show();
      if (temp->next!=NULL)
      do
      {
            temp=temp->next;
            temp->cell.show();
      }while(temp->next!=NULL);
}


//function for drawing the Jig System
void JigSystem::draw(int x0, int y0)
{
            int x1,y1;
            Jig_Node* temp;
            temp=pfc;
            x1=x0;
            y1=y0;
            temp->cell.draw(x1,y1);
            while(temp->next!=NULL)
            {
                  x1+=(int)(TIMES*temp->cell.get_H()/temp->cell.get_slope());
                  y1+=(int)(TIMES*temp->cell.get_H());
                  temp=temp->next;
                  temp->cell.draw(x1,y1);
            };
            temp=pfc;
            x1=x0;
            y1=y0;
            while(temp->prev!=NULL)
            {
```

```
                        temp=temp->prev;
                        x1-=(int)(TIMES*temp->cell.get_H()/temp->cell.get_slope());
                        y1-=(int)(TIMES*temp->cell.get_H());
                        temp->cell.draw(x1,y1);
                };
        }
```

3. Main CPP File

```
/**************************************************************/
/*     The main() file for Packed Column Jig Simulation      */
/*                                                            */
/*             File name: jig_main.cpp                        */
/*             Designed by Qiang Dai                          */
/*             Advisor: Dr. David C. Yang                     */
/*             Date: Jan. 1999                                */
/*                                                            */
/**************************************************************/

#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>         // for graphics functions
#include"jig_m.h"

void main()
{
      char name1[20];
      boolean mag_feed,is_mag,IsF;
      unsigned cellno=0;
      int driver, mode,opt,ID,*ps,n,part_no;
      double TT,time,L,W,H,S,V,F,A,VC,density,max_size,dis_const,grad;

      Jig_Node_List * JNL=new Jig_Node_List;
      Jig_Node* pfc;
      JigSystem YangJig;
      Mineral heavy, light;
      FILE* inp1,*inp2,*inp3,*inp4,*inp5,*inp6;
      inp1=fopen("D:\\ddd\\option.dat","r");
      fscanf(inp1," %d",&opt);
      if (opt!=2)
      {
            printf("Please input the particle number for simulation: \n");
            scanf("%d",&part_no);
      }
      else
      {
            printf("Please input the observation time (s): \n");
            scanf("%lf", &TT);
      };
      random(11);
      random(101);
      random(1001);

      switch(opt)
      {
            case 1: //single simulation test

            inp2=fopen("D:\\ddd\\minl1.in","r");
            inp3=fopen("D:\\ddd\\cell1.in","r");
            inp4=fopen("D:\\ddd\\grad1.in","r");
```

```
inp5=fopen("D:\\ddd\\result1.dat","w");
inp6=fopen("D:\\ddd\\condn1.in","r");
fscanf(inp6,"%d",&mag_feed);
fscanf(inp2,"%s %lf %lf %lf %d",
            name1,&density,&max_size,&dis_const,&is_mag);
heavy.set(1,density,max_size,dis_const,is_mag);
fscanf(inp2,"%s %lf %lf %lf %d",
            name1,&density,&max_size,&dis_const,&is_mag);
light.set(2,density,max_size,dis_const,is_mag);

do
{
      Jig_Node* jNode=new Jig_Node;
      Vcell cell;
      fscanf(inp3,"%d %lf %lf %lf %lf %lf %lf %lf %lf %d",
                        &ID,&L,&W,&H,&S,&V,&F,&A,&VC,&IsF);
      V=V*sqrt(1.0+S*S)/fabs(S);
      cell.set(ID,L,W,H,S,V,F,A,VC,IsF);
      jNode->set(cell);
      JNL->add_Cell(jNode);
      if(IsF==true) pfc=jNode;
      cellno++;
} while(ID>1);

fscanf(inp4,"%lf",&grad);
YangJig.set(&heavy,&light,JNL,cellno,pfc,grad,mag_feed);
YangJig.simulate(inp5,part_no);
JNL->remove_all();

break;

case 2://observe the particle movement

Jig_Node_List* JNL=new Jig_Node_List;
Particle* part1=new Particle,*part2=new Particle;
double x1,x2=0.0,x3,x4,x5,x6,dsize;
driver = DETECT;          // set to graphics mode
initgraph(&driver, &mode, "C:\\turbocpp\\bgi");
inp2=fopen("D:\\ddd\\minl2.in","r");
inp3=fopen("D:\\ddd\\cell2.in","r");
inp4=fopen("D:\\ddd\\grad2.in","r");
inp5=fopen("D:\\ddd\\result2.dat","w");
inp6=fopen("D:\\ddd\\condn2.in","r");
fscanf(inp6,"%d",&mag_feed);
fscanf(inp2," %s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
heavy.set(1,density,max_size,dis_const,is_mag);
fscanf(inp2," %s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
light.set(2,density,max_size,dis_const,is_mag);

do
{
      Jig_Node* jNode=new Jig_Node;
      Vcell cell;
      fscanf(inp3," %d %lf %lf %lf %lf %lf %lf %lf %lf %d",
                        &ID,&L,&W,&H,&S,&V,&F,&A,&VC,&IsF);
      V=V*sqrt(1.0+S*S)/fabs(S);
      cell.set(ID,L,W,H,S,V,F,A,VC,IsF);

      jNode->set(cell);
      JNL->add_Cell(jNode);
      if(IsF==1)
      pfc=jNode;
      cellno++;
} while(ID>1);

fscanf(inp6," %lf",&dsize);

x1=pfc->cell.get_length()*(float)random(101)/100.0;
x3=pfc->cell.get_width()*(float)random(101)/100.0;
pfc->cell.get_U(x1,x2,x3,0.0,&x4,&x5,&x6);
part1->set(&heavy,&(pfc->cell),dsize,x1,x2,x3,x4,x5,x6,0.0);
fscanf(inp6," %lf",&dsize);

x1=pfc->cell.get_length()*(float)random(101)/100.0;
x3=pfc->cell.get_width()*(float)random(101)/100.0;
pfc->cell.get_U(x1,x2,x3,0.0,&x4,&x5,&x6);
part2->set(&light,&(pfc->cell),dsize,x1,x2,x3,x4,x5,x6,0.0);

fscanf(inp4," %lf",&grad);

YangJig.set(&heavy,&light,JNL,cellno,pfc,grad,mag_feed);
YangJig.draw(300,150);
YangJig.Observer(inp5,part1,part2,300,150,TT);
fprintf(inp5,"999 999 999");
closegraph();
JNL->remove_all();
break;

case 3: //simulate the cell dimmensional effect
int subopt,pn3,i3;
double tempd;
fscanf(inp1,"%d",&subopt);

if(subopt==1)
{
      inp2=fopen("D:\\ddd\\minl31.in","r");
      inp4=fopen("D:\\ddd\\grad31.in","r");
      inp5=fopen("D:\\ddd\\condn31.in","r");
      inp6=fopen("D:\\ddd\\result31.dat","w");

      fscanf(inp5,"%d %d",&mag_feed,&pn3);
      fscanf(inp2,"%s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
      heavy.set(1,density,max_size,dis_const,is_mag);
      fscanf(inp2,"%s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
      light.set(2,density,max_size,dis_const,is_mag);
      fscanf(inp4,"%lf",&grad);
      fprintf(inp6,"%d\n",pn3);
```

```
            for(i3=1;i3<=pn3;i3++)
            {
                  inp3=fopen("D:\\ddd\\cell31.in","r");
                  fscanf(inp5,"%lf",&L);
                  fprintf(inp6,"%lf\t",L);
                  do
                  {
                        Jig_Node* jNode=new Jig_Node;
                        Vcell cell;
                        fscanf(inp3,"%d %lf %lf %lf %lf %lf %lf %lf
                               %d",&ID,&W,&S,&H,&V,&F,&A,&VC,&IsF);
                        V=V*sqrt(1.0+S*S)/fabs(S);
                        cell.set(ID,L,W,H,S,V,F,A,VC,IsF);
                        jNode->set(cell);
                        JNL->add_Cell(jNode);
                        if(IsF==true) pfc=jNode;
                        cellno++;
                  } while(ID>1);
                  fclose(inp3);

YangJig.set(&heavy,&light,JNL,cellno,pfc,grad,mag_feed);
                  //YangJig.show();getchar();

                  YangJig.multi_sim(inp6,part_no);
                  cellno=0;
                  JNL->remove_all();

            };
      };

      if(subopt==2)
      {
            inp2=fopen("D:\\ddd\\minl32.in","r");
            inp4=fopen("D:\\ddd\\grad32.in","r");
            inp5=fopen("D:\\ddd\\condn32.in","r");
            inp6=fopen("D:\\ddd\\result32.dat","w");
            fscanf(inp5,"%d %d",&mag_feed,&pn3);
            fscanf(inp2,"%s %lf %lf %lf %d",
                        name1,&density,&max_size,&dis_const,&is_mag);
            heavy.set(1,density,max_size,dis_const,is_mag);
            fscanf(inp2,"%s %lf %lf %lf %d",
                        name1,&density,&max_size,&dis_const,&is_mag);
            light.set(2,density,max_size,dis_const,is_mag);
            fscanf(inp4,"%lf",&grad);
            fprintf(inp6,"%d\n",pn3);

            for(i3=1;i3<=pn3;i3++)
            {
                  inp3=fopen("D:\\ddd\\cell32.in","r");
                  fscanf(inp5,"%lf",&W);

                  fprintf(inp6,"%lf\t",W);
                  do
                  {
                        Jig_Node* jNode=new Jig_Node;
                        Vcell cell;
                        fscanf(inp3,"%d %lf %lf %lf %lf %lf %lf %lf
                               %d",&ID,&L,&S,&H,&V,&F,&A,&VC,&IsF);
                        V=V*sqrt(1.0+S*S)/fabs(S);
                        cell.set(ID,L,W,H,S,V,F,A,VC,IsF);
                        jNode->set(cell);
                        JNL->add_Cell(jNode);
                        if(IsF==true) pfc=jNode;
                        cellno++;
                  } while(ID>1);
                  fclose(inp3);

YangJig.set(&heavy,&light,JNL,cellno,pfc,grad,mag_feed);

                  YangJig.multi_sim(inp6,part_no);
                  cellno=0;
                  JNL->remove_all();

            };

   };

   if(subopt==3)
   {
            inp2=fopen("D:\\ddd\\minl33.in","r");
            inp4=fopen("D:\\ddd\\grad33.in","r");
            inp5=fopen("D:\\ddd\\condn33.in","r");
            inp6=fopen("D:\\ddd\\result33.dat","w");
            fscanf(inp5,"%d %d",&mag_feed,&pn3);
            fscanf(inp2,"%s %lf %lf %lf %d",
                        name1,&density,&max_size,&dis_const,&is_mag);
            heavy.set(1,density,max_size,dis_const,is_mag);
            fscanf(inp2,"%s %lf %lf %lf %d",
                        name1,&density,&max_size,&dis_const,&is_mag);
            light.set(2,density,max_size,dis_const,is_mag);
            fscanf(inp4,"%lf",&grad);
            fprintf(inp6,"%d\n",pn3);
            for(i3=1;i3<=pn3;i3++)
            {
                  inp3=fopen("D:\\ddd\\cell33.in","r");
                  fscanf(inp5,"%lf",&S);
                  fprintf(inp6,"%lf\t",S);
                  do
                  {
                        Jig_Node* jNode=new Jig_Node;
                        Vcell cell;
                        fscanf(inp3,"%d %lf %lf %lf %lf %lf %lf %lf
                         %d",&ID,&L,&W,&H,&V,&F,&A,&VC,&IsF);
                        V=V*sqrt(1.0+S*S)/fabs(S);
                        cell.set(ID,L,W,H,S,V,F,A,VC,IsF);
                        S=S*(-1.0);
                        jNode->set(cell);
                        JNL->add_Cell(jNode);
                        if(IsF==true) pfc=jNode;
                        cellno++;
                  } while(ID>1);

                  fclose(inp3);
```

```
YangJig.set(&heavy,&light,JNL,cellno,pfc,grad,mag_feed);

                  YangJig.multi_sim(inp6,part_no);
                  cellno=0;
                  JNL->remove_all();
            };
      };

   if(subopt==4)
   {
            inp2=fopen("D:\\ddd\\minl34.in","r");
            inp4=fopen("D:\\ddd\\grad34.in","r");
            inp5=fopen("D:\\ddd\\condn34.in","r");
            inp6=fopen("D:\\ddd\\result34.dat","w");
            fscanf(inp5,"%d %d",&mag_feed,&pn3);
            fscanf(inp2,"%s %lf %lf %lf %d",
                        name1,&density,&max_size,&dis_const,&is_mag);
            heavy.set(1,density,max_size,dis_const,is_mag);
            fscanf(inp2,"%s %lf %lf %lf %d",
                        name1,&density,&max_size,&dis_const,&is_mag);
            light.set(2,density,max_size,dis_const,is_mag);
            fscanf(inp4,"%lf",&grad);
            fprintf(inp6,"%d\n",pn3);
            for(i3=1;i3<=pn3;i3++)
            {
                  inp3=fopen("D:\\ddd\\cell34.in","r");
                  fscanf(inp5,"%lf",&L);
                  W=L;
                  fprintf(inp6,"%lf\t",L);
                  do
                  {
                        Jig_Node* jNode=new Jig_Node;
                        Vcell cell;
                        fscanf(inp3,"%d %lf %lf %lf %lf %lf %lf %d",
                                          &ID,&S,&H,&V,&F,&A,&VC,&IsF);
                        V=V*sqrt(1.0+S*S)/fabs(S);
                        cell.set(ID,L,W,H,S,V,F,A,VC,IsF);
                        jNode->set(cell);
                        JNL->add_Cell(jNode);
                        if(IsF==true) pfc=jNode;
                        cellno++;
                  } while(ID>1);

                  fclose(inp3);

YangJig.set(&heavy,&light,JNL,cellno,pfc,grad,mag_feed);

                  YangJig.multi_sim(inp6,part_no);
                  cellno=0;
                  JNL->remove_all();
            };
      }

      else printf("\n The input is invalid!");

      break;
```

```
case 4://simulate the effect of density
int pn4,i4;

inp2=fopen("D:\\ddd\\minl4.in","r");
inp3=fopen("D:\\ddd\\cell4.in","r");
inp4=fopen("D:\\ddd\\grad4.in","r");
inp5=fopen("D:\\ddd\\result4.dat","w");
inp6=fopen("D:\\ddd\\condn4.in","r");
fscanf(inp6,"%d %d",&mag_feed,&pn4);
fscanf(inp4,"%lf",&grad);
do
{
      Jig_Node* jNode=new Jig_Node;
      Vcell cell;
      fscanf(inp3,"%d %lf %lf %lf %lf %lf %lf %lf %lf %d",
                       &ID,&L,&W,&H,&S,&V,&F,&A,&VC,&IsF);
      V=V*sqrt(1.0+S*S)/fabs(S);
      cell.set(ID,L,W,H,S,V,F,A,VC,IsF);
      jNode->set(cell);
      JNL->add_Cell(jNode);
      if(IsF==true) pfc=jNode;
      cellno++;
} while(ID>1);
fprintf(inp5,"%d\n",pn4);

for (i4=1; i4<=pn4;i4++)
{
      fscanf(inp2,"%s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
      heavy.set(1,density,max_size,dis_const,is_mag);
      fprintf(inp5,"%lf\t",density);

      fscanf(inp2,"%s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
      light.set(2,density,max_size,dis_const,is_mag);
      fprintf(inp5,"%lf\t",density);

      YangJig.set(&heavy,&light,JNL,cellno,pfc,grad,mag_feed);

      YangJig.multi_sim(inp5,part_no);
};
JNL->remove_all();
break;

case 5://simulate the effect of pulsating frequency
int pn5,i5;
inp2=fopen("D:\\ddd\\minl5.in","r");

inp4=fopen("D:\\ddd\\grad5.in","r");
inp5=fopen("D:\\ddd\\condn5.in","r");
inp6=fopen("D:\\ddd\\result5.dat","w");
fscanf(inp5,"%d %d",&mag_feed,&pn5);
fscanf(inp4,"%lf",&grad);

fscanf(inp2,"%s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
```

```
heavy.set(1,density,max_size,dis_const,is_mag);
fscanf(inp2,"%s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
light.set(2,density,max_size,dis_const,is_mag);

fprintf(inp6,"%d\n",pn5);
for (i5=1; i5<=pn5;i5++)
{
      inp3=fopen("D:\\ddd\\cell5.in","r");
      fscanf(inp5,"%lf",&F);
      fprintf(inp6,"%lf\t",F);
      do
      {
            Jig_Node* jNode=new Jig_Node;
            Vcell cell;
            fscanf(inp3,"%d %lf %lf %lf %lf %lf %lf %lf %d",
                              &ID,&L,&W,&H,&S,&V,&A,&VC,&IsF);
            V=V*sqrt(1.0+S*S)/fabs(S);
            cell.set(ID,L,W,H,S,V,F,A,VC,IsF);
            jNode->set(cell);
            JNL->add_Cell(jNode);
            if(IsF==true) pfc=jNode;
            cellno++;
      } while(ID>1);
      fclose(inp3);
      YangJig.set(&heavy,&light,JNL,cellno,pfc,grad,mag_feed);

      YangJig.multi_sim(inp6,part_no);
      cellno=0;
      JNL->remove_all();
};
break;

case 6: //simulate the effect of jig height
int pn6,i6;
inp2=fopen("D:\\ddd\\minl6.in","r");

inp4=fopen("D:\\ddd\\grad6.in","r");
inp5=fopen("D:\\ddd\\condn6.in","r");
inp6=fopen("D:\\ddd\\result6.dat","w");
fscanf(inp5,"%d %d",&mag_feed,&pn6);
fscanf(inp4,"%lf",&grad);

fscanf(inp2,"%s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
heavy.set(1,density,max_size,dis_const,is_mag);
fscanf(inp2,"%s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
light.set(2,density,max_size,dis_const,is_mag);

fprintf(inp6,"%d\n",pn6);
for (i6=1; i6<=pn6;i6++)
{
      inp3=fopen("D:\\ddd\\cell6.in","r");
      fscanf(inp5,"%lf",&H);
      fprintf(inp6,"%lf\t",H);
      do
```

```
			{
				Jig_Node* jNode=new Jig_Node;
				Vcell cell;
				fscanf(inp3,"%d %lf %lf %lf %lf %lf %lf %lf %d",
							&ID,&L,&W,&S,&V,&F,&A,&VC,&IsF);
				V=V*sqrt(1.0+S*S)/fabs(S);
				cell.set(ID,L,W,H,S,V,F,A,VC,IsF);
				jNode->set(cell);
				JNL->add_Cell(jNode);
				if(IsF==true) pfc=jNode;
				cellno++;
			} while(ID>1);
			fclose(inp3);
			YangJig.set(&heavy,&light,JNL,cellno,pfc,grad,mag_feed);

			YangJig.multi_sim(inp6,part_no);
			cellno=0;
			JNL->remove_all();
		};
		break;


		case 7://simulate the effect of particle size
		int pn7,i7;

		inp2=fopen("D:\\ddd\\minl7.in","r");
		inp3=fopen("D:\\ddd\\cell7.in","r");
		inp4=fopen("D:\\ddd\\grad7.in","r");
		inp5=fopen("D:\\ddd\\result7.dat","w");
		inp6=fopen("D:\\ddd\\condn7.in","r");
		fscanf(inp6,"%d %d",&mag_feed,&pn7);
		fscanf(inp4,"%lf",&grad);
		do
		{
			Jig_Node* jNode=new Jig_Node;
			Vcell cell;
			fscanf(inp3,"%d %lf %lf %lf %lf %lf %lf %lf %lf %d",
						&ID,&L,&W,&H,&S,&V,&F,&A,&VC,&IsF);
			V=V*sqrt(1.0+S*S)/fabs(S);
			cell.set(ID,L,W,H,S,V,F,A,VC,IsF);
			jNode->set(cell);
			JNL->add_Cell(jNode);
			if(IsF==true) pfc=jNode;
			cellno++;
		} while(ID>1);

		fprintf(inp5,"%d\n",pn7);
		for (i7=1; i7<=pn7;i7++)
		{
			fscanf(inp2,"%s %lf %lf %lf %d",
					name1,&density,&max_size,&dis_const,&is_mag);
			heavy.set(1,density,max_size,dis_const,is_mag);
			fprintf(inp5,"%lf\t",max_size);

			fscanf(inp2,"%s %lf %lf %lf %d",
					name1,&density,&max_size,&dis_const,&is_mag);
			light.set(2,density,max_size,dis_const,is_mag);
```

```
      fprintf(inp5,"%lf\t",max_size);

      YangJig.set(&heavy,&light,JNL,cellno,pfc,grad,mag_feed);
      YangJig.multi_sim(inp5,part_no);
};
JNL->remove_all();

break;

case 8://simulate the effect of stream velocity
int pn8,i8;
double temp;
inp2=fopen("D:\\ddd\\minl8.in","r");

inp4=fopen("D:\\ddd\\grad8.in","r");
inp5=fopen("D:\\ddd\\condn8.in","r");
inp6=fopen("D:\\ddd\\result8.dat","w");
fscanf(inp5,"%d %d",&mag_feed,&pn8);
fscanf(inp4,"%lf",&grad);

fscanf(inp2,"%s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
heavy.set(1,density,max_size,dis_const,is_mag);
fscanf(inp2,"%s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
light.set(2,density,max_size,dis_const,is_mag);

fprintf(inp6,"%d\n",pn8);
for (i8=1; i8<=pn8;i8++)
{
      fscanf(inp5,"%lf",&temp);
      fprintf(inp6,"%lf\t",temp);
      inp3=fopen("D:\\ddd\\cell8.in","r");
      do
      {
            Jig_Node* jNode=new Jig_Node;
            Vcell cell;
            fscanf(inp3,"%d %lf %lf %lf %lf %lf %lf %lf %d",
                                  &ID,&L,&W,&H,&S,&F,&A,&VC,&IsF);
            V=temp*sqrt(1.0+S*S)/fabs(S);
            cell.set(ID,L,W,H,S,V,F,A,VC,IsF);
            jNode->set(cell);
            JNL->add_Cell(jNode);
            if(IsF==true) pfc=jNode;
            cellno++;
      } while(ID>1);
      fclose(inp3);
      YangJig.set(&heavy,&light,JNL,cellno,pfc,grad,mag_feed);
      YangJig.multi_sim(inp6,part_no);
      JNL->remove_all();
      cellno=0;
};
break;

case 9://simulate the effect of stroke amptitude
int pn9,i9;
inp2=fopen("D:\\ddd\\minl9.in","r");
```

```
inp4=fopen("D:\\ddd\\grad9.in","r");
inp5=fopen("D:\\ddd\\condn9.in","r");
inp6=fopen("D:\\ddd\\result9.dat","w");
fscanf(inp5,"%d %d",&mag_feed,&pn9);
fscanf(inp4,"%lf",&grad);

fscanf(inp2,"%s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
heavy.set(1,density,max_size,dis_const,is_mag);
fscanf(inp2,"%s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
light.set(2,density,max_size,dis_const,is_mag);

fprintf(inp6,"%d\n",pn9);
for (i9=1; i9<=pn9;i9++)
{
      inp3=fopen("D:\\ddd\\cell9.in","r");
      fscanf(inp5,"%lf",&A);
      fprintf(inp6,"%lf\t",A);
      do
      {
            Jig_Node* jNode=new Jig_Node;
            Vcell cell;
            fscanf(inp3,"%d %lf %lf %lf %lf %lf %lf %lf %d",
                              &ID,&L,&W,&H,&S,&V,&F,&VC,&IsF);
            V=V*sqrt(1.0+S*S)/fabs(S);
            cell.set(ID,L,W,H,S,V,F,A,VC,IsF);
            jNode->set(cell);
            JNL->add_Cell(jNode);
            if(IsF==true)
            pfc=jNode;

            cellno++;
      } while(ID>1);

      fclose(inp3);
      YangJig.set(&heavy,&light,JNL,cellno,pfc,grad,mag_feed);

      YangJig.multi_sim(inp6,part_no);
      JNL->remove_all();
      cellno=0;
};
break;

case 10://simulate the effect of solid volume concentrate
int pn10,i10;
inp2=fopen("D:\\ddd\\minl10.in","r");

inp4=fopen("D:\\ddd\\grad10.in","r");
inp5=fopen("D:\\ddd\\condn10.in","r");
inp6=fopen("D:\\ddd\\result10.dat","w");
fscanf(inp5,"%d %d",&mag_feed,&pn10);
fscanf(inp4,"%lf",&grad);

fscanf(inp2,"%s %lf %lf %lf %d",
                  name1,&density,&max_size,&dis_const,&is_mag);
            heavy.set(1,density,max_size,dis_const,is_mag);
            fscanf(inp2,"%s %lf %lf %lf %d",
                              name1,&density,&max_size,&dis_const,&is_mag);
            light.set(2,density,max_size,dis_const,is_mag);

            fprintf(inp6,"%d\n",pn10);
            for (i10=1; i10<=pn10;i10++)
            {
                  inp3=fopen("D:\\ddd\\cell10.in","r");
                  fscanf(inp5,"%lf",&VC);
                  fprintf(inp6,"%lf\t",VC);
                  do
                  {
                        Jig_Node* jNode=new Jig_Node;
                        Vcell cell;
                        fscanf(inp3,"%d %lf %lf %lf %lf %lf %lf %lf %d",
                                    &ID,&L,&W,&H,&S,&V,&F,&A,&IsF);
                        V=V*sqrt(1.0+S*S)/fabs(S);
                        cell.set(ID,L,W,H,S,V,F,A,VC,IsF);
                        jNode->set(cell);
                        JNL->add_Cell(jNode);
                        if(IsF==true)
                        pfc=jNode;

                        cellno++;
                  } while(ID>1);

                  fclose(inp3);
                  YangJig.set(&heavy,&light,JNL,cellno,pfc,grad,mag_feed);

                  YangJig.multi_sim(inp6,part_no);
                  JNL->remove_all();
                  cellno=0;
            };
            break;

            default: break;
            }

      fclose(inp1);
      fclose(inp2);
      fclose(inp3);
      fclose(inp4);
      fclose(inp5);
      fclose(inp6);

      return;
}
```

Vita

Born in P.R.China, Qiang (John) Dai has devoted himself on Mineral Processing technology research and development. He received his BS degree in Mineral Engineering, Central-South University of Technology, Changsha, China (1985) and MS degree in Mineral/Metallurgical Engineering, Wuhan University of Technology (WUT), Beijing, China (1988). From 1988 to 1997, he served as a lecturer, engineer and then associate research professor in the Beijing Graduate School of Wuhan University of Technology.

He went to United States in 1997 for further graduate study. Currently, he is a candidate for master of science degree in Mining engineering Department , West Virginia University.