University of Twente

EEMCS / Electrical Engineering
Control Engineering

Fast dynamic-model based weight measurement

Peter Pieper

M.Sc. Thesis

Supervisors
prof.dr.ir. J. van Amerongen
dr.ir. P.C. Breedveld
dr. N.E. Ligterink

September 2005

Report nr. 034CE2005
Control Engineering
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Summary

This Master thesis is done at the Control Lab of the University of Twente. One of the topics at the Control Lab is the design and control of mechatronic systems. Within this field of interest I have chosen my Master thesis.

Many products are made out of different raw materials. These raw materials should be mixed together in the right proportions. To do so, the materials should be dosed into a weighting scale where the weight can vary form a few grams up till several tons.

Problems arise when dynamic effects, caused by falling material, lead to errors during measurements. The material that is situated between the valve and the sandpile, will provide a rest-drop when the valve is closed. To preserve the accuracy slow dosing of the weight to a desired value is a solution. However, a weighting process is always a compromise between speed and accuracy.

This thesis provide modeling and simulation methods to design dynamic models of granular heaping. Methods will be reviewed for their usefulness. These models should provide the insight of the dynamics of granular heaping.

The goal of this project is to come up with a fast and accurate dynamic weighting process.

Contents

Chapter 1

Problem description

1.1 Introduction

Many products are made out of different raw materials. These raw materials should be mixed together in the correct amounts, given certain error margins. To do so, the materials should be dosed into a weighting scale where the weight can vary from a few grams up till several tons. Problems arise when dynamic effects caused by falling material lead to errors during measurements. Figure 1.1 shows a setup where granular material is dropped

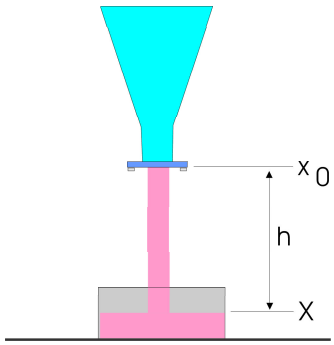

Figure 1.1: Silo setup.

out of a silo onto a weighting scale. At the time when the valve is closed an amount of particles that is situated in the air between x_0 and x will produce a rest-drop. Nowadays, the rest-drop is predicted on the basis of former measurements. The rest-drop depends on the height of the granular pile. However, not every dose demands the same amount of weight. The varying dose will result in different heights of the granular pile. And so, the accuracy of a desired weight cannot be reached due to the varying rest-drop.

The use of a weight load trajectory can improve the accuracy. The weight load trajectory is divided into two stages. In the first stage the valve of the

silo is fully opened. The valve will be closed half way if the desired weight is almost reached. The valve will be completely closed after the desired weight minus the predicted rest-drop is reached. The amount of rest-drop will fill up the weight till it reaches the desired weight. With the use of stages the mass flow will be reduced and so, the amount of rest-drop will be reduced. With the use of a weight load trajectory the accuracy will be improved with respect to the predicted rest-drop. The use of more stages during a load trajectory is time consuming but it can improve the accuracy further more. The given problem indicates that weight measurement is always a compromise between accuracy and speed.

1.2 Goal

Faster dosing requires insight into the dynamic process of a weighting and the rest-drop. Dynamic models have to be made to get these insights. The models should be compared with a real-setup. Finally, dynamic weighting must be implemented to get a faster and more accurate weighting. This demands a real time implementation of a model. Real time finite element models for control engineering demands real time simulations. Therefore, research to appropriate simulation methods must be done.

1.3 Force calculation

In the introduction is mentioned that dynamic effects caused by falling material lead to errors during weighting. To get a better insight in the dynamic forces an example will be given. The dynamic forces consist of an impulse force due to the falling material and a gravitational force produces by the rest-drop.

The example describes an ideal situation of a single state weight load trajectory. As starting point, the setup of figure 1.1 is taken. The granular material leaves the silo at the position x_0 with a velocity at place x_0 of v_0. The material falls over a distance $h = x - x_0$. The ideal situation assumes that the distance h and the velocity v_0 do not change during a weight load trajectory. In contrast with a practical setup where the distance h decreases due to the pile grow. Even so, the velocity v_0 is not constant in a practical setup due to granular settling. The common definition of granular settling is that granular material has to rearrange, settle before a static situation exists. The phenomena of granular settling can be caused by three effects. The first granular settling effect is caused by the silo where granular material has to rearrange before a static flow exists. The second cause is the dispersion effect. This effect exists in the scale container where granular material disperse before a static pile grow arise. The third cause is the avalanche effect. The avalanche of granular material exists at a grown

pile where the slope of the pile becomes to steep. By addition of granular material at this instable pile causes granular material to tumble down the slope. This tumbling effect is called an avalanche.

With the assumption of an ideal situation the forces, produced by the dynamic and static weight measured at the bottom of the scale container, can be calculated. Let g denote as the gravitation force and t the time. Starting with the law of motion these forces can be calculated, which is defined as:

$$x_{(t)} = \frac{1}{2}gt^2 + v_0 t + x_0, \tag{1.1}$$

and Newton's second law:

$$F = \frac{d(mv)}{dt}. \tag{1.2}$$

Newton's second law, equation 1.2, describes that a force F exists if the velocity v times the mass m changes in time. The velocity v depends on time, which means that the deceleration of the granular material should be determined if it hits the granular pile. However, the impulse force caused by fallen grain particles can also be determined with the mass flow $\dot{m}$ and the velocity v_h. The velocity v_h is the velocity of the granular material after it has fallen over a distance h. Therefore, the mass flow $\dot{m}$ and the granular velocity v_h should be determined first. This can be done by calculating the time t such that it depends on the height h.

The time t that it takes for the granular material to fall from x_0 to x can be calculated by rearranging equation 1.1:

$$t_{1,2} = \frac{-v_0 \pm \sqrt{v_0^2 - 2g(x_0 - x)}}{g}. \tag{1.3}$$

From figure 1.1it can be observed that $h = x - x_0$. Taken into account that time must be positive, equation (1.3) yields one physical solution:

$$t_{(h)} = \frac{\sqrt{v_0^2 + 2gh} - v_0}{g}. \tag{1.4}$$

The time imply, in an ideal situation, that all grains have a falling time t during the whole weight-load-cycle. However, this equation does not hold in a practical setup due to the decreasing height h, caused by pile grow.

To determine velocity $v_{(t)}$ of the granular material at place x the time derivative of equation (1.1) is taken:

$$v_{(t)} = gt + v_0. \tag{1.5}$$

Here, the velocity dependents on time. For calculating the impulse force at the bottom of the scale container, the velocity at that place is needed. Therefore, the velocity is calculated as function of the height h. This is done by substitution of equation (1.4) into equation (1.5). The substitution gives

the velocity of the material after it has fallen over the distance h, and is given by:

$$v_{(h)} = \frac{\sqrt{v_0^2 + 2gh} - v_0}{g} g + v_0 = \sqrt{v_0^2 + 2gh}. \tag{1.6}$$

Now the mass flow rate can be calculated. If the granular material initially passes through an area A at velocity v_0, a volume of mass can be defined that leaves the silo in some amount of time t. The volume V is given by:

$$V = Av_0t.$$

The mass m contained in this volume is the density ρ times the volume:

$$m = Av_0t\rho. \tag{1.7}$$

To determine the mass flow rate $\dot{m}$ the time derivative of the mass is taken,

$$\dot{m} = v_0A\rho \tag{1.8}$$

As mentioned these calculations are based on an ideal situation. However, this equation does not hold in a practical situation due to the fact that it can not be assumed that the velocity v_0 is constant during the entire weight-load-cycle. Even so, equation 1.8 only holds at place x_0.

During a weight-load-cycle the integral over the time of the mass flow rate will produce the gravitational static force. This gravitational force is the force of interest. However the measured force during a weight-load-cycle is a sum of the gravitational static force and a impulse fore. This impulse force is produced by the momentum change of mass flow rate at place x. The momentum change is caused by the granular stop when the granular mass flow hits the scale with a velocity v_h. The impulse force as function of the mass flow rate and velocity v_h can be obtained by rearranging equation 1.2:

$$F_{imp} = \frac{dp}{dt} = \frac{dm_{(t)}v_{(h)}}{dt} = v_{(h)}\dot{m}. \tag{1.9}$$

Equation 1.9 depends on the mass flow rate, equation 1.8. However, it is mentioned that this equation only holds at place x_0.

Even if it seems that the mass flow rate is independent of the height, it is mentioned that equation 1.8 only holds at place x_0. Even so, it is imaginable that the mass that flows out of the silo with a certain rate also hits the pile with the same rate. However, the next example will illustrate that this is not the case. Let the mass flow rate out of the silo be 1 kg/sec and that the valve is open for 10 seconds, and so a total amount of 10 kg will fall. Let the fist grains hit the bottom of the scale container 2 seconds after the valve is opened. During the 10 seconds that the valve is opened a granular pile will grow. To illustrate the problem let assume that the height decrease to $\frac{1}{2}h$ due to the pile grow. This means that the last grains that are falling

hit the pile after 1 second instead of the 2 seconds that it takes in the fist place. This means that the granular material creates an impulse force only during 9 seconds instead of the 10 seconds when the valve is opened. And so the effective mass flow rate $\frac{10}{9}$ kg. The example has illustrated that the effective mass flow rate depend on the height h. However, it is assumed that in an ideal situation the height is constant and therefore the impulse force can be given by substituting equation (1.8) and (1.4) into equation (1.9):

$$F_{imp} = v_0 A\rho\sqrt{v_0^2 + 2gh}. \tag{1.10}$$

The impulse force depends on tree parameters A, ρ and g and the two values v_0 and h, which assumed to be constant in the ideal situation. In a practical setup the fist grains that hit the scale container has fallen over the distance h with an initial velocity v_0, which imply that this equation holds. However, depending on the dimensions of the container, the grains will not instantaneous be stopped due to the dispersion of the grains at the bottom of the container. This phenomenon will provide a relaxation factor, which implies that the equation does not hold in a practical situation.

Now the rest-drop force of an ideal situation can be calculated. The rest-drop depends on the total mass Δm that is situated between x_0 and x times the gravitational constant g. The total mass is given by equation 1.7 where the time t is defined as $t_{(h)}$, with substitution of equation 1.4 we obtain:

$$\Delta m = Av_0\rho t_{(h)} = Av_0\rho\{\frac{\sqrt{v_0^2 + 2gh} - v_0}{g}\}. \tag{1.11}$$

The rest-drop force F_{dm} produced by Δm after the valve is closed is given by:

$$F_{dm} = \Delta mg = v_0 A\rho\{\sqrt{v_0^2 + 2gh} - v_0\} \Rightarrow$$

$$F_{dm} = v_0 A\rho\sqrt{v_0^2 + 2gh} - v_0^2 A\rho. \tag{1.12}$$

The deciding factor for fast dynamic weighting is the time when the valve should be closed. After this time t only the rest-drop give a contribution to the desired static weight. The difference between the measured weight at time t and the final static weight is the difference between the contribution of the rest-drop and the impulse force, given by:

$$\Delta F = F_{dm} - F_{imp}. \tag{1.13}$$

Substituting equation 1.12 and equation 1.10 into equation 1.13 results in:

$$\Delta F = +v_0^2 A\rho. \tag{1.14}$$

This example has shown that the measured force at time t when the valve is close and the final weight after all grains has settled is non zero.

As mentioned, the example is an ideal situation. However, it is illustrated that real setups are far from ideal. In figure 1.2 all situations are sketched, where the dashed line represents the ideal situation and the solid line the non ideal situation. It starts with a impulse force, figure 1.2A, followed by the mass flow rate figure 1.2B. The non ideal situation shows slopes. These slopes are caused by the settling behavior of the granular material. Even so, there is no instantaneous stop of the mass flow rate what implies that there is no instantaneous decrease of the impulse force. The ideal situation assumed that the height h is constant. This assumption does not hold in a practical situation due to the growth of the granular pile. The decrease of the height causes a decrease of the impulse force, figure 1.2A. The scale in a practical setup provides also dynamics, figure 1.2C. These dynamics are caused by the momentum changes and will finally be damped out.

The total force, figure 1.2D, shows a linear increase of the fore. The linear increase is caused by the integral over time of the mass flow rate, which is explained with equation 1.8. And so, the maximum total force, is the sum of the impulse force, plus the integral over the mass flow rate, plus the scale dynamics.

For fast weighting an amount of granular material all the aspects of a non ideal situation should be taken into account. Problems arise when there are unexpected dynamics. To reveal all these unexpected dynamics a real setup is made. Experiments with the real setup should provide insight into unaccounted dynamics and validate the ideal model.

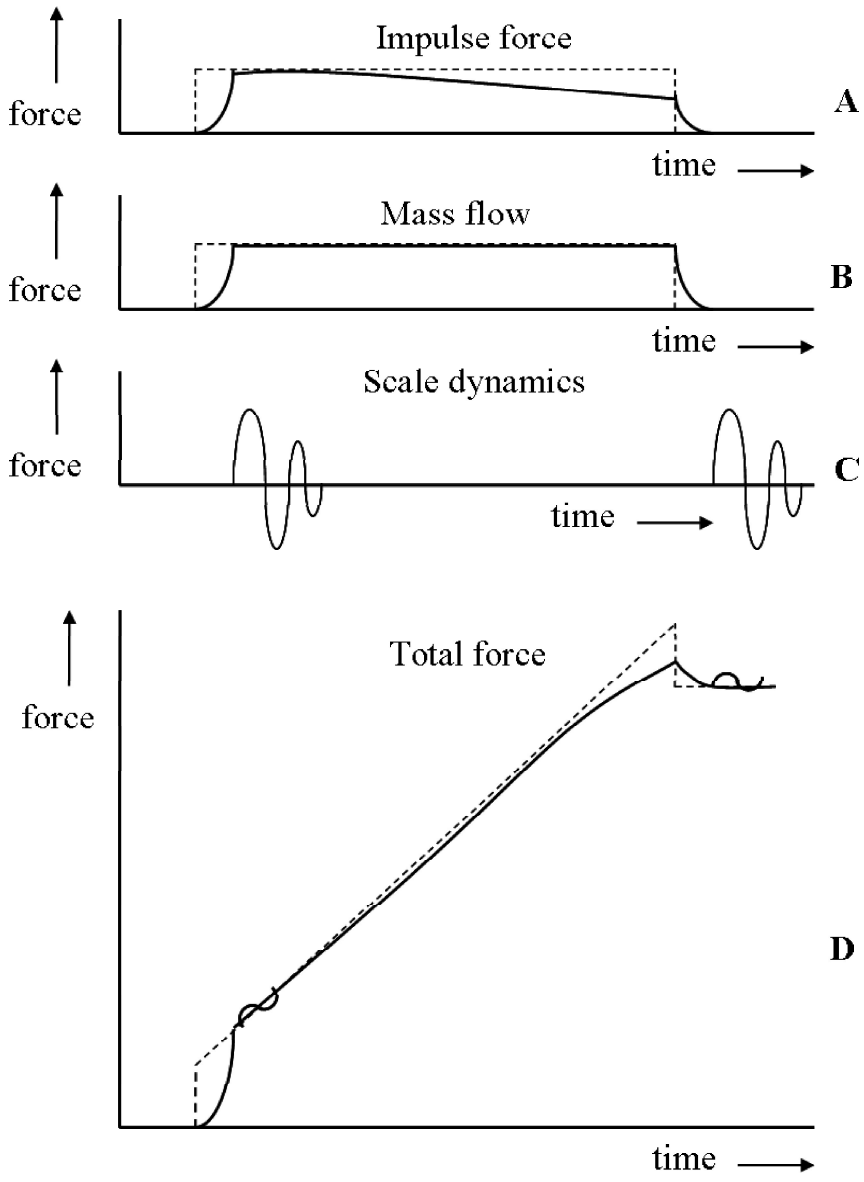

Figure 1.2: Ideal versus reality.

Chapter 2

Preliminary experiments

2.1 Expectations

In section 1.3 the impulse force, rest-drop force, mass flow and the time as function of the height are calculated. These calculations are based on an ideal situation. However, it is expected that a practical situation does not behave like an ideal situation. The assumptions, v_0 is zero and constant height can not be made. The next example will illustrate a weight-load-trajectory at a non ideal situation.

Initially the scale is empty what implies that there is no force acting on the scale. At $t = t_1$ the valve opens and a death-time set in. The death-time is defined as the time that it takes for the granular material to fall from x_0 to x. This time is given in equation 1.4 where the distance x to x_0 is presented as the distance h. During the death-time zero force is acting on the scale. At $t = t_1 + t_{(h)}$ granular material hits the scale producing an impulse force F_{imp}, equation 1.10. After $t = t_1 + t_{(h)}$ the force on the scale will increase due to the mass flow. The increasing force is proportional to the integral over time of the mass flow rate $\dot{m}$ times the gravitational constant g. During the filling process the granular pile will grow and so the height h will decrease, this in contrast to the ideal situation where the height h is kept constant. The decreasing height h will result in a decreasing impulse force. Therefore the height h after $t = t_1 + t_{(h)}$ will be denoted as $\tilde{h}$. When the valve is closed and all grains has settled onto the scale at $t = t_2 + t_{(\tilde{h})}$, rest-drop will produce a total force according equation (1.12).

The impulse force fades out after $t > t_2 + t_{(\tilde{h})}$ due to the fact that there are no grains falling onto the scale. Table (2.1) gives a overview of the force during a weight-load-cycle.

$F_{tot} = 0$	for $t < (t_1 + t_{(h)})$
$F_{tot} = v_0 A\rho\sqrt{v_0^2 + 2gh}$	at $t = t_1 + t_{(h)}$
$F_{tot} = v_0 A\rho\sqrt{v_0^2 + 2g\tilde{h}} + \int_{t_1+t_{(h)}}^{t} g\dot{m}\,\mathrm{d}t$	for $t_1 + t_{(h)} < t \leq t_2$
$F_{tot} = v_0 A\rho\sqrt{v_0^2 + 2g\tilde{h}} + \int_{t_1+t_{(h)}}^{t_2} g\dot{m}\,\mathrm{d}t + v_0 A\rho\{\sqrt{v_0^2 + 2g\tilde{h}} - v_0\}$	at $t = (t_2 + t_{(\tilde{h})})$
$F_{tot} = \int_{t_1+t_{(h)}}^{t_2} g\dot{m}\,\mathrm{d}t + v_0 A\rho\{\sqrt{v_0^2 + 2g\tilde{h}} - v_0\}$	for $t > (t_2 + t_{(\tilde{h})})$

Table 2.1: Total force during a load trajectory.

The total force in table 2.1 is based on a non ideal situation where the decreasing height h is taken into account. The force during a weight-load-trajectory is sketched in figure 2.1. The valve openings time is represented with the dashed line. The impulse force will decrease due to the decreasing height h. This is represented by the curved line in figure 2.1.

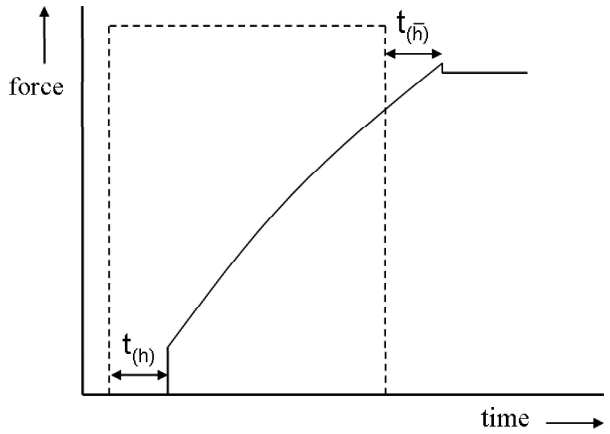

Figure 2.1: Predicted output.

It also been expected that the velocity v_0 is not constant. This will lead to unexpected dynamic effects during weighting. This is not taken into account in the previous example. To get a better insight into this problem a preliminary experiment has been carried out. This experiment is even been used to review the expectations and to research if no other unexpected dynamics occur during a weight-load-cycle.

2.2 Real setup

To validate the expected weight load trajectory a real setup is made (see figure 2.2 a). The silo can contain approximately 18 kg of dry sand. The

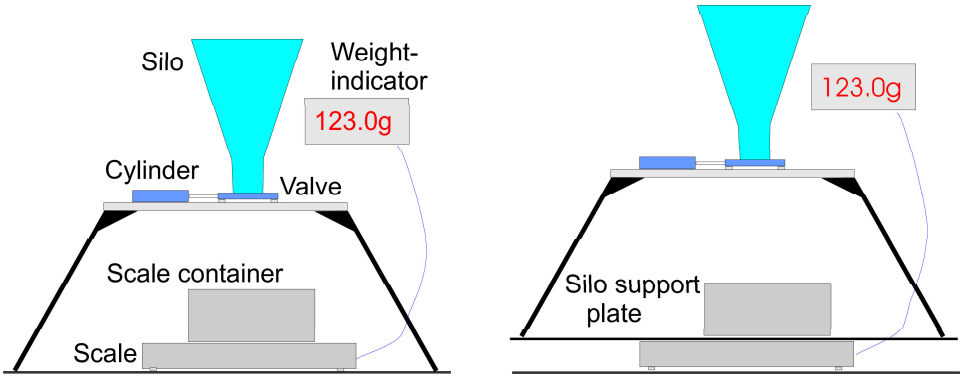

Figure 2.2: a Real setup. b Hourglass setup.

boring hole of the valve can vary up till 60 mm. The valve consists of a pneumatic cylinder, where the velocity can be varied through adjustable air filters.
The scale is based on loadcell principle and it has a maximum load of 50 kg. The weight indicator determines the force on the loadcell by measuring the voltage difference over the loadcell. The software of the weight indicator is adapted to store data during a weight-load-cycle. The data consist of jiffies[1], raw sample values, filtered sample values and a marker valve open valve closed. One weight-load-cycle has a fixed time period of 10 seconds and can be started through a *start command*. The valve can be opened and closed manually or through additional settings of the *start command*. An example of the *start command* is *starttestdata 50 600*, which means that the valve opens after 50 jiffies times 10 ms is 0.5 seconds and closes after 6 seconds. The data can be read with a *read command*, *readtestdata*.
The weight indicator is calibrated in a range of 0 kg till 50 kg in steps of 1 gr (50.000 scale parts).

The real setup is also used in a different configuration where the setup acts like a hourglass. In the hourglass setup the silo with the scale container is placed on top of the scale trough an additional plate (point one in figure 2.2 b), such that only the dynamic effects are measured.

2.3 Experiment findings

In section (2.2) a real setup is described. With this setup a series of weight-load-cycles are performed.

The fist test is done with the settings given in table (2.2). The valve

[1] one jiffie corresponds to a period of 10 ms.

height h	1200 mm
boring hole	60 mm
valve speed	fast
opening time	0.5 seconds
closing time	6.0 seconds

Table 2.2: Settings test 1.

speed is given in terms of *fast* or *slow*. The term *fast* refers to completely opened air filters and the term *slow* to half opened air filters. Unfortunately, the reaction time and speed could not be measured in terms of meters per seconds. The output of the data is stored into a file and plotted in figure (2.3).

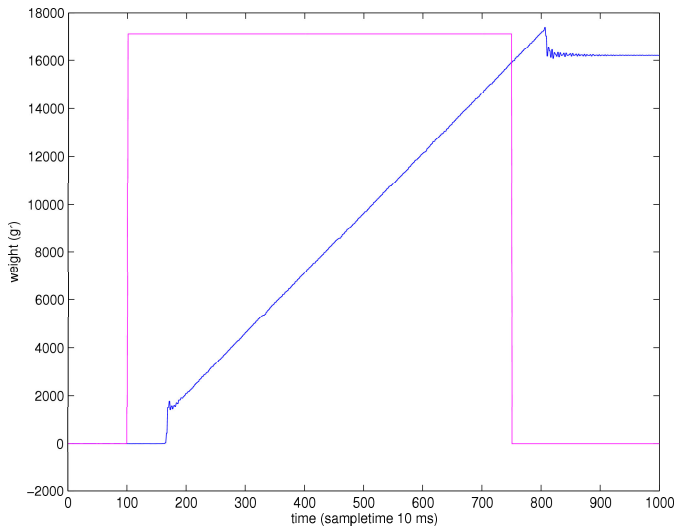

Figure 2.3: Real setup output.

2.3.1 Expectations versus measurements

The measured output of figure 2.3 shows similarities and differences compared with the expected output of figure 2.1.

Unfortunately, the test shows that the expected decreasing height h that should result in a deceasing F_{imp} is not significant, such that it is recognizable in the plot of figure 2.3. However, this only implies that the config-

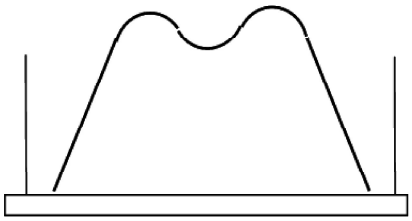

Figure 2.4: Top of a pile.

uration of the setup does not provide answers to the problem and it does not imply that the problem does not exist. It is plausible that a decreasing height can be observed in the plot if the difference in height can be decreased more. Vise versa, it is observed that the impulse force causes the effect that the pile is smoothen with a dent in the middle. And so, the total height h will be reduced due to the impulse force. Because of the dented and smoothen top, the shape of the pile is not cone shaped. Figure 2.4 shows a sketch of a top of a pile that is observed during the experiments. However, the decreasing height h is of great importance for the calculation of the rest-drop, equation 1.12. Therefore, the pile grow should become more clear.

The times $t_{(h)}$ and $t_{(\tilde{h})}$ should be different due to the decreasing height h. However, the decreasing height is not observed in contrast to the difference in time, which means that the difference in time should have another cause. The most plausible cause should be a non linear valve.

The two momentum changes, figure 2.3, shows a slope. This phenomenon can be a result of several reasons:

- The first grains disperse at the bottom of the scale container, causing damping.
- The velocity of the valve is not fast enough such that this can be neglected.
- v_0 is not constant, due to the fact that grains has to settle before a static flow exists.

The dispersion of grains is verified by observations during the preliminary tests. Even so, the valve is not infinite fast. Some remarks should be made with respect to v_0. The plot of figure 2.3 shows a linear increase of the force. This is caused by the integral of the mass flow rate, equation 1.8. The mass flow rate dependents on v_0, which would imply that with a linear increase of the force, the value of v_0 is constant. Even so, it is observed that the momentum changes are followed by an unexpected damped oscillating

behavior. This effect is a result of the dynamic characteristics of the weight scale platform.

The measurements are based on one single setup configuration. However, the real setup can be varied with varying dynamics as result. So will, by increasing the distance between the silo and the scale container result into an increasing impulse force. Even so, the dimensions of the scale container can be varied. Smaller container area will result into a higher granular pile and a decreasing impulse force. A larger container area will result into larger grain dispersion and so, a longer settling time.

The ideal setup provided a global answer to the dynamic problems during a weight-load-cycle. However, the expectations and preliminary experiment has shown exceptions to the model. For fast weighting, all the dynamics should be fully understood. Modeling the behavior of these dynamics should provide answers to the problems caused by these dynamics. Modeling has its advantage that it can easily provide insights if parameters or dimensions changes.

Chapter 3

Modeling

3.1 introduction

In Chapter 1 all forces that are acting on a scale during a weight load trajectory are explained. However, the given example is an ideal situation. The decreasing height h, changing velocity v_0 and the setting-time of a unit granular particles are not included in the ideal model example. For accurate control of a certain amount of granular material these changes should be taken into account. Modeling the dynamics of a granular pile should provide more insight in the pile grow and behavior of granular material.

Settling time of granular material is one of the unrevealed effects during a weight-load-cycle. The avalanches that exists during a granular pile grow is the main factor of the settling behavior of grains. The first model approach is a self-organized-critical or shortly SOC and it provide insight into the avalanches during weighting.

For getting a better insight into the changing velocity v_0 a fluid dynamic model approach is used. This model approach models the silo of the real setup.

The Discrete Element Method (DEM) is used to implement a unified model. In this model all dynamic aspects should come together.

These model approaches are implemented into a c-code with OpenGL and X11 windows as simulation output. The use of OpenGL and X11 is explained in Appendix A.

3.2 Grain particle update model

A first approach is a static model where simple update rules moves grain particles. The grain particles can only move to the left and to the right. Grain particles can also tumble down a hill if the hill becomes to steep. This model approach can be compared with self-organized criticality or shortly SOC. SOC is first introduced by Bak, Tang and Wiesenfeld (BTW) [5].

BTW have introduced a simple model that contains some important features of a dynamic granular pile. It describes a stochastic natural evolution process, like sandpiles, that starts at an arbitrary initial state. The sand pile organizes itself if the slope of a pile reaches a critical state (SOC). If a

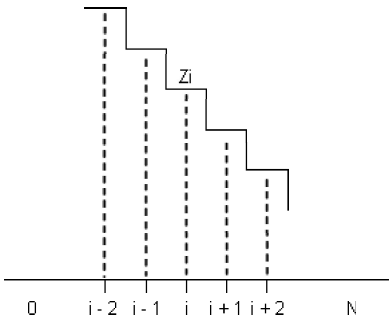

Figure 3.1: State of a sandpile slope.

grain is added to a stable pile at the ith position, see figure 3.1, than the slope z of the pile updates as follows:

$$\begin{aligned} z_{i-1} &\rightarrow z_{n-1} - 1, \\ z_i &\rightarrow z_n + 1. \end{aligned} \tag{3.1}$$

When the height difference becomes higher than a fixed critical value z_c, one unit of granular material tumbles to the lower lever and the slopes updates as follows:

$$\begin{aligned} z_{i-1} &\rightarrow z_{n-1} + 1, \\ z_i &\rightarrow z_n - 2, \\ z_{i+1} &\rightarrow z_{n+1} + 1. \end{aligned} \tag{3.2}$$

These update rules are implemented into a simulation program. Figure 3.2A shows a sub-flowchart of the SOC model approach. The sub-flowchart revers to the sub-block *handling* that is part of the basic flowchart. This basic flowchart is given in appendix A.2. Figure 3.2B give a screen-shot of a 1D SOC model. A 3D implementation of a SOC model is given in figure 3.3A/B. These plots have shown the evolution of a granular pile as function of the angle of the slope. By using the slope, the evolution of an avalanche becomes visible. During tests with the real setup, these avalanches have been observed. However, the avalanches where not uniform distributed over the slope, this in contrast to the model. The contrast can be explained by the addition of grains in the model, which is precise at the center of the pile. The addition of grains in the center will result into a uniform distributed avalanche.

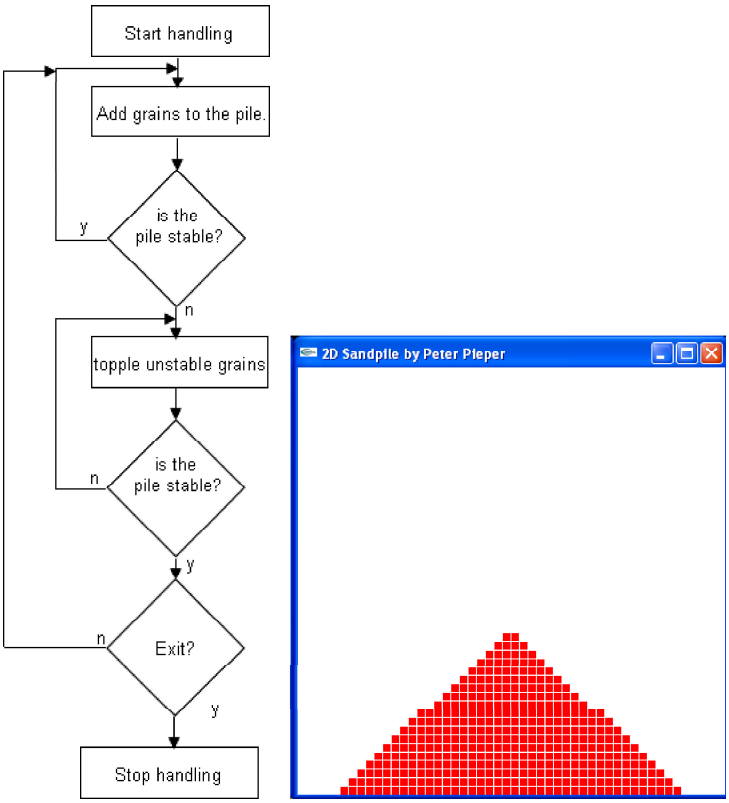

Figure 3.2: A Flowchart of a SOC model. B Implementation.

Nevertheless, it is observed and modelled that the granular pile starts in a stable state. The addition of granular material is concentrated in the middle of the top of the pile. An avalanche exists if the top of the pile becomes unstable. After an avalanche the pile becomes in a stable state again. This procedure will be repeated until the mass flow stops. The conditions, shape, stability etcetera, at the top of a pile are the same after every avalanche. If the conditions at the top are repeated after every avalanche then it can be concluded that the avalanches exists in a repeating pattern. This repeating pattern is observed during the experiments and during simulation of the model. The repeating pattern will imply that a same amount of mass at the top of the pile will cause an avalanche.

The differences during a pile grow is the duration of an avalanche. So will a larger pile result in a longer duration of an avalanche. This effect is caused by the longer slopes of a pile where the grains are rolling over. The avalanche that exists causes a momentum changes. The momentum changes imply that there is a force and so, that there is a force acting on the scale. The contribution of these forces is not observable in the measured data.

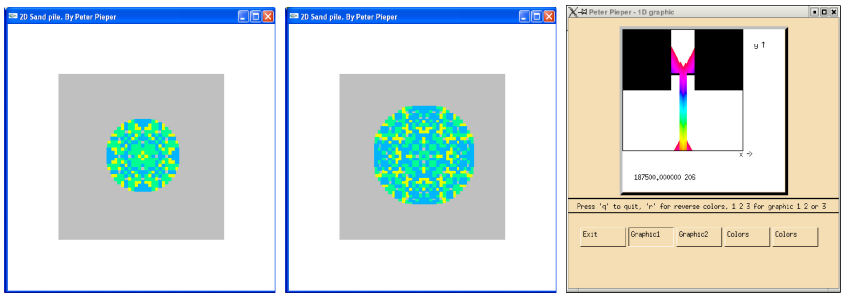

Figure 3.3: Evolution of a 2D SOC model. A after 750 iterations. B after 1500 iterations. C 2D granular heap model.

A disadvantage of a SOC model is the addition of a sand grain, which is simply stacked at an arbitrary place. The behavior of "fallen" grains is not modelled in this matter. Therefore, the SOC model is adapted with extra update criteria. The first criterion of a fallen particle is that it is free to move (FTM) and so, that it is not boxed between other grain particles. An empty space that exists due to an fallen grain particle should be filled before next grain particles can fall. Figure 3.3C shows a screen-shot of a 2D granular heap program, where one pixel represents one unit of grain. The 2D SOC model provides also height information, figure 3.2B and figure 3.3C. However, the preliminary experiments have shown that the top of a granular pile is smoothen and dented due to the impulse of the granular material, this in contrast to the SOC model approach. Expanding the update rules should model these dynamic effects. In the expanded model the maximum pressure on a grain is the decisive criteria. However, this leads to an inappropriate model behavior. After every iteration step the model comes in a static state where the pressure is calculated. In this state the pressure at the bottom is always higher than the pressure at the top of the pile. And so, the grains at the bottom sides moves first to the left or right.

3.3 Fluid model

A fluid model approach is chosen to determine the mass-flow of unit grain particles out of a silo. If some assumptions are made with respect to the fluid flow model, the model can be compared with a granular flow model. These assumptions assumes incompressible ($\nabla \cdot u = 0$) laminar flow (no turbulence or vorticity, $\nabla \times u = 0$), assumes that the distance from the larger diameter of the silo to the smaller is short enough that viscous losses can be neglected, and assumes that the velocity profile follows that of theoretical laminar flow. An equation describing a fluid-flow velocity u that meet these assumptions,

is given by Poisson's law:

$$\nabla^2 u = f(x, y), \tag{3.3}$$

where ∇^2 is the Laplacian defined as:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \tag{3.4}$$

and where f is a function of pressure. This means that the velocity u depends on the pressure distribution inside the silo. The first step to model equation 3.3 is to replace the continuous domain by a finite difference mesh or grid. As an example, a solution should be found for $\nabla^2 u$, where u is defined

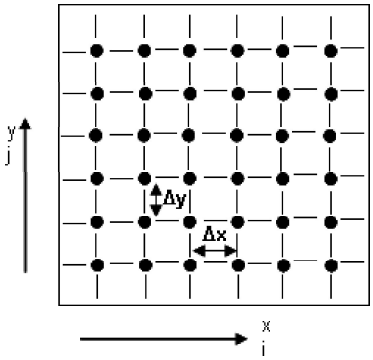

Figure 3.4: Finite element mesh or grid.

as $u_{(x,y)}$, than $u_{(x,y)}$ can be replaced by $u_{(i\Delta x, j\Delta y)}$, where the parameters i, Δx, j and Δy are illustrated in figure 3.4. With the use of the central difference equation, equation 3.4 can be rewritten as:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta y^2} + TE. \tag{3.5}$$

The truncation error (T.E.) is the difference between the partial derivative an its finite-element representation. However, when $\Delta x \to 0$ the truncation error can be neglected. Let the function $f(x, y)$ of equation 3.3 be denoted as $R_{i,j}$, so the approximation to Poisson's equation is:

$$\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta y^2} = R_{i,j}. \tag{3.6}$$

As the fluid slips along a wall, the flow velocity of the fluid nearby that wall is zero, meaning that boundary conditions must be set. The Dirichlet conditions satisfy these conditions, given by:

$$u|_{boundary} = 0, \tag{3.7}$$

meaning that $u_{0,j} = 0$, $u_{N,j} = 0$, $u_{i,0} = 0$ and $u_{i,N} = 0$.

The flow velocity U at place i, j can now be determined by rearranging equation 3.6:

$$u_{i,j} = \left(\frac{u_{i-1,j} + u_{i+1,j}}{\Delta x^2} + \frac{u_{i,j-1} + u_{i,j+1}}{\Delta y^2} - R_{i,j} \right) C^{-1}, \tag{3.8}$$

where the constant factor C is denoted as:

$$C = \frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}. \tag{3.9}$$

Example 3.2 gives a pseudo c-code of the Poisson equation with Dirichlet boundary conditions.

Example 3.1

```
for (int i=1; i < MAXIMUM_ITERATION_STEPS; i++) // limitation of iteration steps.
{
    change = 0;
    /*  for all grid or mesh points
        except for x = 0, x = N, y = 0 and y = N */
    for (x=1;x < nx-1; x++)
    {
        for (y=1; y < ny-1; y++)
        {
            Uold[x][y] =  U[x][y];
            U[x][y] = ((U[x-1][y] + U[x+1][y]) / dx2 +
                (U[x][y-1] + U[x][y+1]) / dy2 - R) / factor;

                change = change + fabs(U[x][y] - Uold[x][y]);
        }
    }
    if(change <TOLERANCE) // is the tolerance reached?
    {
        printf("TOLERANCE reached at %d\n",i);
        break;
    }
}
```

Figure 3.5 gives a plot of the simulated code, which parameters are given by table 3.1. The fluid flow model gives an accurate prediction of the fluid

Number of $x_N = 50$
Step size $\Delta x = 0.2$
Number of $y_N = 50$
Step size $\Delta y = 0.2$
Tolerance $= 1.e - 28$

Table 3.1: Simulation parameters.

velocity v_0. Unfortunately this model depends on the pressure distribution function $f(x, y)$ which is difficult to find. Without this function the outflow

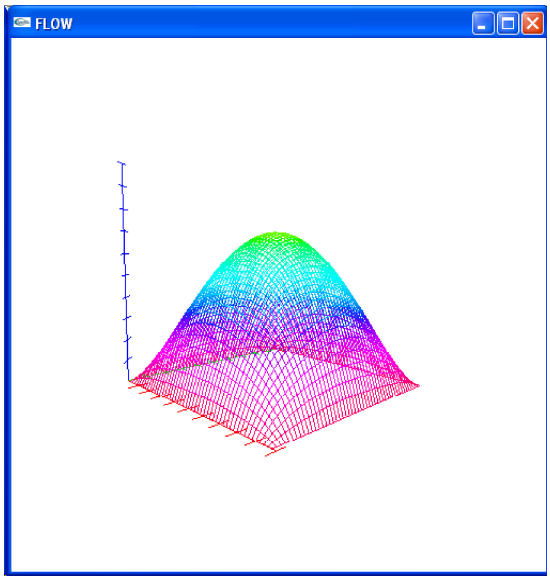

Figure 3.5: Flow velocity trough a duct.

of the granular material can not be determined. Even if this can be done, the fluid model provides only insight in the velocity profile.

3.4 DEM model

The discrete element method (DEM) for simulating granular material is introduced by Cundall and Strack [7]. The DEM is used to track the position, velocity, orientation and other parameters of particles during the simulation. In the DEM model the granular material is considered as a spherical particle referred to a discrete element. The motion of the particles is calculated from the applied forces and moments defined by Newton's second law (see equation 1.2). To explain the basics of DEM, a two-particle contact model is described in a 2-dimensional space, where the grains assumed to have no rotation. Two discs x and y are defined to be in contact if the distance D between the centers is less than the sum of the individual radius:

$$D < R_x + R_y. \tag{3.10}$$

The velocity of interest is only that along the line of impulse. The relative velocity of point x with respect to y, defined as $\dot{X}$ is given by:

$$\dot{X} = \dot{x}_i - \dot{y}_i. \tag{3.11}$$

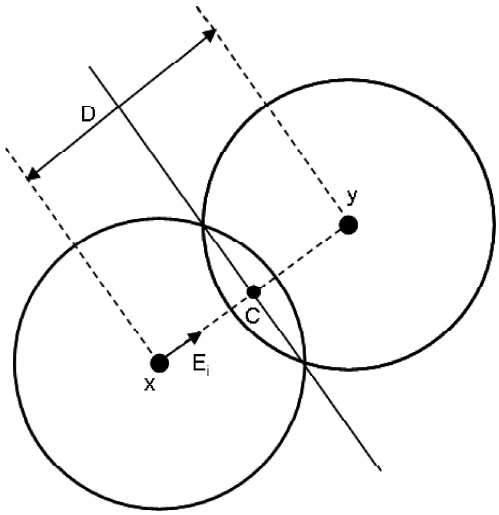

Figure 3.6: Contact model.

The velocity of the impact is given by:

$$v_c = \dot{X}_i E_i + \dot{X}_j E_j, \tag{3.12}$$

where E is defined as the unit vector pointing out from x_1 to point x_2, and is given by:

$$E_i = \frac{y_i - x_i}{D} = (\cos\alpha, \sin\alpha). \tag{3.13}$$

The impulse due to the collision is now given by:

$$p = E\ v_c\ m_x\ m_y, \tag{3.14}$$

where m_x and m_y are the masses of particles. The difference in velocity produced by the impulse force can be calculated with:

$$dv_{x1} = v_{x1} - \frac{p}{m_{x1}}, \tag{3.15}$$

where v_{x1} is the velocity of a particle at time $t - \Delta t$. With equation 3.15 the new direction and velocity of a particle can be estimated.

However, the total amount of collision increases exponentially if the total amount of grains grows. It is time inefficient if all grains should be checked for collision pairwise. Therefore, advanced collision detection applied, where the two dimensional space is divided into square cells. If a grain is located in a cell, that cell will be marked with *state occupied*, such that the cell and particle are linked. Now only the neighboring cells where a grain is located should be checked for occupation. If so, and both particles are collided, the particle from the neighboring cell will be add into a grain-list. Problems

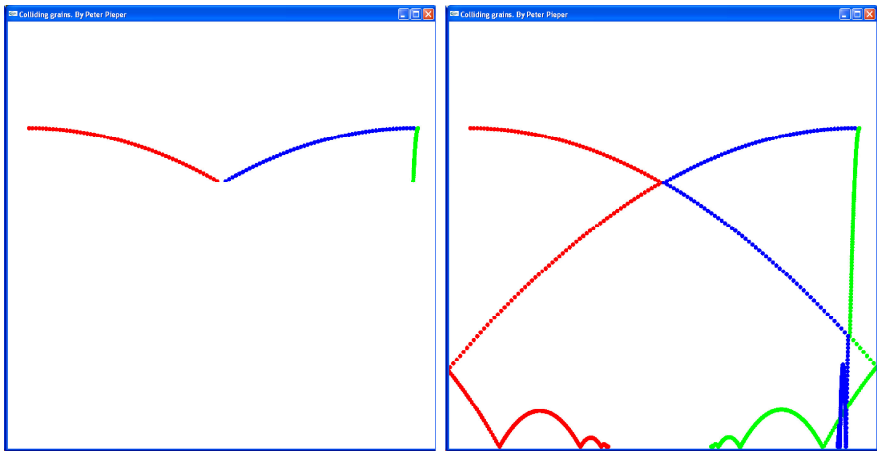

Figure 3.7: Grains before and after collision.

arise when the possibility exists that a chain of grain particles grows, which means that the first particle is not directly in contact with the last grain particle. Therefore, the grain-lists of all grains that are merged together, are put into a contact-list. Verifying the model is done with the assumption that contacted grains can be treated as two directly contacted grains like figure 3.6. However, this is not a realistic solution. The unit vector E (see equation 3.13) should be shifted through the chain and be multiplied with all unit vectors in it's path. This demands a more complex implementation, where a chain should be iterative updated. Hereby, provides the fluid model a structural example of iterative updating. In contrary to the two grain models and the simplified chain model, these can be updated after every iteration step.

Unfortunately due to the lack of time it was not possible to implement a full scale model of a granular heap. However, it can be predicted that the DEM model provide more insight into the dynamics of a granular heap.

3.5 Findings

Three model approaches are developed to research the dynamics of granular heap formations. These three models are reviewed here and compared with the expected dynamics and test results.

It can already be concluded that for control design these models does not provide an answer. However, they do give insight into the dynamics. For control design a suitable model is needed. The ideal model is a suitable approach. Along with the insights of the dynamics, exceptions to the ideal model can be made.

3.5.1 SOC model approach

The application of a SOC model is appropriate for modeling the avalanches of a granular heap. These avalanches are also observed during the preliminary experiments. Even so, the SOC model shows that grains not instantaneous come to rest if a unit of granular material is added to the pile.

The test-measurements have shown that the force "fades" out after the rest-drop is fallen. The SOC model gives an appropriate approximation for this behavior. These two phenomena are mentioned in chapter 1.3 and that it causes settling behavior of the granular material.

It is already mentioned that the SOC model is not suited for model control design. However, the SOC model approach can give an answer to the exceptions of the ideal model. So will, the repeating pattern of an avalanche result into a repeating impulse force on the weighting scale.

The impulse force given in equation 1.10 can be expand with the knowledge of the dynamics of the granular pile. The added mass at the top will be the decisive factor. Note, the added mass causes an avalanche. A pseudocode of an implementable realistic model as function of the impulse force can be:

Example 3.2

```
if(added mass > critical mass)
{
    F_imp = v * mass-flow-rate + Fa;
}
else
{
    F_imp = v * mass-flow-rate;
}
```

where F_a is the added avalanche force factor. Even so, conditions should be set for the duration of an avalanche.

Expansion of the update rules gives a more accurate model with respect to falling grain particles. However, the model is not suited for implementation of dynamic forces. The tests have shown that the top of the pile is smoothed and dented due to the impulse force of fallen granular material. This in contrast to the pile growth of the SOC model, which top is cone shaped. If more advanced update rules are used into the SOC model, the smoothen top of a real pile can be modelled. Just as the SOC model can provide a contribution to the impulse force it can provide height information of a pile if advanced update rules are modelled. This height information should be implemented into the ideal model such that it provides an estimated rest-drop force. The height information should be based on simple criteria like it is done with the impulse force.

3.5.2 Fluid model approach

Although, fluid model approach provides an accurate prediction of the variations of the velocity v_0, the behavior of rolling grains are lost. The calculations are based on the knowledge of the boundary conditions. Therefore, this model approach can only be used for modeling granular flow within the silo. Outside the silo no boundary conditions can be set and so, no calculations can be carried out.

The variation, of the velocity v_0, has shown that the mass flow rate is not homogeneously distributed. This will imply that the mass flow rate should be scaled with an average value of the variation of v_0. The equation 1.8 can be rewritten as:

$$\dot{m} = \alpha(v_0 A \rho), \tag{3.16}$$

where α is the average value of the velocity v_0.

3.5.3 DEM model approach

DEM models provide a realistic behavior of granular particles. The direct implementation of Newton's laws makes it possible to determine the dynamic forces during simulation.

Simulations with a DEM model has shown that grains bounce and roll after they hit walls or each other, which imply that grains has to settle before a static flow exists and before they come to rest after they hit the pile. Settle behavior after the grains hit the pile is already observed with the SOC model approach and during tests.

The DEM model should be used for modeling sub-problems. As mentioned the SOC model give insight into the avalanche dynamic effects. Even so, the SOC approach give an answer how to update the ideal model with a avalanche force factor. However, it does not provide an answer to the value of the avalanche force factor. By implementing a DEM model for this specific problem, the calculated dynamics can estimate a value for the avalanche force factor. Even so, for implementing a model for the output of a silo, the DEM model can estimate a value for the average value, α, of the velocity v_0 which is given in equation 3.16. With the prediction of the height produced by the advanced SOC model approach, the ideal model is suited for model control design.

Chapter 4

Conclusions & recommendations

4.1 Conclusions

During the last months an ideal model of fallen granular material and a real setup provided insight into the dynamics of fallen granular material. However, for fast and accurate dosing an amount of granular material, this ideal model does not satisfy. Therefore three model approaches are used to model the behavior and dynamics of granular material.

The SOC and fluid model approach proven not to be suited for a unified model, such that it can be used for control design of an amount of granular material. Nevertheless, the models gave a great insight into the behavior of granular heap formation. The fluid model approach is based on Poisson's law (equation 3.3) and it depends on the pressure distribution function $f(x, y)$. For a realistic flow model this function should be found, but finding this function is a difficult task. With the SOC and fluid model approaches the ideal model can be improved such that it can be used for a model control design. The DEM model approach gave the best expectations with respect to the dynamic values. This due to the fact that forces, velocities and positions are directly implemented into the model. The SOC and fluid model gave insight in how the ideal model can be improved, but it did not provide the values for the parameters. By implementing a DEM model for these specific problems these parameter values can be estimated.

A full scale model can be made with the DEM model approach. However, implementing a large amount of grains demands a enormous amount of computing power. This can become unpractical and so, a full scale model is not suited for model control design.

Just as the improved parameters all the models should be scaled with a real setup. If so, the model parameters should be determined. The identification of the model parameters demands more time.

The use of OpenGL and X11 provided real-time simulations of the models. It is illustrated that the 2D graphics of X11 can not beat the benefits of the 3D graphics of OpenGL. The use of OpenGL is easy and well documented and so, easy to learn.

4.2 Recommendations

The used models did not end up with a control model for granular material. If all dynamics are understood and they are representable, a suitable model approach should be found for granular heap control. The ideal model discussed in chapter 1 should be a starting point of this model.

Exceptions to the equations can be made with the knowledge of the dynamics of the granular pile. For example the impulse force equation 1.10 can be expand with a constant force factor F_a which is caused by an avalanche. Even so, the mass flow equation 1.8 can be expand with a factor α. This model adaption is discussed in section 3.5.

Another control approach can be chosen for controlling an amount of granular material. Control the amount of rest-drop is hereby the starting point for fast weighting. Height information is the deciding factor for the amount of rest-drop if the velocity is assumed to be constant during a weight-load-cycle. If all circumstances during different weight-load-cycles are the same, a simple table can provide height information with respect to the desired amount of weight. This table can iterative be learned by the controller. However, often the material composition differs from time to time, which leads to different behavior of the granular material. The complexity of granular behavior expands if the differences in composition increases, which imply that the information in the height table increases. For a real setup this is a unpractical situation.

A combination of a table and measured data can become a solution, this due to the fact that the fist impulse response and the mass flow rate are related to the composition of the material. For example if the moister of the granular material increases, the weight of the material increases and so, the fist impulse increases. Even so, it is mentioned in section 1.3 and predicted that settling behavior of granular material influence the impulse force. This settling behavior depends also on the composition of the material.

And so, before combining the composition and measurements this behavior, the relations between the dynamics and the composition should be fully understood. Therefore, more research in the field of material behavior should be done.

One major drawback should be taken into account. Iterative learning for large amounts of granular material can become unpractical. Therefore, it is recommended that a suited model approach should be found.

The competence of a DEM model could not be tested. Therefore, it is recommended to test the competence of the DEM method with a realistic case situation.

The predicted decreasing height h, chapter 2, is not observed during the preliminary experiments. To validate these predictions the real setup should be vary. Even so, predictions with respect to the dimensions of the setup can be validated, this can be done by changing the real setup. Even so, at all times during the real setup validations it must be reviewed that all dynamics are taken into account and so that no unexpected dynamics exists.

OpenGL is recommended for simulating the evolution of FEM calculations. Output of the SOC model, fluid model and DEM model has shown that OpenGL can provide real time output during the simulations.

Bibliography

[1] http://www.x.org

[2] http://www.opengl.org/

[3] Robert W. Scheifler Massachusetts Institute of Technology Laboratory of Computer Science
X Window System Protocol

[4] Jon Leech
GLX Extentions For OpenGL Protocol Speccification (Version 1.3)

[5] Bak, P., Tang, C., and Weisenfed, K. (1988)
Self-organized criticality.
Phys. Rev. A, 38:364-373

[6] J.C. Tannehill, D.A. Anderson, R.H. Plecher
Computational Fluid Mechanics and Heat Transfer second edition
Taylor & Francis ISBN 1-56032-046-X

[7] P.A. Cundall, O.D.L. Stack
A Discrete numerical model for granular assemblies.
Geotechnique, **Vol. 29,** No. 1, 1979, 47-65

[8] Experimental study of the Neumann and Dirichlet boundary conditions in two-dimensional electrostatic problems

Appendix A

Simulation

A.1 Introduction

Often finite element methods (FEM) or discrete element methods (DEM) are used to model complex systems. These calculations are often time consuming and the outcome of the model is often represented with a pictures or put into raw data files for evaluation. Even so, raw data files or pictures do not provide the interaction that is needed for model changes during simulation. For example the slope of the SOC model approach can not be altered if only the outcome of the model is simulated. With the use of dynamic simulation the slope of the SOC model can be changed without waiting for the complete model simulation to be finished. Even so, the behavior of grains in the DEM model could not be reviewed without realtime simulation. Therefore, real time simulation is needed. To do so, fast output to the screen is required. Convenient programs like Matlab does use graphical hardware acceleration, however, the overhead of the program itself makes fast algorithm slow. Additional, for implementing mathematics, the basic principles should be understood. This in contrast to conventional programs where standard library functions can be used. Even so, these library functions can contain extra features what causes extra runtime. For fast modeling and simulating this is an unwanted effect. Even so, conventional applications do not always supports the appropriate functionality for the given problem. Therefore, there is chosen to review two graphical libraries and to implement model code into a c-code.

For fast simulations, two open-source graphical libraries are reviewed. These libraries are X11 and OpenGL. Why X11 and OpenGL? Because of the fact that X11 and OpenGL have hardware acceleration and are widely used. OpenGL and X11 are easy to learn and the tutorials are excellent. A third well known graphical application programming interface is DirectX. However DirectX is only supported by Microsoft Operating systems, DirectX is difficult to learn and documentation is not always helpful.

A.2 Usage

X11 and OpenGL are software interfaces to graphical hardware. X11 is runs on a UNIX based operating system. OpenGL on the other hand runs on every major operating system including Mac OS, OS/2, UNIX, Windows 95/98, Windows 2000, Windows NT and Linux. However OpenGL is supported by X if it is used on a UNIX based operating system. The software interface consist of functions that can be used to generate objects in a specified X or OpenGL window. These functions are brought together into a library. (The GL of OpenGL stands for Graphics Library.) The libraries do not contain complicated shapes such as automobiles, parts of bodies or airplanes. With these libraries, you must create your own desired models from a set of geometric primitives like points, lines and polygons. These libraries are freely available and should be installed before usage [1][2]. The library functions are based on a protocol [3][4]. The protocol defines a client-server relationship between an application and its display. The X client can be run on a remote machine which has access to the X server to display a graphical window. The X client only has to support the X protocol with respect to the possibility to send a request to the server, receive an event from the server and receive errors from the server. With the use of the protocol X and OpenGL request functions can be written, this for usage on a small embedded systems with no supporting X client operation system. In that case the embedded system is defined as the X client who has access to a UNIX based operating system that displays and support the X server. Example A.1 shows a C-code of a function DrawLine. This function uses the X11 protocol instead of the library function to draw a line.

Example A.1

```
int DrawLine(int sFd,int Xs, int Ys, int Xe, int Ye, int Color) {
   struct _ReqGC
   {
      xChangeGCReq req;
      CARD32 ForeGround;
      CARD32 ClipMask;
   }ReqGC;

  struct _ReqSegment
   {
      xPolySegmentReq req;
      xSegment Line;
   }ReqSegment;

   ReqGC.req.reqType  = ChangeGC; /* opcode defined as integer 56*/
   ReqGC.req.length   = sizeof(ReqGC) >> 2;
   ReqGC.req.gc       = GCid;
   ReqGC.req.mask     = GCForeground | GCClipMask;
   ReqGC.ForeGround   = ColorTrans[Color];
   ReqGC.ClipMask     = None;

   if(write(sFd, (char *)&ReqGC, sizeof(ReqGC)) == ERROR)
```

```
   {
      return 0;
   }
   ReqSegment.req.reqType  = PolySegment;  /* opcode defined as integer 66*/
   ReqSegment.req.length   = sizeof(ReqSegment) >> 2;
   ReqSegment.req.gc       = GCid;
   ReqSegment.req.drawable = DynaWindow;
   ReqSegment.Line.x1      = Xs;
   ReqSegment.Line.y1      = Ys;
   ReqSegment.Line.x2      = Xe;
   ReqSegment.Line.y2      = Ye;

   if(write(sFd, (char *)&ReqSegment, sizeof(ReqSegment)) == ERROR)
   {
      return 0;
   }
   return 1;
}
```

The line drawing request consists of two protocol arguments. The first argument is a request to change the graphic context of an X window. The second argument is the actually line drawing. Similar constructions are used for drawing arc, rectangles and polygons. The difference is the opcode used to identify the request type and the structure Segment. Line drawing with the use of OpenGL is basally the same. In this project the library functions are used to program simulations.

A.3 Implementation

A program can become complicated because so many things can be drawn with a OpenGL or X11 graphic system. However the basic structure of a useful program can be simple. Figure A.1 shows a flowchart of a basic program. First of all you have to define and open a display where you want to display a window. This can be local or remote with the use of X or only local with the use of OpenGL. After a display is opened a root window can be opened. All drawings can placed in the root window. However, the root window can also be divided into subwindows. Now the drawings can begin. As already suggested X and OpenGL are fundamentally the same. The greatest difference is the drawing dimension. OpenGL offers 2D and 3D graphic functions in contrary X offers only 2D graphics. Figure A.2 shows a root window with two subwindows, made with X running on a Linux operation system. One subwindow is used for a 2D plot of the data and the second subwindow is used as a button-screen. Figure A.3 shows a OpenGL program running on Windows XP with only a root window with a 3D output of a flow through a pipe.

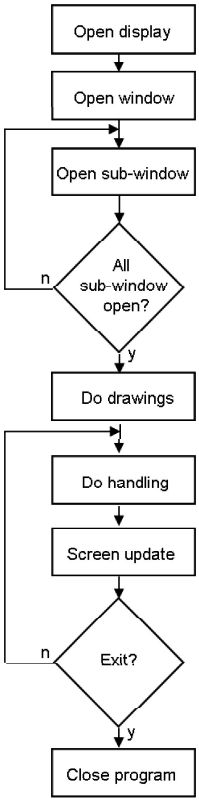

Figure A.1: flowchart of a basic program.

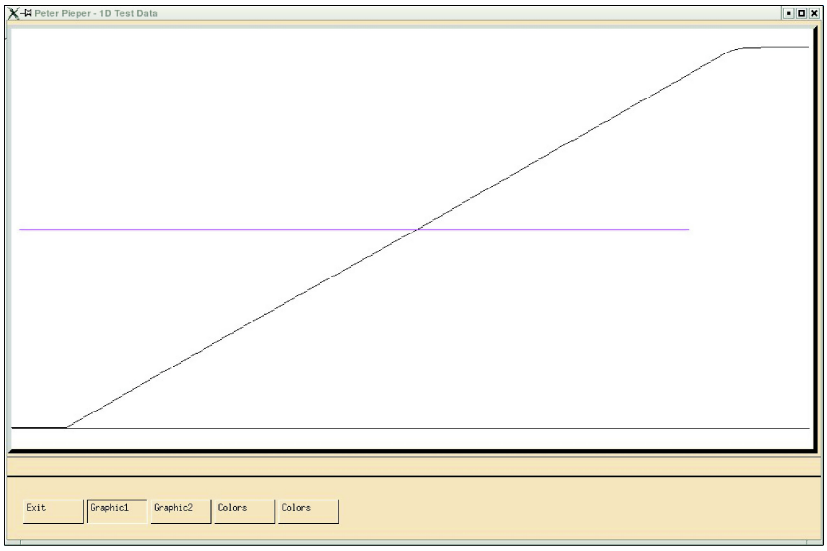

Figure A.2: Output plot with X11.

A.4 X11 versus OpenGL

Although the suggestion is made that OpenGL and X is the same, there are some major differences. This concerns the initialization of a window and so the user ability. For a simple OpenGL graphic window you have to set the lighting, model shading, view position, etcetera. X, however, has to allocate every color and font before usage. For more advanced graphics OpenGL has its advance. In contrast to X11, OpenGL supports 3D graphic functions. X11 is standard supported on a Linux operation system. In contradiction OpenGL has to be installed before use. Finally X windows system cannot beat the benefits of OpenGL.

A.5 Program implementations

One of the main reasons to write own model and simulation applications was, that it is faster at runtime level than conventional applications. But to make and keep applications fast, knowledge about fast code implementation is required. During the project codes are constant reviewed for redundancy and updated. However, no runtimes are measured for comparison some constructions are known as “slow” runtime processes. Next example will illustrate one of these constructions.

Example A.2

```
void allocate() {
```

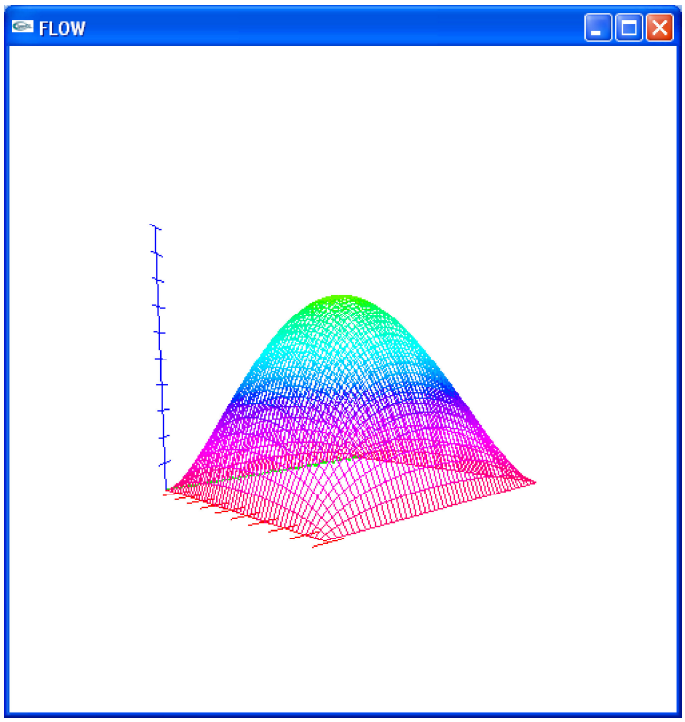

Figure A.3: Flow through a duct.

```
    static int oldL = 0;
    if (oldL != L) {
        if (slope) {
            for (int i = 0; i < oldL; i++) {
                delete [] slope[i];
                delete [] stable[i];
                delete [] height[i];
            }
            delete [] slope;
            delete [] stable;
            delete [] height;
        }
        height = new int* [L];
        slope = new int* [L];
        stable = new bool* [L];

        for (int i = 0; i < L; i++) {
            slope[i] = new int [L];
            height[i] = new int [L];
            stable[i] = new bool [L];
        }
        oldL = L;
    }
}
```

Example A.2 shows a code with three dynamic arrays. In the example the arrays are redefined. New space should be allocated if dynamic arrays

expand during runtime. This is a time consuming process. If the length of an array is a priori known, than it is time saving to define an array as in example A.3.

Example A.3

```
int slope[L][L];
int height[L][L];
int stable[L][L];
```

Example A.2 is a C++ solution. This should not imply that C++ cannot be used. There are C++ code examples that give a faster solution. And so, there are C codes that should also be avoided. As an example, the C solution for example A.2 will give some similar construction with `malloc`. The given example should only apply that some constructions should be avoided. Figure A.3 shows a plot of a flow through a duct. Here, the grid size is pre-defined to avoid constructions like `malloc` or `new`.

After *handling* (see figure A.1) output to the screen is required. Often this *handling* consists of one single iteration step of a model. It is not always required to update the screen if a single iteration step is fast. Therefore, there is chosen to update the screen after every 40 milliseconds to save runtime. This implies that every second 25 frames will be produced, similar to a video stream.