SELF-CALIBRATING MASS FLOW SENSOR

BY

RYAN E. REINKE

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Adviser:

Professor Harry Dankowicz

Abstract

A comprehensive analysis was conducted for increased accuracy and self-calibration for a mass flow sensing system on a combine. This was undertaken as part of the John Deere Technology Innovation Center (JDTIC)-sponsored research program "Self-calibrating mass-flow sensor", in turn part of a John Deere Moline Technology Innovation Center (MTIC) effort toward optimization and closer integration of the components of the mass-flow sensing system in Deere harvesting combines. The long-term objective was to achieve a self-calibrating sensor system capable of adapting to varying input conditions due, for example, to changes in grain moisture content and aging of the system's elevator paddles.

In analyzing the mass flow sensing system, a physics-based model was developed to describe the relationship between the rate of mass flow through the combine and the measured force imparted to the impact plate in terms of mechanical properties of the grains and the interior geometry of the combine. A computational realization of this model was constructed in MATLAB. Accurate mass flow rate estimation was achieved through model-based estimation based on nonlinear regression applied to the physics-based model and data acquired through simulation and experimentation. Model-based estimation was also extended as a means for self-calibration of the sensing system. Through development of the physics-based model, the dependence of the force imparted to the impact plate on the orientation of the impact plate was identified. By inducing known changes to the impact plate orientation and implementing model-based estimation, a means of self-calibration of the sensing system was achieved.

Three methods of model-based estimation were successfully demonstrated using data generated from the physics-based model. Additionally, these were further verified using data collected from discrete element modeling simulations, and experimental data collected in two fashions: using a full-scale replica of the mass flow sensing system, and using a small-scale, benchtop testing apparatus. Furthermore, the ability of the developed algorithm to update theoretical model parameters while simultaneously estimating mass flow rate was shown to enable the system to self-calibrate. This was argued to allow the system to accommodate different operating conditions that may be encountered during combine harvesting, such as changes in crop moisture, grain variety, and aging of combine components.

Acknowledgments

This thesis would not have been possible without the support of many people. Many thanks to my adviser, Harry Dankowicz, who devoted a great deal of his time and effort to discuss and aid in the development of the concepts for this work, and for spending his time to truly advise and teach throughout the various aspects of this effort. Thanks to John Deere and Co. for their grant to make this project possible, and to Jim Phelan, Staff Engineer with John Deere Intelligent Vehicle Systems, for his direction and numerous hours spent in consultation for the development of experiments, the results of which were an integral part of this work. And finally, thanks to my parents: to my mother for her encouragement and support throughout graduate school, and to my father for his engineering design insights.

The following outlines contributions made to this thesis by a variety of sources:

- Chapters 1-9, written for a progress report for Deere and Company, were authored by the author of this thesis along with the assistance of Harry Dankowicz and contain feedback from Jim Phelan, and portions of chapter 9 originated from work by Wonmo Kang.
- Portions of the thesis, excluding material related to self-calibration concepts, have been submitted for publication in the journal of Precision Agriculture and the SAE Commercial Vehicle Engineering Congress.
- Small-scale laboratory experiments were performed by the author of this thesis in the Applied Dynamics Laboratory at the University of Illinois at Urbana-Champaign, and discrete element modeling simulations were performed by the author of this thesis at the John Deere Technology Innovation Center.
- Data was obtained from Deere and Company from large-scale experiments conducted at the Combine Yield Monitor Test Facility.

Table of Contents

Chapter 1

Introduction

Several mechanisms can be implemented to monitor grain flow through a combine. These are typically grouped into volumetric flow sensors, mass flow sensors, and indirect measurement devices. Volumetric flow sensors include paddlewheels [6] and optical devices mounted in the clean grain elevator [23, 26]. Mass flow measurement principles include plate impact sensors [11], torque sensors mounted on the elevator paddles [11], torque sensors mounted on the clean grain elevator drive shaft [27], a diaphragm impact sensor [21], a pivoted-auger-fed device [25], and weighing of the clean grain auger [12]. Indirect methods encompass a capacitive sensor [7], an ultrasonic sensor [7], and x-ray measurement principles [4].

Many state-of-the-art commercial implementations rely on impact-type sensors and empirical relations between measured impact force and the rate of mass flow. Impact-type sensors are simple structures that allow for independent operation and reduced risk for material build-up. They consist of an impact plate and a force transducer that converts the net time-averaged impact force into a voltage signal. Typically, curve-fit schemes are used to characterize the relation between the impact force and the rate of grain mass flow. However, these are highly dependent upon the conditions at which calibration is performed. For instance, significant errors can result in estimating mass flow rates at a certain threshold above calibration flow rates [10], and increased errors have also been observed for low flow rates [17].

A schematic realization of an impact-based mass-flow sensor system is shown in Figure 2.1. Here, elevator paddles are filled with grain via an auger and are attached to a chain that is cycled by a rotating sprocket. As the paddles rotate around the sprocket, grain is propelled towards an impact plate. As momentum is lost in the subsequent collision, an effective force is measured on the impact plate. This measured force, along with knowledge of the dynamics of the system, allow for the mass flow rate of the grain to be estimated.

Simple linear models have been employed in the past to relate the rate of mass flow and the impact force [15, 22, 5, 20]. In contrast, experimental results exhibit a strongly nonlinear dependence of the impact force on the rate of mass flow at larger flow rates, (e.g., observations in [21]). Additionally, model-

based designs have been proposed for mass flow sensors to account for changes in material properties [2, 22]. Assuming that model parameters are known, the model described in this thesis proposes a physical mechanism for relating the impact force to the rate of mass flow in terms of the frictional interactions between grains, collisions between the grains and the impact plate, the geometric nonlinearities associated with the free-flight motion of grains upon leaving the elevator paddles, and the orientation of the impact plate.

A challenge to a practical implementation of such a nonlinear model is the lack of knowledge of model parameters that characterize the grain behavior, for example effective coefficients of friction and restitution and their dependence on grain moisture levels. Similarly, mechanical aging of system components, such as elevator paddles, affects the values and physical interpretation of model parameters. Similar difficulties arise in the use of empirical models in which model coefficients lack physical origin. As a result, repeated in-field calibration of a mass-flow-sensor system is typically required to reset model parameters. Coupled with a model that accurately captures the relationship between mass flow rate and impact force, sensors that attempt to measure the total moisture content of grain may aid in the effort to provide real-time estimates of model parameters, as it is noted that grain frictional and shape properties change depending upon the moisture content and variety of grain [19].

It is desirable that commercial mass-flow-sensor systems allow for flow rate measurements in the range of 0-25 kg/s with errors below 5% and sampled at rates on the order of 1 Hz [18]. Mass-flow sensing components should be closely integrated into the combine and be easily accessible for service. A commercially highly competitive system should require no more than 1 manual sensor calibration per crop per season. Finally, such a system should afford a certifiable method for measuring the accumulated mass in ranges exceeding 2000 kg with absolute errors below 1% (to be used in marketing and crop sharing decisions) and relative errors below 3% (to be used in yield comparisons).

Chapter 2

Figures

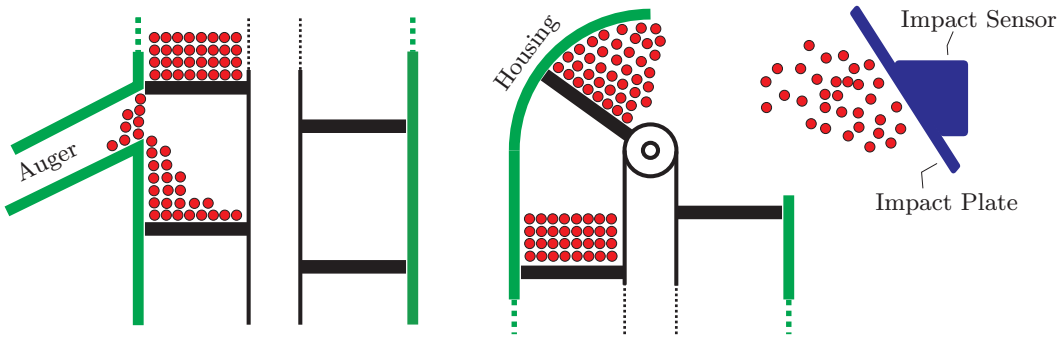

Figure 2.1: Schematic representing the operation of the mass-flow sensor system. Grain enters the system via an auger and is deposited onto elevator paddles. The elevator paddles are attached to a chain that is cycled by a rotating sprocket. As the paddles rotate over the top of the sprocket, grain is propelled toward an impact plate which measures impact force.

Chapter 3

Models

The fundamental objective of the mass-flow-sensing system is to provide accurate yield estimates during real-time operation of harvesting combines under a variety of operating conditions. To achieve this objective it is necessary to arrive at a relationship between the input(s) and the output(s) that can be reliably interrogated to *predict* outputs given measured inputs or to *estimate* inputs given measured outputs. Such a relationship will be referred to in this section of the document as a *model*.

In the present discussion, we distinguish between three different types of models, namely, *data-based*, *empirical*, and *physical* models.

3.1 Data-based models

Data-based models consist of numerical look-up tables, indexed by a collection of experimentally collected values of inputs and containing the corresponding experimentally collected values of the outputs. Prediction is then achieved by interpolating the content of the table elements whose indices agree most closely with the given input conditions. Similarly, estimation is achieved by interpolating the indices of the table elements whose content agree most closely with the given output conditions.

In the simplest case, suppose that, under otherwise constant conditions, experimental values have been collected for the amount of mass that flows past a given cross-sectional area per unit time and the corresponding values of the time-averaged force on the impact plate. The look-up table is then equivalent to a two-dimensional graph of the discrete force data versus the mass flow data. Under the same otherwise constant conditions, prediction and estimation is the straightforward act of locating intersections of vertical and horizontal lines with a suitably chosen interpolant (e.g., piecewise linear, spline, fourier).

In the case of data-based models, model development is accomplished through an act of *calibration* in which (possibly vast quantities of) data is experimentally collected to generate an approximate cover of typical operating conditions. Given the uncertainty in the collected experimental data and the difficulty in controlling for operating conditions outside of the measurable inputs, data-based models would

typically only provide reliable operation in severely constrained applications.

3.2 Empirical models

Empirical models seek to experimentally identify functional relationships between dimensionless combinations of input and output variables.

In the case of the mass-flow sensor, the ratio

$$\alpha = \frac{\langle m \rangle V}{\tau \langle F \rangle} \tag{3.1}$$

is a dimensionless combination of the average mass $\langle m \rangle$ that flows past a given cross-sectional area over some characteristic time τ, the average force $\langle F \rangle$ on the impact plate during this characteristic time, and the translational velocity V of the elevator paddles during their vertical ascent. An empirical model based solely on this dimensionless quantity is then a functional relationship of the form

$$f(\alpha) = 0, \tag{3.2}$$

where f is some arbitrary (dimensionless) function of a single variable. This constitutes a (nonlinear) equation in α with (typically, at best) isolated solutions of the form

$$\alpha = \alpha_0 \Rightarrow \dot{m} \stackrel{def}{=} \frac{\langle m \rangle}{\tau} = \alpha_0 \frac{\langle F \rangle}{V} \tag{3.3}$$

for some dimensionless constant α_0. Calibration of this model simply amounts to computing α_0 estimated from a large statistical sample of numerical values for α given otherwise constant conditions.

Reliance on the single dimensionless quantity α amounts to the modeling assumption that the value of α_0 is independent of $\langle m \rangle$, V, τ, and $\langle F \rangle$ and only a function of other, not controlled-for, operating parameters that describe the experimental apparatus. The extent to which this can be reliably assumed can be statistically evaluated by exploring the distribution of values of α for otherwise constant conditions. Biasing behavior as a function of any one of the four fundamental quantities $\langle m \rangle$, V, τ, and $\langle F \rangle$, then suggests a need to enlarge the model to (at least) one additional dimensionless quantity.

Consider, for example, the dimensionless combination

$$\beta = \frac{\eta w l^2 V \Omega}{\langle F \rangle} \tag{3.4}$$

of the grain volume density η, the width of the elevator paddle w, the length of the elevator paddle l, the translational velocity V of the elevator paddle during the vertical ascent, the angular speed of the sprocket Ω, and the average force $\langle F \rangle$ on the impact plate during the characteristic time τ. An empirical model based on α and β is then a functional relationship of the form

$$f(\alpha, \beta) = 0, \tag{3.5}$$

where f is some arbitrary (dimensionless) function of two variables. This constitutes a (nonlinear) equation in α as a function of β with (typically) a continuum of solutions. Suppose, for example, that

$$f : (x, y) \mapsto xy - c_1 y - c_2, \tag{3.6}$$

in which case

$$\alpha\beta - c_1\beta - c_2 = 0 \Rightarrow \dot{m} = c_1 \frac{\langle F \rangle}{V} + c_2 \frac{1}{\eta w l^2 \Omega} \left(\frac{\langle F \rangle}{V} \right)^2 \tag{3.7}$$

for some dimensionless constants c_1 and c_2. Calibration of this model amounts to computing c_1 and c_2 by fitting the straight line $c_1\beta + c_2$ to the product $\alpha\beta$ using a large statistical sample of numerical values for α and β given otherwise constant conditions.

The systematic reliance on dimensionless quantities allows one to enlarge the model complexity in successive steps. From this perspective it would be inappropriate to bulk together c_2 with $\left(\eta w l^2 \Omega\right)^{-1}$ into a single non-dimensionless quantity, as such would immediately suggest a hidden dependence on unmodeled parameters.

Moreover, a systematic approach to identifying suitable empirical relationships is to first identify all possible (measurable) dimensionless quantities and then to investigate experimentally observed relationships between these. This amounts to experimentally identifying the function f first and only subsequently (and when possible) arriving at an explicit relationship for $\dot{m}$ as a function of other measurable quantities.

3.3 Physical models

Physical models are arrived at through a successive causal association between inputs and outputs at various stages of the physical process. The model for mass flow measurements of sugar beets reported in [16] is a successful example of such an approach. In the case of the model development undertaken for the present effort, the physical process has been divided into four stages, namely,

1. the *filling* stage, in which a certain mass of grain is deposited on an elevator paddle;

2. the *settling* stage, in which the mass deposited on the elevator paddle achieves an equilibrium configuration while the paddle is rotated in space through the interaction with the sprocket;

3. a *release* stage, in which angular sectors of mass slide outwards following the disappearance of the physical constraint imposed by the enclosure; and

4. a *flight stage*, in which individual grains travel from the paddle to the impact plate and deposit part of their momentum with the plate, thus resulting in an effective time-averaged force $\langle F \rangle$.

A complete description of each stage of the physical process requires assumptions on the conditions of the system at the onset of each stage, on the behavior of the system during each stage, and on the conditions of the system at the conclusion of the each stage. For example, in the case of the filling stage, it is assumed that no mass is deposited on the paddle prior to the onset of the stage; that mass is deposited onto the paddle during the stage based entirely on the rate of mass flow across the available opening cross section and not based on the amount of mass currently deposited onto the paddle; and, finally, that the amount of mass deposited onto the paddle during the filling stage equals the amount of mass present on the paddle at the conclusion of the filling stage so that no leakage is allowed for.

Similarly, it is assumed that the amount of mass present on the paddle at the onset of the settling stage equals that present at the conclusion of the filling stage; that no mass is lost during the settling stage; and that the velocity distribution across the deposited mass at the end of the settling stage equals that of a rigid body of some shape undergoing pure rotation about the center of the sprocket. In particular, the input-output relationship for the settling stage corresponds to a description of the shape of the corresponding rigid body for a given amount of mass present on the paddle at the onset of the settling stage. In the developed model, a functional relationship parametrized by the total available mass, has been proposed between the radial thickness of the mass distribution and the angular displacement from the elevator paddle. Finally, the settling stage is assumed to conclude at different times for different angular sectors of the equilibrium mass distribution, as each such sector reaches the limit of the physical constraint imposed by the enclosure.

It is assumed that, at the onset of the release stage, the grains in each angular sector have achieved a slight non-zero radial velocity with increasing values the further the distance from the center of the sprocket, so that no radial interactions are present between grains during the release stage. Moreover, each grain is assumed to experience frictional interactions with grains in adjacent sectors governed by a (dimensionless) coefficient of friction μ. Finally, the release stage is assumed to conclude at different

times for grains at different initial radial distances from the center of the sprocket, as each such grain reaches the distal end of the elevator paddle. Using conservation of momentum, an implicit relationship can then be found between the initial radial position of a grain and its velocity vector at the conclusion of the release stage. For typical operating conditions, this relationship is largely unaffected by gravity and is given by the value of a known nonlinear function evaluated at the elapsed time for which the distal end of the elevator paddle is reached.

Finally, it is assumed that at the onset of the flight stage, each grain has the velocity vector achieved at the conclusion of the release stage, that it is unaffected by the presence of other grains during the flight stage, and that its interactions with the impact plate (should a collision occur) can be described in terms of the laws of conservation of momentum, a dimensionless kinematic coefficient of restitution e for the change in relative normal velocity, and a conserved relative tangential velocity. For typical operating conditions, the relationship found between the momentum lost in such a collision and the initial grain velocity is largely unaffected by gravity but depends on the position and orientation of the impact plate. In particular, the latter determine not only the fraction of grains that collide with the plate, but also the amount of momentum lost in such a collision.

The composition of the input and output relationships for each stage of the physical process described above results in an implicit, but computable, (algorithmic) relationship between the input and output variables for the overall process. As in the case of an empirical model, this relationship could, in principle, be written in terms of a functional relationship between dimensionless quantities. In contrast to the empirical model development, however, in the physical model, model coefficients would be explicitly expressed in terms of physical quantities, including those describing the mechanical properties of the grains; the geometry of the elevator paddles, the enclosure, and the impact plate; and the laws of physics.

Chapter 4

Estimation

Three distinct uses of model-based regression are possible in the context of the physical model described in the previous section, and to a lesser extent the (quasi-)empirical models currently in use. In particular, these will be referred to as *calibration* or *system identification*, *open-loop estimation*, and *closed-loop estimation*.

4.1 The use of regression and optimization

Calibration or system identification is the task of determining the values of model parameters that are not directly accessible to measurement. In the case of the empirical models described in the previous section, this would include estimation of the values of α_0 or c_1 and c_2 under different experimental conditions. In the case of the physical model described in the previous section, this would include estimation of the values of the coefficient of friction μ, the coefficient of restitution e, the density of the grain η, and a variable describing the shape of the grain on the paddle h_f during the settling stage, under different experimental conditions, for example, under varying grain moisture content.

In the context of the physical model, it is assumed that a computable function f has been arrived at, such that

$$\langle F \rangle = f\left(\dot{m}, \mu, e, \eta, h_f, \mathbf{p}\right), \tag{4.1}$$

where $\mathbf{p}$ is a collection of system parameters that describe the otherwise constant experimental conditions (assumed independent of moisture). Given a large sample of values of $\langle F \rangle$ and the corresponding values of $\dot{m}$ for a given moisture, nonlinear regression could now be employed to estimate (calibrate) μ, e, η, and h_f in order to achieve a close agreement between the predictions from (4.1) and the experimental data. By repeating this for different values of moisture, a data-based and/or empirical relationship between moisture (already a dimensionless quantity) and μ, e, η, and h_f could be arrived at. In each case, the success of the nonlinear regression would depend on the quality of the initial guess for the values of μ, e, η, and h_f as well as on the relative "flatness" (or insensitivity) of f with respect to variations in μ, e,

η, and h_f.

In open-loop estimation, a single real-time measurement of $\langle F \rangle$ would need to be paired to an estimated value of $\dot{m}$ given known (or approximately known) values of μ, e, η, and h_f. Here, optimization would be employed to minimize the difference between the single predicted value from (4.1) and the experimental measurement of $\langle F \rangle$. Given a known value for the moisture, the empirical or data-based predictions of the values of μ, e, η, and h_f could be used as fixed inputs to the model. Employing a bisection algorithm would facilitate the computation of $\dot{m}$.

Finally, in closed-loop estimation, variations in $\mathbf{p}$ would be intentionally introduced so as to generate a large sample of values of $\langle F \rangle$ for a single value of $\dot{m}$ for a given moisture. Here, nonlinear regression could be employed to estimate μ, e, η, h_f, and $\dot{m}$ in order to achieve as close agreement between the predictions from (4.1) and the experimental data. Initial guesses for μ, e, η, and h_f could be obtained from the system calibration process. The advantage of the closed-loop estimation is its ability to adapt to varying moisture conditions and crop varieties.

Both open-loop and closed-loop estimation are possible using empirical models, although the physical significance of the estimated variables (e.g., c_1 and c_2) is not as immediate. Indeed, while the physical model is appealing from the point of view of establishing causal relationships between inputs and outputs at various stages during the physical process, it must be recognized that it relies on a series of assumptions whose validity must reasonably be questioned. Physical models must typically be supplemented by some numerical flexibility in model coefficients, so that while individual assumptions may not be well supported by experimental data, the overall model still captures some essential characteristics of the process. Thus, μ, e, η, and h_f when matched against experimental data may not really correspond to the coefficient of friction, coefficient of restitution, density, or shape variable in the original mechanical sense, but may still suffice to achieve satisfactory agreement between model predictions and experiments.

The three distinct categories of model-based estimation described above can be conveniently analyzed in terms of a block diagram in which numerical values provided to certain leads enable the computation or estimation of a consistent set of numerical values for the remaining leads. In the context of (4.1), consider the block diagram shown in Figure 5.1. Here, $\dot{m}$, μ, e, η, h_f, $\mathbf{p}$, and $\langle F \rangle$ refer to a collection of consistent numerical values of the four different leads for which the content of the block represents a truism. It follows that when presented with limited information about the numerical values of one or several leads, sufficient data for the remaining leads and the requirement that the block be trivially satisfied would allow for the estimation tasks described previously. The block representation thus demonstrates how numerical values for not-directly-measurable parameters may be inferred from knowledge of other inputs/outputs of the black box.

For example, in the case of calibration, it is assumed that $\mathbf{p}$ represents a collection of known numerical quantities, and that collections of pairs of consistent values of $\dot{m}$ and $\langle F \rangle$ are available (cf. Figure 5.2a). The condition on consistency now imposes constraints on the values of μ, e, η, and h_f. Similarly, in open-loop estimation, it is assumed that $\mathbf{p}$ again represents a collection of known numerical quantities and that values of μ, e, η, h_f, and $\langle F \rangle$ are available (cf. Figure 5.2b). The condition on consistency now imposes a constraint on the value of $\dot{m}$. Finally, in the case of closed-loop estimation, several known collections of numerical values of $\mathbf{p}$ combined with consistent numerical values for $\langle F \rangle$ enable the regression-based estimation of consistent numerical values for μ, e, η, h_f, and $\dot{m}$ (cf. Figure 5.2c).

4.2 Mechanisms for self-calibration

It has been noted that when using a yield monitoring system "the most important factor in achieving good accuracy was good calibration" [13], and that the requirement for calibration of yield monitors each time they are used is an inconvenient procedure and one that is prone to error [1]. These calibration difficulties are further confounded by the need to calibrate over the entire range of flow rates encountered during harvest [3, 17] and for each type of grain that is to be harvested [14], as well as the need to *recalibrate* to account for changes in grain moisture contents or crop conditions [14]. The method of self-calibration proposed in this document seeks to eliminate these shortcomings associated with calibration.

The physical model developed shows a dependence on the position and orientation of the impact plate that could be exploited to achieve the closed-loop estimation proposed above. Here, the position and/or angle of orientation of the plate could be varied over a time scale that includes at least a single period of averaging the impact force so as to yield variations in $\langle F \rangle$ that in turn could be used to estimate $\dot{m}$ as well as model parameters used during open-loop estimation. As suggested previously, the position and orientation of the plate affect the force measurements both as a result of variations in the number of grains that actually collide with the plate, but also as a result of variations in the amount of momentum lost during such collisions.

Alternatives to changing the position and orientation of the plate are to vary the angular velocity Ω of the sprocket, which in turn would cause a variation in the speed of the elevator paddles. This would affect the force measurements as a result of variations in the number of grains deposited onto the paddle during the filling stage and consequently variations in the number of grains that actually collide with the plate, as well as the velocity at which the grains collide with the plate.

A further candidate strategy for enabling self-calibration during closed-loop estimation might be the use of additional information regarding the total mass flow gathered by a weighing mechanism that

would collect grains over many cycles of operation and then compute a total mass at discrete sample intervals. Such a setup would be equivalent to the following mathematical formulation

$$\begin{aligned} \langle F \rangle_1 &= f\left(\dot{m}_1, \mu, e, \eta, h_f, \mathbf{p}\right), && (4.2) \\ &\vdots && (4.3) \\ \langle F \rangle_n &= f\left(\dot{m}_n, \mu, e, \eta, h_f, \mathbf{p}\right), && (4.4) \\ \dot{M} &= \sum_{i=1}^{n} \dot{m}_i, && (4.5) \end{aligned}$$

where $\langle F \rangle_i$ and $\dot{M}$ would be experimentally measured quantities and $\dot{m}_i$, μ, e, η, and h_f would be unknown quantities to be estimated. Although knowledge of $\dot{M}$ would further constrain the problem of finding values for $\dot{m}_i$, μ, e, η, and h_f, the number of unknowns still exceeds the number of equations.

In contrast, the closed-loop estimation scheme proposed previously is equivalent to the following mathematical formulation

$$\begin{aligned} \langle F \rangle_1 &= f\left(\dot{m}, \mu, e, \eta, h_f, \mathbf{p}_1\right), && (4.6) \\ &\vdots && (4.7) \\ \langle F \rangle_n &= f\left(\dot{m}, \mu, e, \eta, h_f, \mathbf{p}_n\right), && (4.8) \end{aligned}$$

where $\langle F \rangle_i$ and $\mathbf{p}_i$ are measured quantities and $\dot{m}$ (assumed effectively the same across all the different values of $\mathbf{p}$), μ, e, η, and h_f would be unknown quantities to be estimated. Here, a surplus of equations provides more statistical strength to the regression analysis.

Chapter 5

Figures

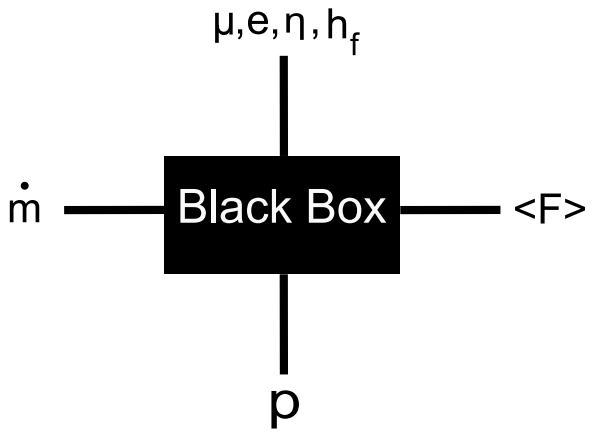

Figure 5.1: Diagram representing the system as a black box with various related inputs and outputs.

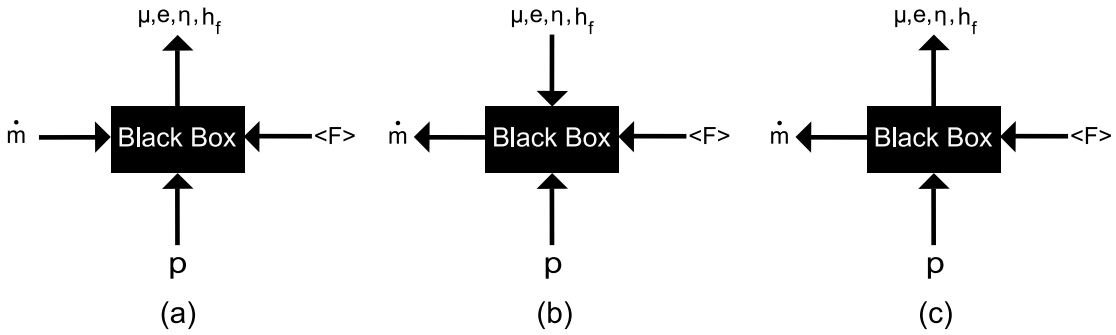

Figure 5.2: Diagrams representing the system as a black box with inputs and outputs for the purpose of (a) system calibration, (b) open-loop estimation, and (c) closed-loop estimation.

Chapter 6

Computational tools

Two distinct computational tools are used extensively in this effort, MATLAB and the discrete element modeling software, EDEM. As described below, these have distinctly different uses.

MATLAB is a computational environment in which user-defined algorithms can be developed that rely on built-in routines for automating common numerical tasks, such as root finding, sorting, and nonlinear regression. In contrast to compiled code, MATLAB is typically used in a parsed mode where scripts and functions are parsed and executed in real time. As a result, MATLAB is probably at least 20 times slower than compiled code.

In this particular effort, MATLAB has been used, for example, to implement a bisection algorithm for numerically locating the elapsed time for an individual grain to reach the distal end of the elevator paddle and for determining the cutoff distance from the center of the sprocket for collisional contact between individual grains and the impact plate. The bisection algorithm is a standard algorithm that would equally well lend itself to implementation in compiled code, say using C. MATLAB has further been used to implement the nonlinear regression algorithm. The development of optimization algorithms, such as those for nonlinear regression, is not quite as straightforward as bisection and there is probably significant advantage to using existing and tested code versus building such code in-house.

In conclusion, MATLAB presently serves as an implementation of the computable relationship f in (4.1) but does not, in itself, constitute a modeling tool. In contrast, EDEM is a simulation tool that enables one to investigate the physical process described in the four stages above based on more fundamental mechanistic assumptions about the rigidity of the environment and the properties of individual grains. In the ongoing effort, EDEM has been used to explore the modeling assumptions employed in the physical model for each of the four stages and to parametrize the shape of the equivalent rigid body at the conclusion of the settling stage. EDEM could also be used as a validation tool for the three uses of model-based estimation described above, for example to evaluate the ability of the regression analysis to estimate mechanical parameters such as μ and e even under somewhat questionable assumptions regarding the input and output relationship for each stage.

6.1 Matlab Implementation

As described above, the implementation of the computable relationship f in (4.1) is accomplished within MATLAB in terms of an algorithm that, given values for $\dot{m}$, μ, e, η, h_f and $\mathbf{p}$ generates a consistent value for $\langle F \rangle$. This is referred to as a forward computational model with a well-defined collection of inputs and a unique and directly computable output. Specifically, the MATLAB implementation is broken down into individual .m functions (see Appendix N) that compute the relevant inputs at each stage of the physical process.

In order to invert the relationship between the physical quantities as described in the estimation part of the report, it is necessary to couple the forward computational model with an optimization or regression algorithm that iteratively updates estimated values of the input variables in order to achieve convergence of the predicted values of $\langle F \rangle$ with existing reference values. In the case of calibration, this has been achieved by wrapping the forward computational model within the MATLAB function *lsqnonlin*. This function seeks to achieve agreement between predicted values of $\langle F \rangle$ and existing reference values by minimizing the sum of squares of errors given a suitable initial guess for the unknown parameters. The MATLAB implementation uses a subspace trust-region method that avoids the need for explicitly computed error gradients. As with most optimization methods, *lsqnonlin* provides at best a local minimum in the computed sum of squares of errors and might not converge to the globally optimal set of values of the unknown parameters unless the initial guess lies sufficiently close to this set (or, more generally, in the basin of attraction of this global minimum). Finally, *lsqnonlin* allows the user to constrain the optimization to bound the values of the unknown parameters to closed intervals.

Figure 7.1 illustrates the computation flow in the MATLAB implementation of the calibration task. Schematically, the *lsqnonlin* block consists of five discrete states, namely the 'start', 'compute', 'condition', 'update', and 'terminate' states. The *lsqnonlin* block initiates in the 'start' state and terminates in the 'terminate' state.

- In the 'start' state, initial guesses for μ, e, η, and h_f are input together with known values for the elements of $\mathbf{p}$ and collections of known reference values for $\dot{m}$ and $\langle F \rangle$. The state of the *lsqnonlin* block then becomes 'compute'.

- In the 'compute' state, current values of μ, e, η, and h_f together with known values for the elements of $\mathbf{p}$ and collections of known values for $\dot{m}$ are used as inputs to the forward computational model which returns predicted values of $\langle F \rangle$. The state of the *lsqnonlin* block then becomes 'condition'.

- In the 'condition' state, the sum of squares of differences between predicted and reference values

of $\langle F \rangle$ is computed. If this quantity exceeds a lower cutoff and no other interrupt conditions are satisfied, then the state of the *lsqnonlin* block becomes 'update'. Otherwise, the state of the *lsqnonlin* block becomes 'terminate'.

- In the 'update' state, the optimization algorithm is employed to provide improved numerical estimates for the values of μ, e, η, and h_f and the state of the *lsqnonlin* block returns to 'compute'.
- Finally, in the 'terminate' state, the current numerical estimates for the values of μ, e, η, and h_f are returned to the outside operating system.

In the actual implementation, several instances of the 'compute' and 'update' states may be combined in generating improved numerical estimates for the values of μ, e, η, and h_f. Figure 7.2 shows the comparison of the experimental data and predicted force as obtained from the forward-computational model using, on the one hand, an initial guess for the numerical values of μ, e, η, and h_f and, on the other hand, numerical estimates returned from the regression algorithm. The regression algorithm clearly tunes the values of the internal parameters to reduce the estimation error and to enable close agreement between predicted and measured force values.

6.2 Discrete Element Modeling

The software package EDEM was used to simulate the operation of the clean grain elevator system and to gain insight into the grain behavior during operation. Information available upon completion of EDEM simulations includes the velocity history of individual grains, trends in the shape and movement of the grains during distinct stages of the simulation, as well as the measured force on the impact plate that results from collisions with the grains. This section outlines the setup and operation of the EDEM simulations. The results and observed trends are subsequently referenced in the detailed development of the mathematical model in a later part of this thesis.

In each simulation, varying numbers of physically separated particles representing individual grains are positioned throughout the operating volume with zero initial velocity. A settling phase is then simulated, in which the particles are allowed to fall under the influence of gravity and contact with the enclosure until a state of essential rest has been reached (see Figure 7.3). Data collection under the motion of the elevator paddle only occurs following this state of rest.

In seeking to inform and validate the development of the forward-computational model, it was decided to restrict attention to an idealized portion of the clean grain elevator system, namely a half-cylindrical enclosure, a single rotating paddle, and the impact plate. The half-cylinder was used to simulate the

shape and effect of the housing as a constraint on the grain during operation. All components were designed with dimensions consistent with those in an actual combine; the paddle was given a length of 0.16 m and a width of 0.21 m, the half-cylinder had a radius of approximately 0.16 m, and the lowermost point of the impact plate was located 0.5 m, measured horizontally, from the center of paddle rotation at an angle of 58 degrees, measured clockwise from the positive x-axis.

The simulation was designed to rotate the paddle clockwise about a pivot point such that its motion swept through the inner portion of the cylinder. The paddle was allowed to rotate at a constant speed of 400 rpm for 360 degrees such that grain was propelled towards the impact plate. The process of grain being thrown by the paddle is shown in Figure 7.4. As seen in the figure, color variations can be used to display characteristics of the grain, in this case the grain speed ranging from blue (lowest) to red (highest).

The grain was assigned material properties described in Table 7.1, all geometry components were assigned properties described in Table 7.2, and the interaction properties between all materials are described in Table 7.3.

Chapter 7

Figures and Tables

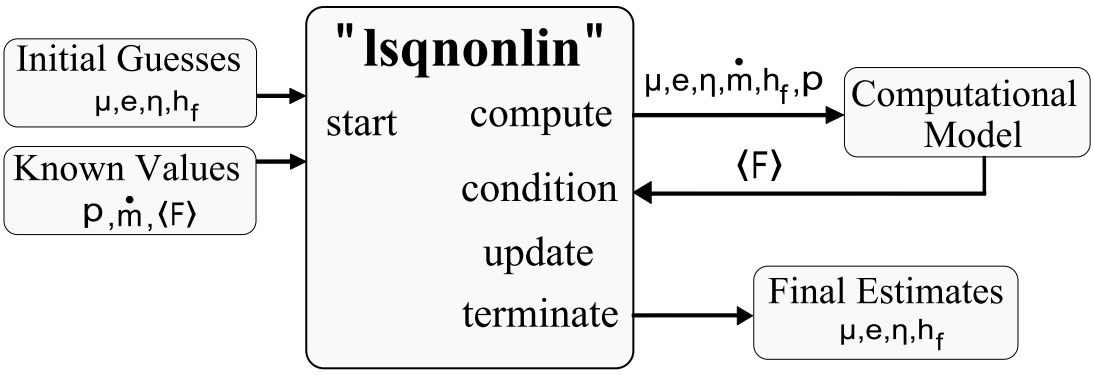

Figure 7.1: Computation flow for the Matlab implementation of the calibration task.

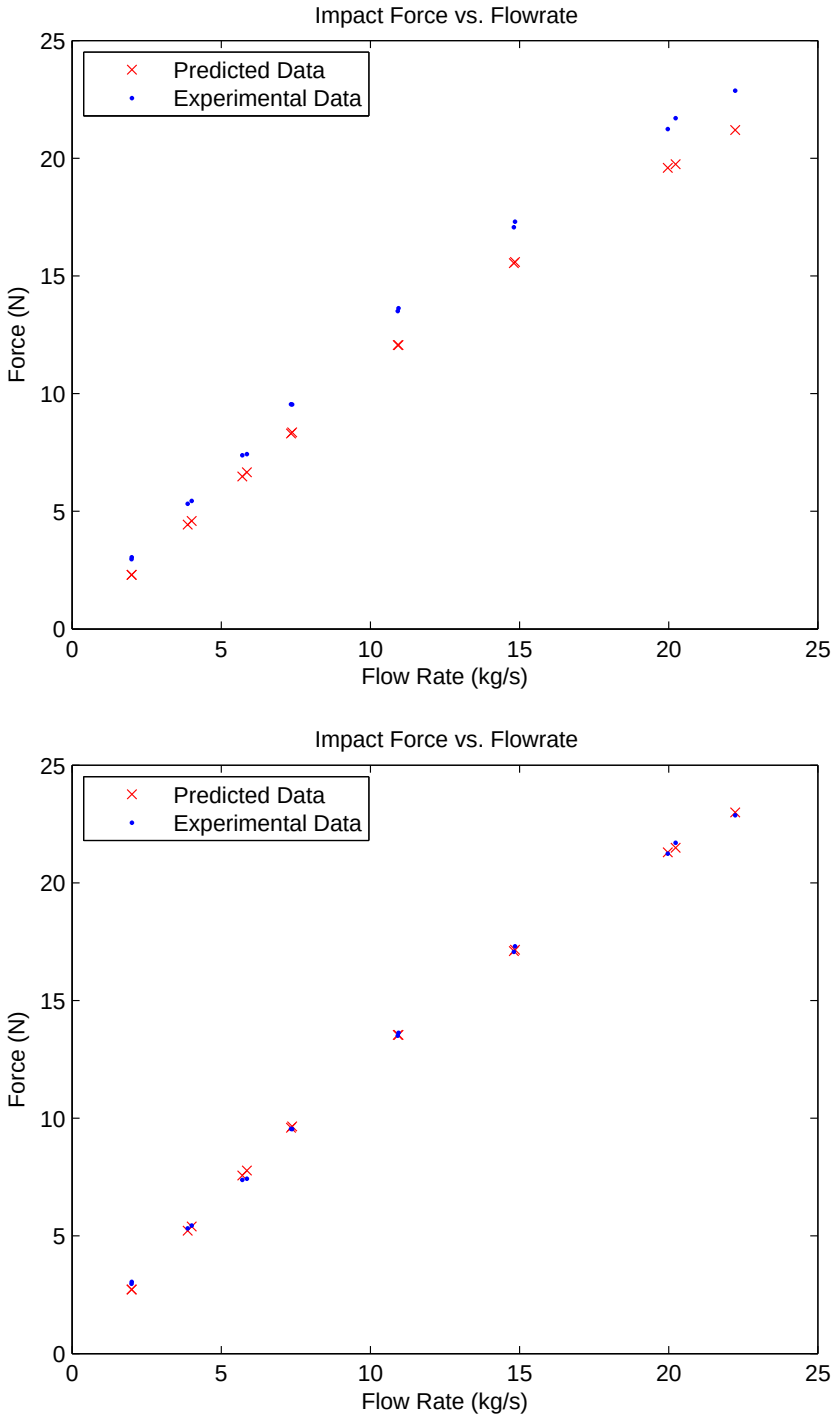

Figure 7.2: Comparison of experimental data and predicted force from the forward-computation model using an initial guess for μ, e, η, and h_f (top), and numerical estimates returned from the regression algorithm (bottom).

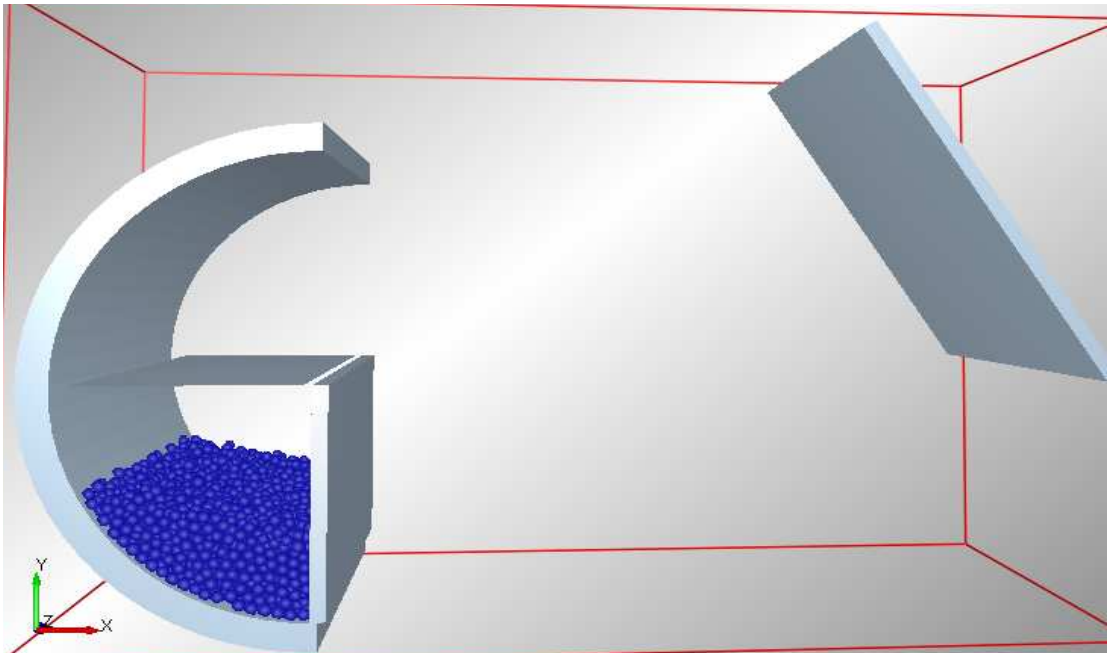

Figure 7.3: Initial setup of the discrete element modeling simulations featuring an enclosure, paddle, impact plate, and settled grain.

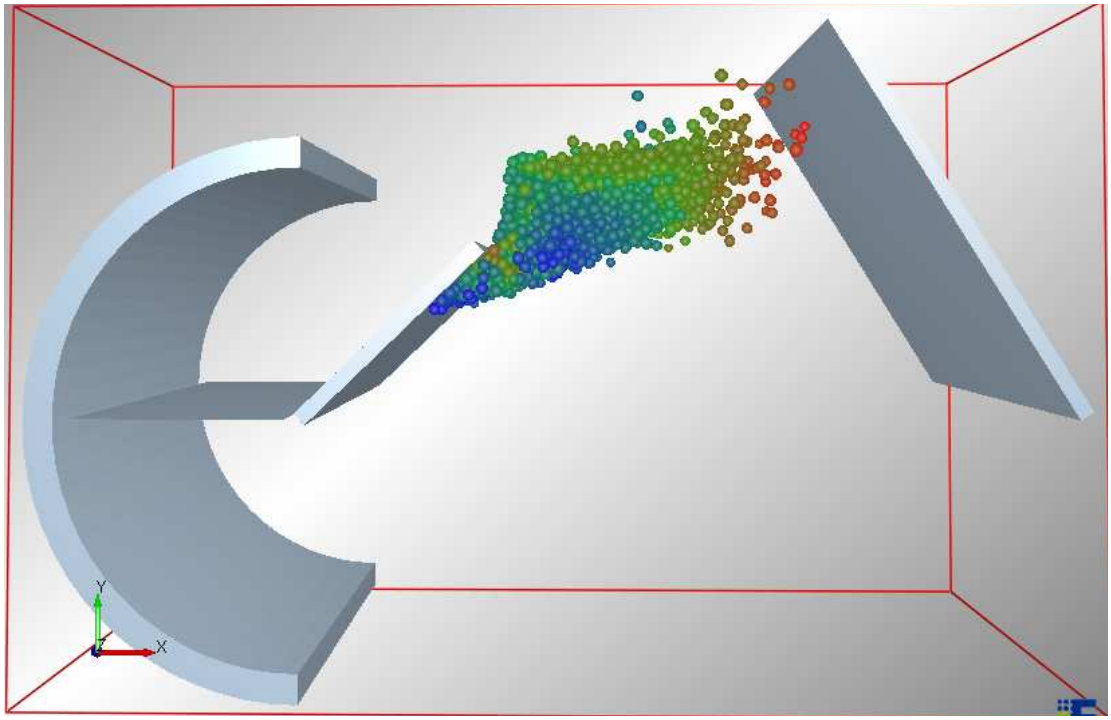

Figure 7.4: Portion of the discrete element modeling simulations demonstrating grain thrown by a paddle toward an impact plate.

Table 7.1: Properties of grain used in development of the mathematical model.

Property	Value
Mass	4.048×10^{-4} kg
Volume	3.037×10^{-7} m^3
Radius	3.870×10^{-3} m
Poisson's Ratio	4.000E×10−1
Shear Modulus	6.300×10^{6} Pa

Table 7.2: Properties of geometry used in development of the mathematical model.

Property	Value
Density	8,000 $\frac{kg}{m^3}$
Poisson's Ratio	3.000×10^{-1}
Shear Modulus	8.000×10^{10} Pa

Table 7.3: Material interaction properties between corn and all materials used in discrete element modeling simulations for development of the mathematical model.

Property	Value
Coefficient of static friction	0.3
Coefficient of rolling friction	0.1
Coefficient of restitution	0.75

Chapter 8

Mathematical Model

As described previously, the flow of grain through the mass-flow sensor system can be conveniently divided into four separate stages: the filling stage where grain is deposited onto a paddle from an auger, the settling stage where the grains are constrained by the motion of the paddle and the presence of the housing, the release stage where the grains are constrained by the motion of the paddle but the constraint of the housing is no longer present, and the flight stage where there are no constraints on the grains as they move freely toward the impact plate.

8.1 The filling stage

This initial stage is one in which grain is deposited on a paddle from an auger located near the bottom of the elevator. Grain exits from a circular opening at the top of the auger and is deposited on the paddles as the paddles move upward.

It is assumed that no mass is deposited on the paddle prior to the onset of the stage; that mass is deposited onto the paddle during the stage based entirely on the rate of mass flow across the available opening cross section and not based on the amount of mass currently deposited onto the paddle; and, finally, that the amount of mass deposited onto the paddle during the filling stage equals the amount of mass present on the paddle at the conclusion of the filling stage so that no leakage is allowed for.

The mass deposited on a paddle is a function of time and is dependent on the percentage of available area of the auger opening for a specific paddle of interest. This is described as:

$$dM = \dot{m} P(t)\, dt \tag{8.1}$$

where dM is the change in mass deposited onto the paddle during an infinitesimal time interval dt and $P(t)$ represents the percentage of opening of the exit of the auger at time t. This percentage of available area for a particular paddle of interest changes as paddles pass by the auger opening as shown in Figure 9.1 and is described by the following:

1. the paddle located ahead of the paddle of interest is passing directly by the auger, such that a portion of the total auger area is available to supply grain to the paddle of interest, and the remaining portion of the auger area is supplying grain to the paddle located ahead of the paddle of interest.

2. the paddle ahead of the paddle of interest has passed the auger completely, such that all of the auger opening is supplying grain to the paddle of interest.

3. the paddle of interest is passing directly by the auger such that the auger is simultaneously supplying grain to the paddle of interest and to the paddle located behind the paddle of interest.

Without loss of generality, assuming a rectangular cross-sectional area and a distance between paddles that is greater than the vertical distance of the auger opening, one obtains

$$P(t) = \begin{cases} \frac{V}{a}t & : 0 < t < \frac{a}{V} \\ 1 & : \frac{a}{V} < t < \frac{d}{V} \\ 1 + \frac{d-Vt}{a} & : \frac{d}{V} < t < \frac{d+a}{V} \end{cases} \tag{8.2}$$

where V is the linear velocity of the paddles, d is the distance between paddles, and a is the vertical distance of the auger opening. In this case, the total mass deposited on a single paddle equals

$$M = \dot{m} \int P \, dt = \dot{m} \frac{d}{V}. \tag{8.3}$$

For a complete derivation for any arbitrary cross-sectional area of the auger, see Appendices A and C.

8.2 The settling stage

This second stage is one in which the grain deposited on the paddle is assumed to settle under the combined influence of gravity, contact with the housing, and contact with the rotating paddle (see also [22]). It is assumed that the amount of mass present on the paddle at the onset of this stage equals that present at the conclusion of the filling stage; that no mass is lost during this stage; and that the velocity distribution across the deposited mass at the end of this stage equals that of a rigid body of some shape undergoing pure rotation about the center of the sprocket.

Denote by l the distance to the distal end of the paddle from the center of the sprocket. Given the above assumptions, the shape of the grain distribution at the end of the settling stage will here be modeled by the dependence of the radial thickness $l - h(\psi)$ on the angle $0 \leq \psi \leq \psi_f$ measured from the

paddle as shown in Figure 9.2. Here, ψ_f represents the maximal angle achieved by the grain distribution.

To arrive at an approximate shape of the grain distribution, discrete element modeling simulations were performed with different numbers of grain deposited onto the paddle. As seen in Figure 9.3, the mass of grain appears to attain a characteristic shape which is largely independent of the number of grains present on the paddle. In particular, as derived in Appendix E, the inner boundary of the grain distribution closely approximates a straight line, consistent with the following assumed function of the angular displacement from the paddle

$$h(\psi) = l\left(\frac{l}{h_0}\cos\psi + \left(1 - \frac{l}{h_0}\cos\psi_f\right)\frac{\sin\psi}{\sin\psi_f}\right)^{-1}, \tag{8.4}$$

which satisfies the boundary conditions

$$h(0) = h_0 \tag{8.5}$$

and

$$h(\psi_f) = l. \tag{8.6}$$

Moreover, while the arclength of the radial layer in contact with the housing, $l\psi_f$, and the thickness of the angular sector resting against the paddle, $l - h_0$, both depend on the number of grains, the ratio

$$h_f = \frac{l\psi_f}{l - h_0} \tag{8.7}$$

was found to be approximately constant and equal to 2 for a range of total grain numbers and fixed angular sprocket speed.

The value of h_0 may be obtained given a total deposited mass M on a paddle, by considering the total volume of grain enclosed by the housing and the rotating paddle. With the profile of the grain characterized by the function $h(\psi)$, Appendix E shows that the cross-sectional area of grain is given by

$$A_{grain} = h_f\frac{l(l - h_0)}{2} - \frac{h_0 l}{2}\sin\left(h_f\frac{l - h_0}{l}\right). \tag{8.8}$$

Given a total mass M of grain deposited onto the paddle during the filling stage and given the grain density η, the width of the paddle w, and the length ratio h_f, h_0 can now be obtained from the unique solution on the interval (0,1) to the nonlinear equation

$$\frac{M}{\eta w l^2} = \frac{A_{grain}}{l^2} = h_f\frac{1 - \tilde{h}_0}{2} - \frac{\tilde{h}_0}{2}\sin\left(h_f\left(1 - \tilde{h}_0\right)\right), \tag{8.9}$$

where $\tilde{h}_0 = h_0/l$. In the forward-computational model implemented in MATLAB this is achieved through bisection on this interval.

8.3 The release stage

This third stage is one in which distinct angular sectors of grain are no longer constrained by the housing but are still constrained by the motion of the paddle. It is assumed that, at the onset of the release stage, the grains in each angular sector have achieved a slight non-zero radial velocity with increasing values the further the distance from the center of the sprocket, so that no radial interactions are present between grains during the release stage. Moreover, each grain is assumed to experience frictional interactions with grains in adjacent angular sectors governed by a (dimensionless) coefficient of friction μ (e.g. Figure 9.2). Finally, the release stage is assumed to conclude at different times for grains at different initial radial distances from the center of the sprocket, as each such grain reaches the distal end of the elevator paddle.

As shown in Figure 9.4, discrete-element-modeling simulations demonstrate a trend of higher radial velocities for individual grains located in radial layers further from the sprocket than for those at closer radial distances. This can be further explained by noting that once the constraint imposed by the housing is removed, the grains in the radial layer furthest from the sprocket are acted on in the radial direction only by a force from the grains in the adjacent radial layer nearer to the sprocket. This causes the grains in the furthest radial layer to accelerate radially outward, which in turn causes the force between those grains and the grains in the adjacent radial layer to decrease to the point that the force is zero. The effect is a separation between grains in these radial layers. This trend continues for radial layers approaching the center of the sprocket and results in no contact forces between grains in adjacent radial layers, in agreement with the assumption made on the interactions during the release stage.

The travel time of grains, t_d, from their initial location to the distal end of the paddle was also examined using discrete-element-modeling as shown in Figure 9.5. Although the times of release from the initial position for each grain and the times at which each grain reached the distal end of the paddle differed between the two grains, the elapsed times between these events were nearly equal and measured at 0.016 seconds for Grain A and 0.015 seconds for Grain B. This corroborates the assumption that grains located at the same initial radial position after the settling phase will have the same traveling time from their initial location to the distal end of the paddle.

Given these assumptions, consider a cartesian coordinate system represented by the coordinates x and y as shown in Appendix F. To determine the traveling time of a grain from its initial position to

its arrival at the distal end of the paddle, as well as its release velocity at the conclusion of the release stage, it is assumed that the grain dynamics are constrained to follow the angular motion of the paddle throughout this stage, i.e., that $\dot{\theta} = \Omega$ throughout the release stage, where $\tan\theta = y/x$.

In terms of the nondimensional radial distance $\tilde{\rho} = \rho/l$ from the center of the sprocket, the equation of motion during the release stage for an individual grain is then given by

$$\tilde{\rho}'' = \tilde{\rho} - 2\mu\tilde{\rho}' + \tilde{g}(-\sin\tau - \mu\cos\tau), \tag{8.10}$$

where primes denote differentiation with respect to non-dimensionalized time $\tau = \Omega t$ and

$$\tilde{g} = \frac{g}{l\Omega^2} << 1 \tag{8.11}$$

given the characteristic paddle length $l = 0.16$ m, sprocket rotational speed $\Omega = 400$ rpm, and gravitational acceleration g. Typical non-dimensional travel times are found to be on the order of 10^{-1}. Consequently, the effects of gravity are ignored in the subsequent calculations. Given a grain initially at rest at radial distance ρ_0, the analytical solution for the simplified equation of motion then equals

$$\tilde{\rho}(\tau) = C_1 e^{\left(-\mu-\sqrt{1+\mu^2}\right)\tau} + C_2 e^{\left(-\mu+\sqrt{1+\mu^2}\right)\tau} \tag{8.12}$$

with coefficients

$$C_{1,2} = \frac{1}{2}\left(\tilde{\rho}_0 \mp \frac{\tilde{\rho}_0\mu}{\sqrt{1+\mu^2}}\right). \tag{8.13}$$

Given ρ_0, the traveling time τ_d can be now obtained as the unique solution to the nonlinear equation

$$0 = 1 - \tilde{\rho}(\tau_d) = 1 - C_1 e^{\left(-\mu-\sqrt{1+\mu^2}\right)\tau_d} - C_2 e^{\left(-\mu+\sqrt{1+\mu^2}\right)\tau_d}. \tag{8.14}$$

In the forward-computational model implemented in MATLAB this is again achieved through bisection with an initial guess derived from an approximate solution in the limit that $\mu \to 0$ as shown in Appendix H. The release velocity at the conclusion of the release stage is now given by

$$\mathbf{v}_d = l\Omega \begin{bmatrix} \left(\tilde{\rho}'(\tau_d)\cos\left(\tau_d + \frac{3\pi}{2}\right) - \sin\left(\tau_d + \frac{3\pi}{2}\right)\right)\mathbf{e}_x \\ + \left(\tilde{\rho}'(t_d)\sin\left(\tau_d + \frac{3\pi}{2}\right) + \cos\left(\tau_d + \frac{3\pi}{2}\right)\right)\mathbf{e}_y \end{bmatrix} \tag{8.15}$$

where

$$\tilde{\rho}'(\tau) = C_1 e^{\left(-\mu-\sqrt{1+\mu^2}\right)\tau}\left(-\mu-\sqrt{1+\mu^2}\right) + C_2 e^{\left(-\mu+\sqrt{1+\mu^2}\right)\tau}\left(-\mu+\sqrt{1+\mu^2}\right). \tag{8.16}$$

8.4 The flight stage

This fourth stage is one in which the grains are in free flight and are traveling towards a flat impact plate, as shown in Figure 9.6. It is assumed that at the onset of the flight stage, each grain has the velocity vector achieved at the conclusion of the release stage and that its motion is unaffected by the presence of other grains. Additionally, it is assumed that interactions with the impact plate (should a collision occur) can be described in terms of the laws of conservation of momentum, a dimensionless kinematic coefficient of restitution e for the change in relative normal velocity, and a conserved relative tangential velocity.

In this section, a computable relationship will be derived for determining the portion of grains that contact the impact plate, as well as for the force exerted on the impact plate due to collisions with the grains. This process of mathematically accounting for only the grains that will hit the impact plate differs from the use of a deflector plate to concentrate the flow of all grains toward the impact plate, as proposed by [22]. Use of such a device is sensitive to build-up of granular material near the plate and high friction losses [22].

The flight path of grains after discharge from the paddle was examined using discrete element modeling simulations, as shown in Figure 9.7. From this analysis it was clear that the flight paths of grains can be closely approximated by straight-line motion with unchanged speed, as the short time of flight again implies a negligible contribution due to gravity.

In order to compute the net force acting on the impact plate, it is necessary to determine the portion of grains that collide with the impact plate. Previous modeling has determined that at least 15% to 20% of grains on a paddle full of grains will have a considerable downward velocity component [22], indicating that these grains will miss the impact plate. Given the assumption of straight-line motion, whether an individual grain reaches such a collision is determined by its position and velocity at the conclusion of the release stage. By the discussion in the previous section, these quantities depend only on the initial radial distance ρ_0 and not on the initial angular displacement of the grain relative to the paddle. It follows that there exists a lower bound ρ_{crit} such that grains initially inside of the radial distance ρ_{crit} miss the impact plate due to arriving too late at the distal end of the paddle.

Specifically, the position of a grain located at the initial distance ρ_0 at the conclusion of the release

stage is given by

$$l\cos\theta_d\mathbf{e}_x + l\sin\theta_d\mathbf{e}_y \tag{8.17}$$

where $\theta_d = 3\pi/2 + \Omega t_d$. It follows that the path described by the grain during the flight stage is given by

$$(l\cos\theta_d + v_x\lambda)\,\mathbf{e}_x + (l\sin\theta_d + v_y\lambda)\,\mathbf{e}_y \tag{8.18}$$

Similarly, the geometry of the impact plate is described by the parametrized straight line

$$(d_x + \kappa\cos\phi)\,\mathbf{e}_x + (d_y + \kappa\sin\phi)\,\mathbf{e}_y \tag{8.19}$$

where d_x and d_y correspond to the horizontal and vertical distances, respectively, from the lowermost point of the impact plate to the center of the sprocket, and ϕ is the angle of the impact plate, measured clockwise from the positive x-axis, as shown in Figure 9.6.

Intersections between the flight path and the impact plate thus occur for

$$\lambda = \frac{(d_x - l\cos\theta_d)\sin\phi - (d_y - l\sin\theta_d)\cos\phi}{v_x\sin\phi - v_y\cos\phi} \tag{8.20}$$

and

$$\kappa = \frac{v_x\,(d_y - l\sin\theta_d) - v_y\,(d_x - l\cos\theta_d)}{v_y\cos\phi - v_x\sin\phi}. \tag{8.21}$$

The critical radius ρ_{crit} is now given by the value of ρ_0 for which $\lambda > 0$ and $\kappa = 0$, since $\kappa < 0$ for grains that impact the plate.

From the analysis in Appendix K, it now follows that the loss of momentum of a grain that impacts the plate equals

$$\delta m\,(1+e)\,[(v_{d,x}\sin\phi - v_{d,y}\cos\phi)\sin\phi\mathbf{e}_x + (v_{d,y}\cos\phi - v_{d,x}\sin\phi)\cos\phi\mathbf{e}_y] \tag{8.22}$$

8.5 Impact force

By conservation of momentum, the momentum lost by the grain is momentum gained by the impact plate. The time-averaged force on the impact plate is therefore approximately the total momentum gained by the impact plate through collisions with grains divided by the relevant time interval ΔT. For simplicity, let the characteristic time interval be the time between successive paddles reaching the end of the housing, i.e., $\Delta T = d/V$.

From the analysis in the appendix, the total momentum in the horizontal direction gained by the impact plate as a result of collisions with the grains deposited on a single paddle then equals

$$\Delta P = \begin{cases} \int_{\rho_{crit}}^{1} \left(v_x \sin\phi - v_y \cos\phi\right) \sin\phi \left(1+e\right) \rho h^{-1}\left(\rho\right) \eta w d\rho & \rho_{crit} > h_0 \\ \int_{h_0}^{1} \left(v_x \sin\phi - v_y \cos\phi\right) \sin\phi \left(1+e\right) \rho h^{-1}\left(\rho\right) \eta w d\rho & h_0 > \rho_{crit} \end{cases} \tag{8.23}$$

where η again denotes the volume density, w denotes the paddle width, and $h^{-1}\left(\rho\right)$ computes the opening angle ψ for which $h\left(\psi\right) = \rho$, such that $\rho h^{-1}\left(\rho\right) w d\rho$ is a volume element of a radial layer. Note that h_0 depends in a nonlinear fashion on η.

The time-averaged force $\langle F \rangle$ on the impact plate is now approximately equal to the total momentum ΔP gained by the impact plate through collisions with grains divided by the relevant time interval ΔT:

$$\langle F \rangle = \frac{\Delta P}{\Delta T}. \tag{8.24}$$

Chapter 9

Figures

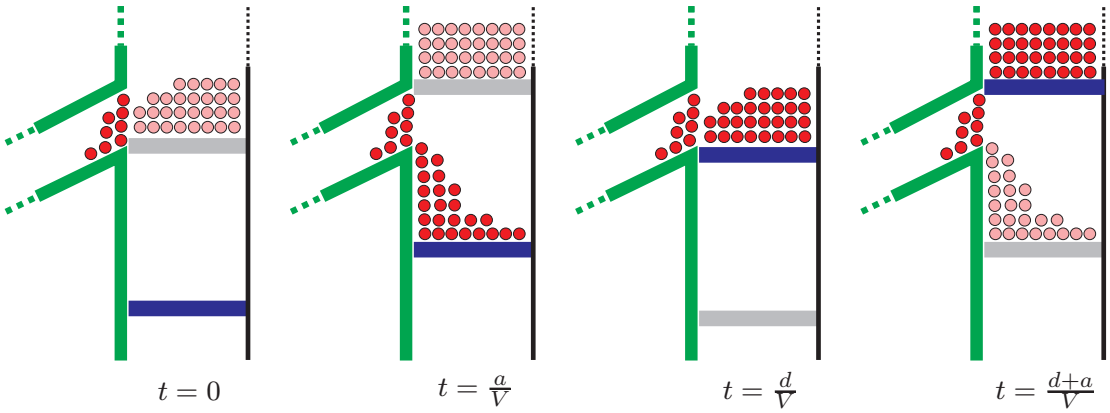

Figure 9.1: Successive stages in the process of depositing grain exiting from an auger onto a single paddle (dark) and onto adjacent paddles (light).

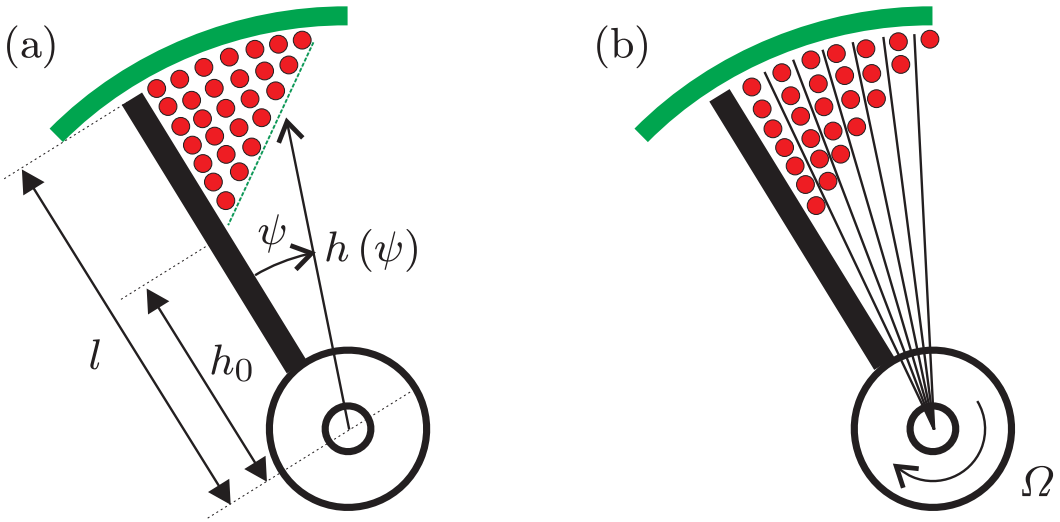

Figure 9.2: (a) Shape of the grain distribution at the end of the settling stage and (b) side view of grains resting on a paddle, demonstrating a continuous distribution of grains divided into angular sectors.

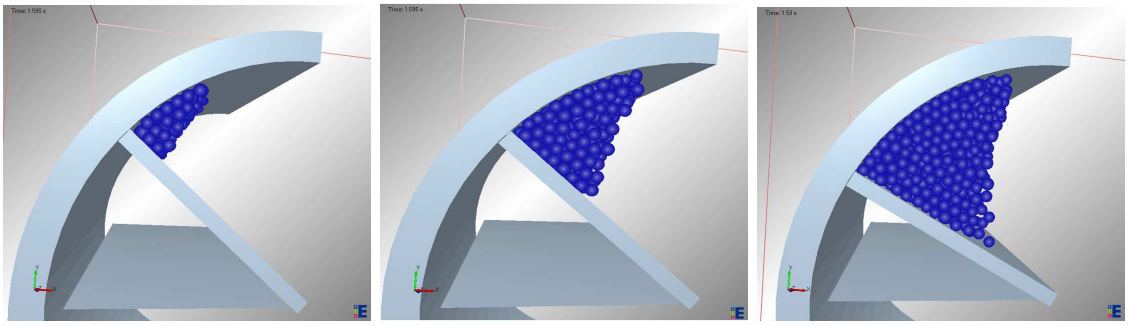

Figure 9.3: Discrete element modeling simulation timestep displaying the distribution of grains on a paddle for the cases of 500 grains (left), 1500 grains (center), and 3500 grains (right). For each case, a characteristic shape is attained by the grains that is largely independent of the number of grains present.

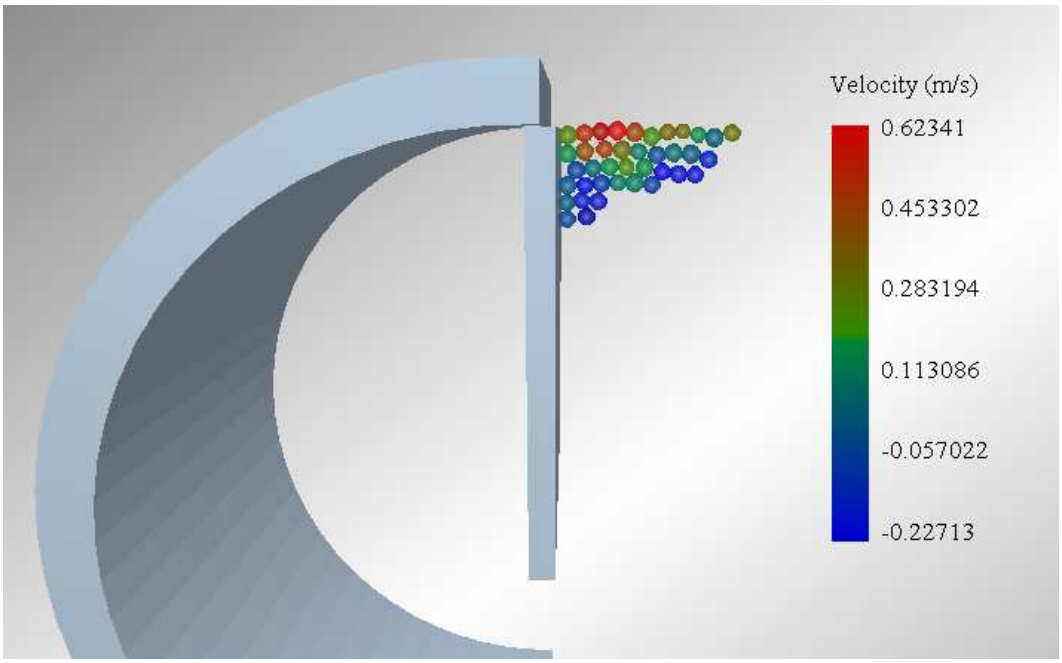

Figure 9.4: Snapshot from a discrete element modeling simulation showing the vertical velocities of grains as the constraint imposed by the housing is removed. Grains furthest from the center of the sprocket exhibit larger vertical velocities (positive upward) than those nearer to the center of the sprocket.

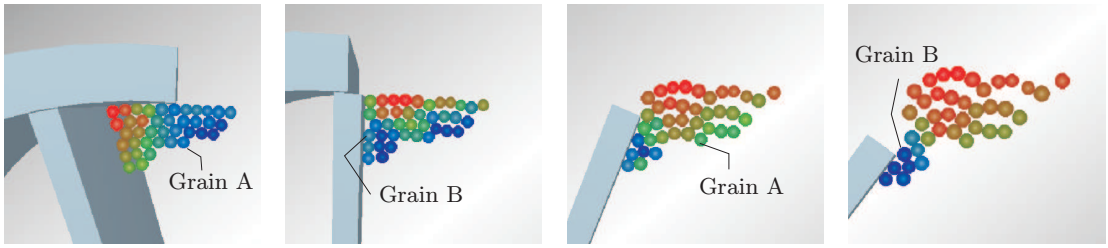

Figure 9.5: Discrete element simulation timesteps showing the state of grains at the beginning and end of their travel to the end of the paddle. The total elapsed travel time for Grain A was 0.016 seconds, and the total elapsed travel time for Grain B was 0.015 seconds.

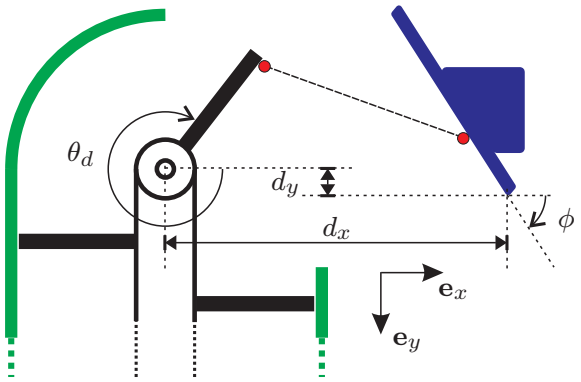

Figure 9.6: Schematic showing the state of an individual grain, the machine geometry, and several parameters describing the machine geometry, and characterizing the travel of a grain from its release from an elevator paddle toward a flat impact plate.

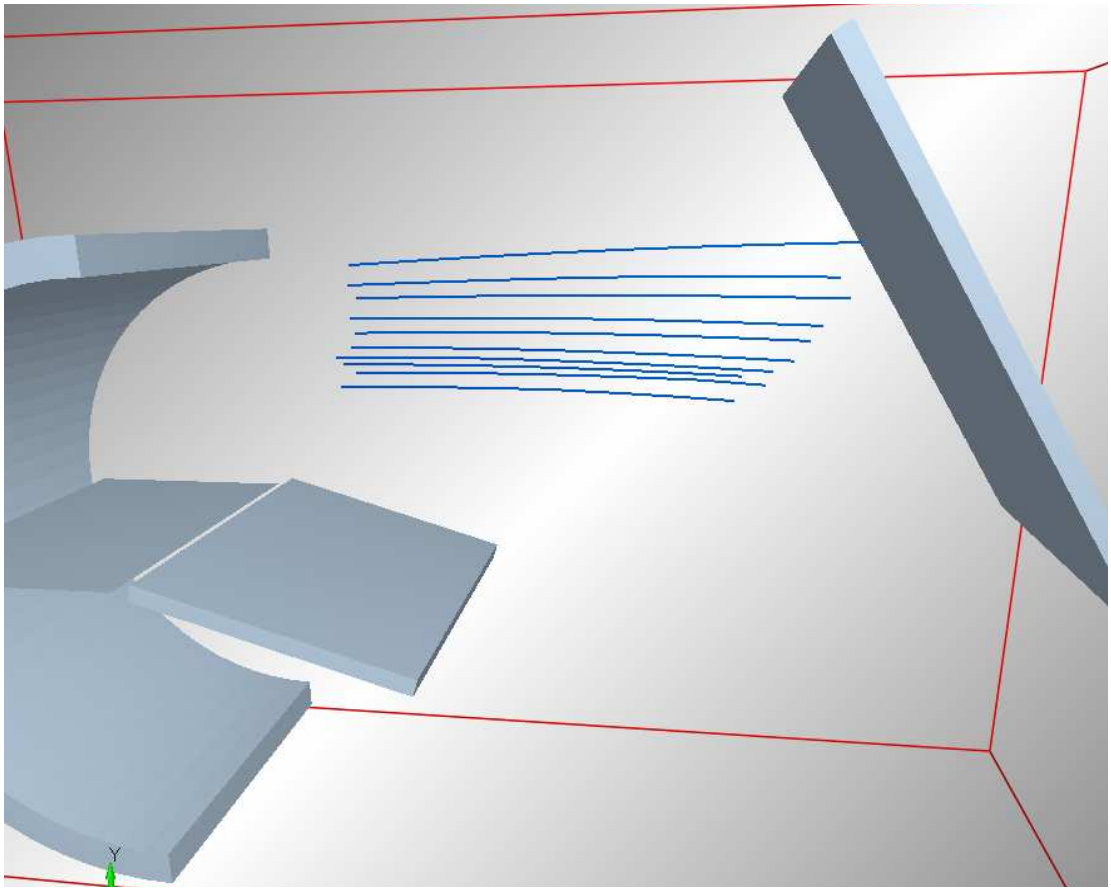

Figure 9.7: Flight path of grains, demonstrating the close approximation of straight-line motion.

Chapter 10

Application of the mathematical model

The mathematical model developed above allows for a prediction of the impact force measured on the plate as a function of the mass flow rate and parametrized by a number of model parameters. In this section we seek to validate the model against data obtained from discrete element modeling simulations, from small-scale benchtop experiments, and from full-scale experiments at the University of Kentucky Combine Yield Monitor Test Facility. Specifically, we seek to ensure that the estimated relationship between mass flow rate and momentum imparted to the impact plate correlates well with experimental data, especially in the nonlinear regions of high flow rates. This is accomplished using a nonlinear regression algorithm, in the context of system calibration, to calibrate the model parameters. In this manner, the model can be fitted for varying grain conditions, such as moisture content, which may have an effect on the frictional characteristics of the system, the coefficient of restitution, grain density, or distribution of grain as it moves through the system. Estimation of mass flow rate is then achievable given constant conditions for model parameters using open-loop estimation. In the case that model parameter values to need to be estimated during real-time operation, the closed-loop estimation routine may instead be used, which simultaneously estimates mass flow rate and self-calibrates the system by updating model parameters.

In the derivation of Eqn. (8.23), it was assumed that, at the end of the settling stage, the grain deposited on a single paddle moves as a rigid body of some shape undergoing pure rotation about the center of the sprocket, i.e., that all radial velocities are equal to zero. This assumption was also relied upon in computing the travel time τ_d for individual grains from their initial radial position to the distal end of the paddle and, consequently, in computing the discharge velocity components. Observations from numerical and physical experiments of the grain dynamics at this junction between the settling and release stages indicate that this approximation may underestimate the outward radial motion at the onset of the release stage. As a consequence, it is believed that the analysis overestimates the proportion of grain that miss the impact plate, an effect that becomes considerably more important for larger mass flow rates. To accommodate this observation, when applying the regression algorithm to experimental

data, the multiplicative factor 1 + e in Eqn. (8.23) is replaced by $\left(k_1 \frac{M}{M_{full}} + k_2\right)$, where M_{full} is the amount of mass, for a given value of the density η, corresponding to $h_0 = 0$.

When implementing the calibration algorithm, μ represents friction, η represents the grain density, and k_1 and k_2 capture the exchange of momentum with the impact plate while compensating for the overly conservative estimates on the number of grains that miss the plate as per the discussion in the previous paragraph. As previously stated the parameter h_f is fixed equal to 2 based upon observations from discrete element modeling simulations. It is also noted that the force measurements obtained from the discrete element modeling simulations likely underestimate the actual force, since data is sampled at a fixed discrete timestep and is therefore unlikely to capture the maximal force applied to the plate from individual particles (in these simulations, contact is modeled by a stiff normal spring). Similarly, in the case of data from the physical experiments, the mass flow sensor does not explicitly measure force, but instead measures a deflection of a sensor element. Due to the inertia of the sensor element, this deflection is a result of the time history of the applied force, but is not uniquely determined by the instantaneous value of the force. In the discussion below, it is assumed that these factors can be captured by the regression process, especially in the estimated values of k_1 and k_2.

10.1 Verification of the regression process

To verify the regression algorithms, values of ΔP were computed by substituting into Eqn. (8.23) numerical values for the machine geometric parameters V, d, Ω, l, w, d_x, d_y, and a range of ϕ varying between 40° and 90°; the theoretical model parameters μ, e, η, and h_f; and a range of mass flow rates. These parameters are summarized in Table 11.1. The results of the computation are summarized in Table 11.2, and Figures 11.1-11.2 display the results for varying rates of mass flow and varying impact plate angles. The MATLAB M-Files used to generate the momentum values and perform the regression algorithms can be found in Appendices N-Q.

To examine the calibration algorithm, MATLAB's *lsqnonlin* function was applied using the mathematical model and the generated data for an impact plate angle of 70° to estimate the model parameters μ, e, and η. The termination criteria defined for this procedure was the first occurrence of: a change of 10^{-6} or less in the sum of squared residuals, a change of 10^{-6} or less in the desired parameters, or 100 iterations. One hundred combinations of randomly generated initial guesses were used for the model parameters, which were restricted to an upper and lower bound for each parameter: $0 \leq \mu \leq 1$, $0 \leq e \leq 1$, and $750\frac{kg}{m^3} \leq \eta \leq 2000\frac{kg}{m^3}$. Estimates for the parameters were obtained by seeking to minimize the sum of squared residuals of the predicted momentum. The combination of parameters that resulted in

the lowest maximum percent error in the predicted momentum was then selected as optimal. These parameter estimates and the corresponding initial guess from which they resulted are summarized in Table 11.3, and Figure 11.3 displays the relationship between mass flow rate and momentum resulting from the regression process. The maximum percent error in the momentum estimation resulting from this process was 0.009%, and the average percent error was 0.003%.

Given the estimated parameter values from the calibration procedure performed at a 70° impact plate orientation, a bisection algorithm was used to estimate the mass flow rates corresponding to the momentum values for a 70° impact plate orientation from Table 11.2. The results of this process are shown in Table 11.4, which equate to a maximum error of 0.69% and an average error of 0.45%.

In the case that model parameter values need to be updated during real-time operation while simultaneously estimating mass flow rate, the closed-loop estimation routine may be used. This procedure was examined by using the values of ΔP from each row of Table 11.2, and implementing MATLAB's *lsqnonlin* function to simultaneously estimate the mass flow rate and model parameters used to arrive at the corresponding values of ΔP. Values of model parameters returned from the system calibration routine (see Table 11.3) were supplied as initial guesses to the *lsqnonlin* function. The closed loop estimation routine successfully estimates the mass flow rate and model parameters, returning values summarized in Table 11.5. For the set of mass flow rates estimated, this equates to a maximum error of 2.92% in the mass flow rate estimate. These low errors demonstrate the viability of the closed loop estimation routine, which uses known alterations in the impact plate orientation as a means of facilitating self-calibration and accurate mass flow rate estimation. Figure 11.4 displays an example of the initial and final states of the closed loop estimation process.

To further validate the ability of the closed-loop estimation routine to accommodate changing grain conditions, noise was introduced into the model parameters μ, e, and η (see Table 11.6) from the values described in Table 11.1. The closed-loop estimation routine was then applied using as initial guesses the model parameters returned from the system calibration procedure (see Table 11.3). Again, this routine accurately estimates mass flow rates, shown in Table 11.7, given the changes imposed to the model parameters. These estimates equate to a maximum error of 4.59% and an average error of 1.87%.

The three methods of estimation are thus proven to be viable algorithms. The system calibration routine estimates values for model parameters in order to fit the mathematical model with experimentally observed pairs of mass flow rate and momentum transferred to the impact plate. Additionally, the open loop estimation routine estimates the mass flow rate in the case that the internal model parameters were estimated from the system calibration process. Finally, the closed loop estimation routine estimates mass flow rate and internal model parameters to achieve real-time calibration of the sensing system,

and accommodates changes in these model parameters. These procedures encompass the methods of model-based estimation required for a self-calibrating sensor, and it is straightforward to adapt these methodologies for use with experimental data.

10.2 Application through simulation

In the model development described in previous sections, EDEM was used to gain insight into the characteristics of the dynamics of the grain. In addition, here EDEM is used as a simulation-based tool for validation of the ability of the model and the regression algorithm to estimate the mass of grain flowing through the system. To accomplish this, data was collected from simulations involving differing amounts of grain and varying impact plate angles using the EDEM software. The simulations were arranged in a similar fashion as previously described, but for the use of more detailed modeled corn and differences in the operational characteristics of the elevator paddle.

The modeled grain was provided by John Deere and Co., and properties such as grain mass, volume, and shape were strictly constructed, as shown in Table 11.8. To approximate the shape of a corn kernel, several overlapping spheres were arranged as shown in Figure 11.5. Tijskens et al. [24] note that this technique is useful in the approximation of irregular shaped bodies and in minimizing the simulation time required for detection of particle contacts. The geometries used in these simulations have properties defined in Table 11.9, and the interaction properties between all materials used in the simulations are shown in Table 11.10.

Operation of the elevator paddle was modified such that the housing fully constrained the grain for several revolutions of the elevator paddle, during slow acceleration from a resting state, to ensure sufficient settling of the grain until the paddle achieved steady state, constant velocity operation. Upon the paddle reaching this condition, the forward constraint of the housing was removed allowing grain to travel toward the impact plate.

Data collected from these experiments was used as a simulation-based framework for validation of the relationship between the amount of grain in the system and the momentum imparted to the impact plate. The momentum was calculated by resolving the horizontal component of the compressive force on the impact plate due to impacts by the grains. Summing the horizontal compressive force components and multiplying by the simulation timestep facilitated the calculation of the momentum imparted to the impact plate, given by Eqn. (10.1)

$$\Delta P = \sum F_N \sin\phi \Delta t, \tag{10.1}$$

where F_N is the normal force. Figure 11.6 displays the results of simulations as a function of mass flowing in the system, and Figure 11.7 displays the results as a function of impact plate orientation. Simulations were conducted three times for each combination of grain mass and impact plate orientation. A complete numerical summary of the results of the simulations can be found in Appendix R.

For purposes of system calibration, MATLAB's *lsqnonlin* regression function was applied using the mathematical model and the experimental data for an impact plate angle of 70° to estimate the internal model parameters μ, η, k_1, and k_2. The termination criteria defined for this procedure was the first occurrence of: a change of 10^{-6} or less in the sum of squared residuals between predicted and measured momenta, a change of 10^{-6} or less in the desired parameters, or 100 iterations. One hundred combinations of randomly generated initial guesses were used for the model parameters, which were restricted to separate upper and lower bounds for each parameter: $0 \leq \mu \leq 1$, $750\frac{kg}{m^3} \leq \eta \leq 2000\frac{kg}{m^3}$, $0 \leq \ k_1 \leq 5$, and $0 \leq \ k_2 \leq 5$. Estimates for the parameters were obtained by seeking to minimize the sum of squared residuals between the predicted and measured values of the momentum imparted to the impact plate. The optimal combination of desired parameters was selected as the combination corresponding to the lowest sum of squared residuals between predicted and measured momenta. The resulting estimated model parameters and the corresponding initial guess used to arrive at those parameters are summarized in Table 11.11, and the comparison between the true and estimated relationship between grain mass and momentum is displayed in Figure 11.8. The resulting root mean squared residual (RMSR) in the momentum estimation is 0.05 $\frac{kg \cdot m}{s}$, and the normalized root mean squared residual (NRMSR) is 1.18%, here obtained by dividing the RMSR by the maximum observed momentum resulting from the EDEM simulations.

For purposes of validating the open-loop estimation routine, given the estimated parameter values from the regression process performed for an impact plate orientation of 70°, a bisection algorithm was applied to estimate the mass of grain used in the simulations. The process returned grain mass estimates shown in Table 11.12. For this estimation of grain mass in the system, these results equate to a RMSR of 0.03 kg and a NRMSR of 1.66%. When basing the estimation on the averaged measured momentum for each amount of grain mass used in the simulations, the maximum error in the mass estimation is 4.28%.

The closed-loop estimation routine was examined by using the values of ΔP for each case of constant mass flow rate at varying impact plate angles, and applying MATLAB's *lsqnonlin* function with the mathematical model to simultaneously estimate the mass used in the simulations and the model parameters μ, η, k_1, and k_2. Values of model parameters returned from the system calibration routine at an impact plate orientation of 70° were supplied as initial guesses to the *lsqnonlin* function. The closed

loop estimation routine successfully estimated the mass while updating model parameters, returning mass estimates summarized in Table 11.13. For the set of amounts of mass estimated, this equates to a maximum error of 12.65% and an average error of 6.87%. These low errors demonstrate the viability of the closed loop estimation routine, which uses known alterations in the impact plate orientation as a means to facilitate self-calibration and accurate mass flow rate estimation.

It should be recognized that the system calibration routine was performed using data acquired at a single impact plate orientation. It is therefore instructive to examine the predicted relationship between momentum and grain mass for other impact plate orientations, while using the same model parameters returned from the system calibration routine that was performed at a single impact plate orientation. This relationship is displayed in Figures 11.9-11.11, which compares the model predictions to data obtained from the EDEM simulations. This analysis shows that the model predictions match well with the experimental data, further supporting the claim that the mathematical model captures the dependence on the impact plate angle. This supports the method of inducing alterations in the impact plate orientation as a viable concept for achieving self-calibration.

10.3 Application through small-scale experimentation

Experimental data was collected in small scale experiments utilizing a lab-sized testbench (see Figure 11.12), which was designed and developed to investigate the relationship between momentum imparted to the impact plate, mass flowing through the system, and geometric parameters of the system. This experimental apparatus replicates the key operational characteristics of the production level system, while simplifying the operation by replacing the elevator paddle system with a single paddle powered by a DC motor. Key components and dimensions of the production level system are still maintained, such as the mass flow sensor, flat impact plate, paddle dimensions, and paddle rotational speeds. Additionally, this system allows for modifications to the orientation of the impact plate in order to facilitate validation of the closed-loop estimation routine.

Components

A variety of components were used in construction of this testbench, which consist of production-level components found on John Deere combines, as well as modifications and simplifications to the production-level system. A list of primary components is provided below:

- Mass flow sensor (Generation 1; Part #: AH163359)

- Flat impact plate assembly (Part #: AH228541)
- Plastic pellets - 6mm diameter, 0.12 g (used to simulate grain)
- Brushless DC motor with speed controller
- National Instruments Data Acquisition System
- Microcontroller board
- Photo interrupter sensor
- 12 volt DC Power Supply
- Solenoids

System description and operation

This unique test system was developed in order to facilitate experiments with a limited amount of setup and operation time. These tests controlled extraneous test factors that may be present on the production-level test stand, such as grain leakage around the elevator paddles, aging of the elevator paddles, and inconsistencies in material properties. Additionally, the test system allowed for changes in the impact plate orientation to facilitate data collection for the purpose of verifying the closed-loop estimation method.

The primary conceptual modifications this testbench uses in comparison to the production level system is the substitution of the series of elevator paddles with a single paddle powered by a brushless DC motor, and the use of experimental plastic pellets in the place of grain. The paddle, made of sheet steel, was fixed to a shaft connected to the motor such that the paddle rotated continuously inside a circular housing. The housing was constructed to fully contain the experimental pellets during startup of the motor to enable the motor to achieve steady-state operation at the desired speed.

An electro-mechanical tripping mechanism was implemented to enable a portion of the housing to constrain the grains during startup of the motor, and subsequently to allow the grains to be released from the forward constraint of the housing following this startup time, enabling them to travel toward the impact plate. This tripping mechanism consisted of solenoids, powered by a 12 V DC power supply, acting as release pins on a spring-loaded, hinged portion of the housing. The solenoids held the housing closed during startup operation of the motor, and when retracted, allowed for the housing to rapidly swing open to allow grain to travel freely toward the impact plate.

The action of the solenoids was controlled via a microcontroller. A photointerrupter was connected to the microcontroller and was used to sense the number of revolutions of the drive-shaft of the motor. After a desired number of motor revolutions was achieved, the microcontroller activated the solenoids, which then retracted, and the hinged portion of the housing was released to allow grain to be propelled toward the impact plate.

The method of mounting the impact plate to the mass flow sensor was modified from the production level arrangement to enable variable orientation of the impact plate. The bracket used to interface the impact plate with the mass flow sensor on John Deere Combines was replaced by a simple mounting structure that allowed for angular changes in the impact plate orientation via rotation about the top axis of the impact plate. The mass flow sensor was mounted rigidly to this mounting structure, and was wired directly to a National Instruments Data Acquisition System sampling at 100 kHz.

Data collection and analysis

Ten experiments were conducted for each amount of bulk mass of experimental pellets at each impact plate angle. The signal from the mass flow sensor was sampled via the National Instruments data acquisition system and output to LabVIEW Signal Express for analysis. The raw signal was then shifted such that the zero-force DC voltage from the sensor was normalized to 0 volts. This was accomplished by averaging the no-load signal and shifting the raw signal by the result of this average. The shifted signal was then summed to obtain a value of the sensor response that is proportional to the momentum imparted to the impact plate (assuming that the deflection of the plate is proportional to the instantaneous force applied to the plate). Figures 11.13-11.14 display the averages of the sensor output for varying amounts of grain mass and varying impact plate angles, and a complete numerical summary of the experiments can be found in Appendix S.

To verify the ability of the model to accurately describe the experimental data, MATLAB'S *lsqnonlin* regression function was applied to estimate the model parameters μ, η, k_1, and k_2. The termination criteria defined for this procedure was the first occurrence of: a change of 10^{-6} or less in the sum of squared residuals between the predicted and measured sensor response, a change of 10^{-6} or less in the desired parameters, or 100 iterations. One hundred combinations of randomly generated initial guesses were used for the model parameters, which were restricted to an upper and lower bound for each parameter: $0 \leq \mu \leq 1$, $750\frac{kg}{m^3} \leq \eta \leq 2000\frac{kg}{m^3}$, $0 \leq k_1 \leq 500$, and $0 \leq k_2 \leq 500$. The somewhat large ranges chosen for k_1 and k_2 were justified by the reference to the summed sensor signal output during the regression process, without performing a direct conversion of the sensor signal to momentum. Estimates

for the parameters were obtained by seeking to minimize the sum of squared residuals of the predicted sensor response. The combination of parameters that minimized the sum of squared residuals between the predicted and measured summed sensor signal was selected as optimal. This results in values for μ, η, k_1, and k_2 shown in Table 11.14, and equates to a RMSR of 13.48 V and a NRMSR (normalized by dividing the RMSR by the maximum observed summed sensor signal) of 5.80% in the momentum estimation.

Knowledge of model parameters coupled with measured sensor values from the mass flow sensor allow for the grain mass to be estimated via the open loop estimation routine, which uses a bisection algorithm applied to the mathematical model. Application of the open loop estimation routine with inputs of model parameters solved from the system calibration routine performed for an impact plate orientation of 70°, and measured voltage values for an impact plate orientation of 70°, returns mass estimates shown in Table 11.15. This equates to a RMSR of 0.022 kg and a NRMSR (normalized by dividing the RMSR by the maximum bulk mass used in the experiments) of 7.46%. When basing the estimation on the average voltage value for each amount of mass, the model estimates the mass with a maximum error of 7.07% and an average error of 2.53%. An example of the relationship between the sensor output and grain mass resulting from this procedure is shown in Figure 11.15. Clearly, the calibrated model is able to capture the general trend of the data, including the non-linear regions at relatively higher flow rates. In particular, the average of the squared residuals between the predicted and measured sensor signal at each mass flow rate are of the same order of magnitude as the variance of the experimental data about its mean. Specifically, the ratio between the root-mean-squared residual for each mass flow rate and the corresponding standard deviation results in values of 0.96, 1.13, 0.95, 0.96, 0.96, and 0.96, corresponding to the paddle masses of 50 g, 100 g, 150 g, 200 g, 250 g, and 300 g, respectively. Similarly, using the calibrated parameters to estimate the mass corresponding to the average summed voltage across the 10 experiments associated with each known paddle mass results in relative differences of 2.6%, 7.1%, 0.6%, 2.9%, 1.0%, and 1.0%, respectively. This close agreement suggests that the model can be calibrated to achieve close agreement with the true relationship between paddle mass and the sensor signal.

To assess the ability of the model and the regression algorithm to successfully perform self-calibration using the closed loop estimation routine, MATLAB'S *lsqnonlin* regression function was applied to estimate the mass in the system and the model parameters μ, η, k_1, and k_2. The model parameters returned from the system calibration routine performed at an impact plate orientation of 70° were used as initial guesses. The resulting estimated mass, the corresponding initial guess used to arrive at the estimated mass, and the error associated with the estimate is summarized in Table 11.16. The low errors associated with the estimation of the mass in the system, while simultaneously estimating model parameters, indicate the

success of this self-calibration procedure.

As with previous methods of experimentation and simulation, it should be recognized that the system calibration routine was performed using a single impact plate orientation. It is therefore instructive to examine the predicted relationship between the sensor signal and grain mass for other impact plate orientations, while using the same model parameters returned from the system calibration routine that was performed at a single impact plate orientation. This relationship is displayed in Figures 11.16- 11.18, which compares the model predictions to data obtained from the experiments. This analysis shows that the model predictions capture the trends of the experimental data, and that the mathematical model approximately captures the dependence on the impact plate angle. This further supports the method of inducing alterations in the impact plate orientation as a viable concept to achieve self-calibration of the sensor system.

10.4 Application through large-scale experimentation

To further facilitate the validation of the model, data was obtained from experiments conducted using a replica of a combine clean grain elevator system at the Combine Yield Monitor Test Facility at the University of Kentucky [8, 9]. This facility is equipped with production level machine components commonly used in mass flow sensing on combines, such as the GreenStar yield monitor and clean grain elevator. The GreenStar sensor has demonstrated accuracy of less than 1% relative error at calibration flow rates and approximately 3% relative error for extreme flowrates as compared to the measured mass [8, 9].

Since it is hypothesized that the parameters of the system depend upon the moisture content of the grain, data from experimental system tests at various moisture contents is desirable. Data was acquired from tests with corn at moisture levels of 14%, 21%, and 26%, with seven mass flow rates at each moisture level, varying between 2 kg/s and 20 kg/s. These tests resulted in data for various rates of mass flow, elevator speeds, and measured impact forces at each moisture level, which is summarized in Figures 11.19-11.21. The mass flow rates were calculated by dividing the scale weight of the grain after the experiment by the time taken to complete the experiment. The resultant forces were calculated by averaging the output of the GreenStar yield monitor, which samples the voltage output of the mass flow sensor at a frequency of 1kHz, over one second intervals corresponding to the periods of stable elevator speeds. Multiplying this voltage output by the characteristic conversion for the mass flow sensor of $0.4448\frac{N}{mV}$ facilitates the conversion to impact force.

To verify the ability of the model to accurately describe the experimental data, MATLAB's *lsqnonlin*

regression function was applied to a subset of the experimental data in order to estimate the model parameters μ, η, k_1, and k_2. The subset of data selected for calibration corresponded to the measured forces and mass flow rates at the operating conditions of the lowest and highest elevator speeds. The number of data points in these subsets were: 77 data points for experiments at 14% moisture, 60 data points for corn at 21% moisture, and 65 data points for corn at 26% moisture. The termination criteria defined for this procedure was the first occurrence of: a change of 10^{-6} or less in the sum of squared residuals between the predicted and measured force, a change of 10^{-6} or less in the desired parameters, or 100 iterations. One hundred combinations of randomly generated initial guesses were used for the model parameters, which were restricted to an upper and lower bound for each parameter: $0 \leq \mu \leq 1$, $750\frac{kg}{m^3} \leq \eta \leq 2000\frac{kg}{m^3}$, $0 \leq k_1 \leq 5$, and $0 \leq k_2 \leq 5$. Estimates for the parameters were obtained by seeking to minimize the sum of squared residuals between the predicted and measured force. The optimal combination of parameters was selected as those corresponding to the lowest sum of squared residuals between the predicted and measured force.

The resulting estimated model parameters and the corresponding initial guess used to arrive at those parameters are summarized in Table 11.17. Using these estimated parameters, calibration curves for each moisture level were defined and are shown in Figure 11.22. These different calibration curves, corresponding to an impact plate orientation of 70°, demonstrate the necessity of a model that is adaptable to varying moisture conditions. Additionally, using the model parameters returned from the system calibration procedure performed at an impact plate orientation of 68° for each moisture level, calibration curves can be graphed for each mass flow rate, as shown in Figures 11.23- 11.25. More specifically, the agreement of these curves with the experimental data is shown in Figures 11.26-11.28 for several distinct mass flow rates. From these calibration curves, it can be seen that the impact force is relatively insensitive to changes in the elevator speed, and is significantly more dependent on changes in mass flow rate. The residuals resulting from the regression process are summarized in Table 11.18, and were calculated using the complete set experimental data. The low residuals resulting from this regression process indicate that this procedure resulted in good agreement between predicted forces and measured forces.

As previously described, the mass flow rate estimation is accomplished using the resultant model parameter values returned from the system calibration routine performed at an impact plate orientation of 68° and application of a bisection algorithm to the mathematical model, using the open loop estimation routine. Application of this procedure returns mass flow rate estimates with residuals described in Table 11.19, which were calculated using a random sample of 100 data points from the complete set of experimental data. The low residuals associated with the estimation of the mass flow rate indicate the

success of this estimation procedure.

To assess the ability of the model and the regression algorithm to successfully perform self-calibration using the closed loop estimation routine, MATLAB'S *lsqnonlin* regression function was applied to the experimental data to simultaneously estimate the mass flow rate and the model parameters μ, η, k_1, and k_2. Initial guesses for model parameters corresponded to those returned from the system calibration routine for an impact plate orientation of 68°. To collect a set of data encompassing a single mass flow rate and varying impact plate orientations, the results of several distinct experiments using the same mass flow rate, but each using differing impact plate angles, were combined. The values for model parameters returned from the system calibration routine were used as initial guesses, and bounds of $\pm 10\%$ were placed on their values during the calibration process. The resulting estimated mass flow rate, the corresponding initial guess used to arrive at the estimated mass flow rate, and the error associated with the estimate is summarized in Table 11.20. Using the model parameters from the calibration procedure performed for an impact plate orientation of 68°, the comparison between the complete set of experimental data at each impact plate angle and the estimated relationship between impact force and mass flow rate at each impact plate angle is shown in Figures 11.29-11.33 for 14% moisture, in Figures 11.34-11.38 for 21% moisture, and in Figures 11.39-11.43 for 26% moisture. These figures show the degree to which the model captures the trend of the experimental data at each impact plate angle. The model captures the trends for impact plate angles of 68°, 80°, 90°, but underestimates the impact force for larger flow rates at angles of 40° and 55°.

Some of the deviations resulting from the closed-loop estimation procedure are rather large. This may be attributable to the spread of the experimental data and to the method with which the experiments were conducted. In conducting the experiments, it was not feasible to achieve the exact same mass flow rate for each of the experiments at each impact plate orientation. Rather, the mass flow rates were only approximately the same, and for the purposes of this effort, the average of these mass flow rates was used. However, the closed-loop estimation routine assumes that the mass flow rate is constant over the course of the data collection. Additionally, the collected data exhibited a significant spread in the measured force, especially over the range of elevator speeds encountered during the experiments, which where not necessarily maintained at a specific operating value. In fact, variations in elevator speed were observed between 300 RPM and 500 RPM. Furthermore, more data points exist for some operating conditions than others. Thus, at certain operating conditions that exhibit only a small collection of data points, the force values at these conditions may not necessarily be representative of the true or average force for that condition, buy may instead contain only a small number of data points that would be considered outliers.

Chapter 11

Figures and Tables

Table 11.1: Summary of model parameters used in verification of the regression algorithm.

$V\ \left(\frac{m}{s}\right)$	$d\ (m)$	$\Omega\ (rpm)$	$l\ (m)$	$w\ (m)$	$d_x\ (m)$	$d_y\ (m)$	μ	e	$\eta\ \left(\frac{kg}{m^3}\right)$	h_f
2.27	0.18635	400	0.16	0.21	0.4645	0.0173	0.2	0.6	1,100	2

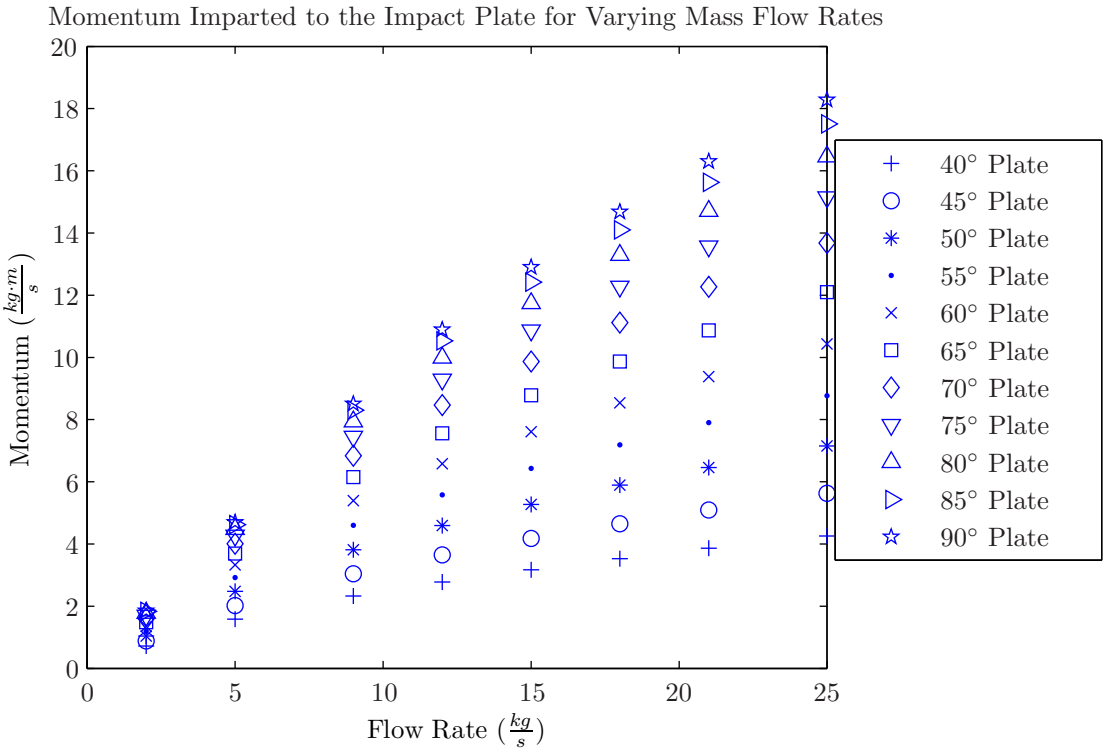

Figure 11.1: Generated data displaying predicted momentum as a function of mass flow rate for varying impact plate angles.

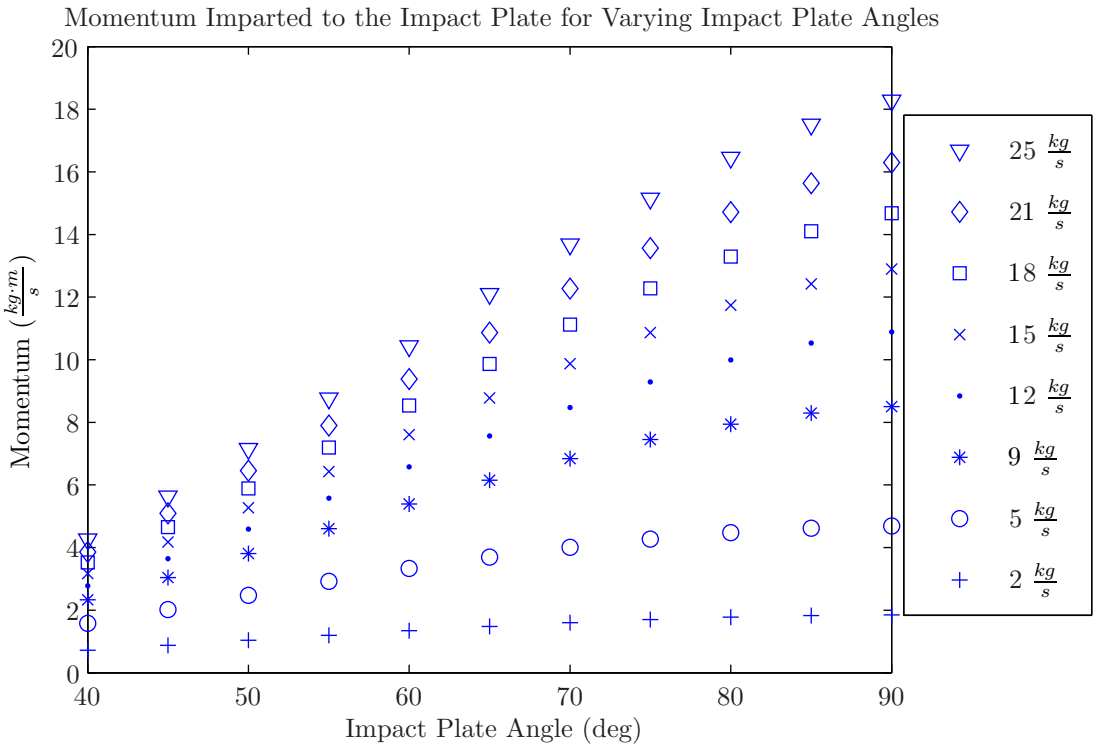

Figure 11.2: Generated data displaying predicted momentum as a function of impact plate angle for varying mass flow rates.

Table 11.2: Summary of momentum ($\frac{kg \cdot m}{s}$) imparted to the impact plate for varying rates of mass flow and varying impact plate angles, generated using the mathematical model.

		Impact Plate Angle										
		40°	45°	50°	55°	60°	65°	70°	75°	80°	85°	90°
Flow Rate	2 $\frac{kg}{s}$	0.72	0.88	1.04	1.20	1.35	1.48	1.60	1.70	1.78	1.83	1.85
	5 $\frac{kg}{s}$	1.58	2.02	2.48	2.92	3.33	3.69	4.01	4.27	4.48	4.62	4.69
	9 $\frac{kg}{s}$	2.33	3.04	3.81	4.60	5.39	6.15	6.84	7.45	7.94	8.30	8.50
	12 $\frac{kg}{s}$	2.78	3.65	4.59	5.58	6.58	7.56	8.47	9.29	9.99	10.53	10.89
	15 $\frac{kg}{s}$	3.17	4.18	5.27	6.43	7.61	8.78	9.87	10.87	11.74	12.42	12.90
	18 $\frac{kg}{s}$	3.53	4.65	5.89	7.19	8.54	9.87	11.12	12.28	13.29	14.10	14.68
	21 $\frac{kg}{s}$	3.86	5.09	6.46	7.90	9.38	10.87	12.27	13.57	14.71	15.63	16.30
	25 $\frac{kg}{s}$	4.26	5.63	7.15	8.77	10.43	12.10	13.68	15.15	16.46	17.51	18.28

Table 11.3: Summary of estimated model parameters returned from the system calibration regression algorithm performed for an impact plate orientation of 70°.

	μ	e	$\eta\left(\frac{kg}{m^3}\right)$
Initial guess	0.18	0.80	917
Final result	0.19	0.60	1,084

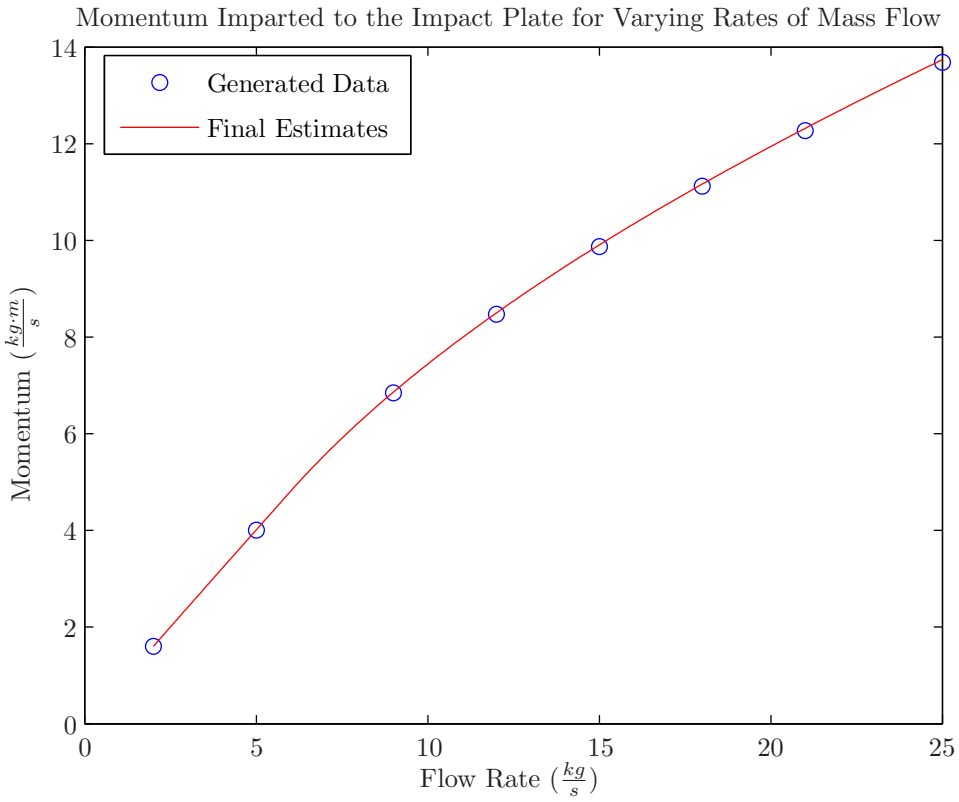

Figure 11.3: Momentum imparted to the impact plate for varying rates of mass flow. Discrete points are data generated using the mathematical model, and the solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 70°.

Table 11.4: Summary of mass flow rate estimates resulting from the open-loop estimation process, performed using estimates for model parameters obtained from the system calibration procedure for an impact plate orientation of 70°.

True Mass (kg)	Estimated Mass (kg)
2	1.996
5	4.997
9	8.960
12	11.941
15	14.918
18	17.889
21	20.869
25	24.828

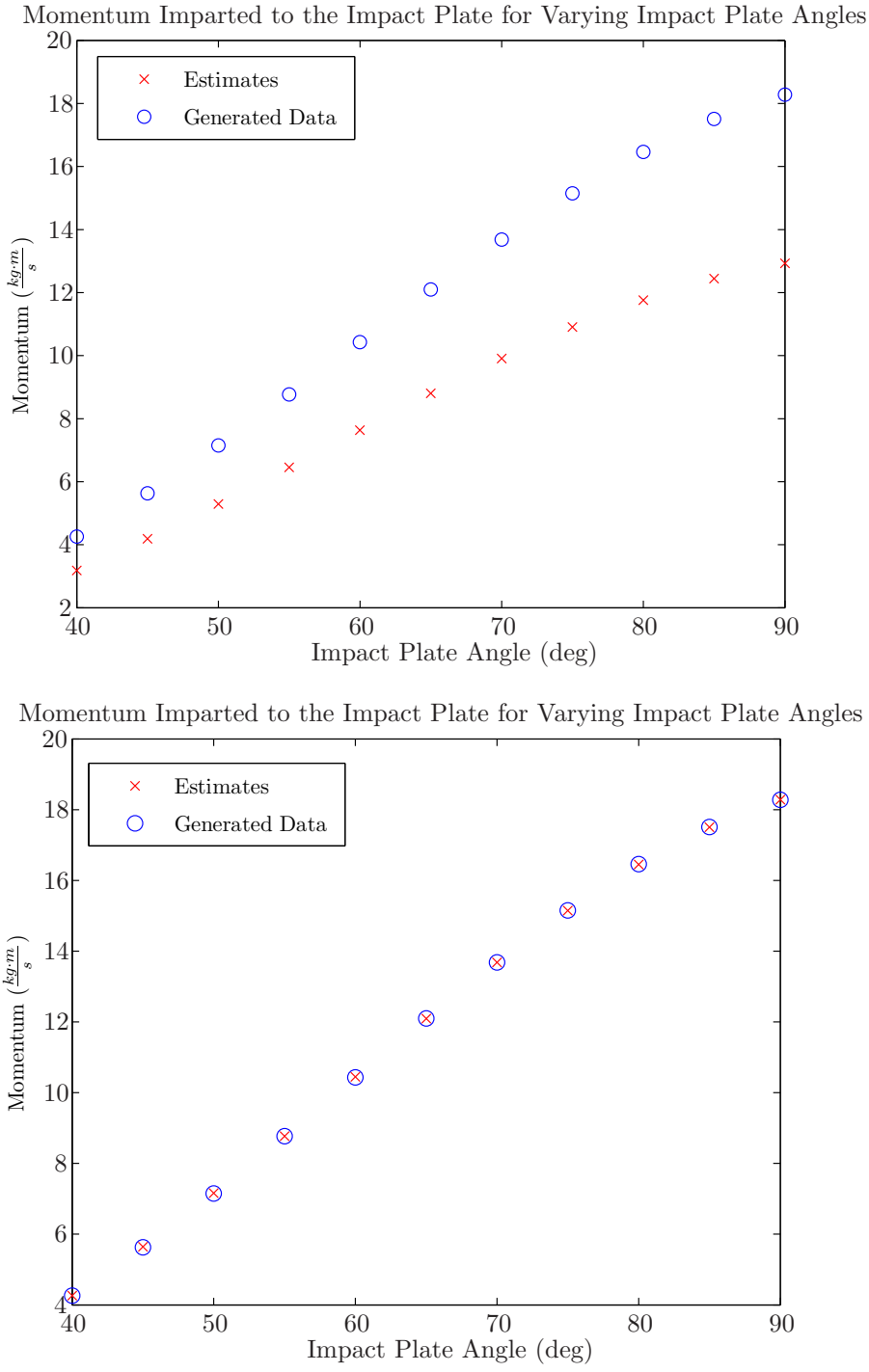

Figure 11.4: Momentum imparted to the impact plate for varying rates of mass flow. Discrete points are data generated using the mathematical model, and the solid curve is the predicted dependence using initial estimates (top) and final estimates (bottom) for model parameters corresponding to the system calibration routine performed for an impact plate orientation of 70°.

Table 11.5: Summary of estimated mass flow rates resulting from the closed-loop estimation process, using model parameters returned from the system calibration process performed for an impact plate orientation of 70°.

True $\dot{m}\left(\frac{kg}{s}\right)$	Estimated $\dot{m}\left(\frac{kg}{s}\right)$
2	2.06
5	5.10
9	8.98
12	12.07
15	15.00
18	17.98
21	20.87
25	24.46

Table 11.6: Updated model parameters used to simulate changes in grain properties.

μ	e	$\eta\left(\frac{kg}{m^3}\right)$
0.1952	0.6360	1,133

Table 11.7: Summary of estimated mass flow rates resulting from the closed-loop estimation process using updated model parameters.

True $\dot{m}\left(\frac{kg}{s}\right)$	Estimated $\dot{m}\left(\frac{kg}{s}\right)$
2	2.09
5	5.13
9	9.20
12	12.14
15	15.30
18	18.23
21	21.21
25	25.04

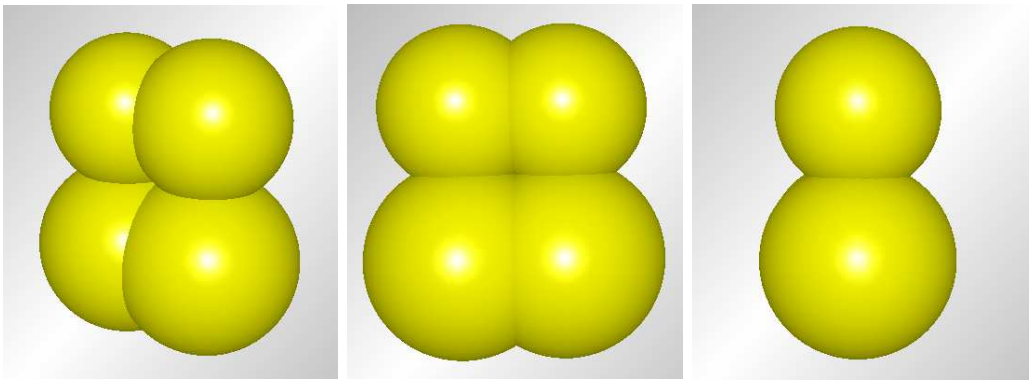

Figure 11.5: Corn particle used in discrete element modeling simulations displayed in the view of an isometric projection (left), a frontal projection (center), and a side projection (right).

Table 11.8: Grain properties used in discrete element modeling simulations.

Property	Value
Mass	4.000×10^{-4} kg
Volume	3.001×10^{-7} m^3
Small Sphere Radius	2.500×10^{-3} m
Large Sphere Radius	3.000×10^{-3} m
Poisson's Ratio	4.000×10^{-1}
Shear Modulus	6.300×10^{6} Pa

Table 11.9: Geometry properties used in discrete element modeling simulations.

Property	Value
Density	940 $\frac{kg}{m^3}$
Poisson's Ratio	4.600×10^{-1}
Shear Modulus	5.309×10^{8} Pa

Table 11.10: Material interaction properties between corn and all materials used in discrete element modeling simulations.

Property	Value
Coefficient of static friction	0.3
Coefficient of rolling friction	0.01
Coefficient of restitution	0.75

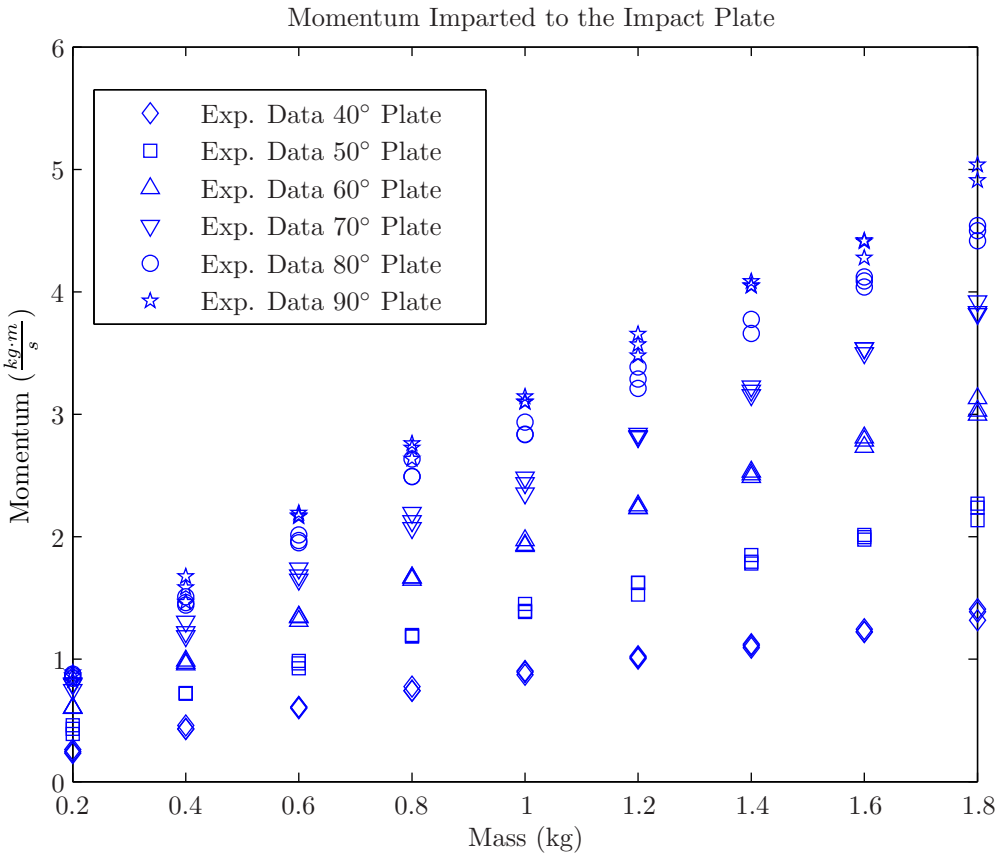

Figure 11.6: Data collected using discrete element modeling software displaying momentum imparted to the impact plate in the horizontal direction as a function of mass, for varying impact plate orientations.

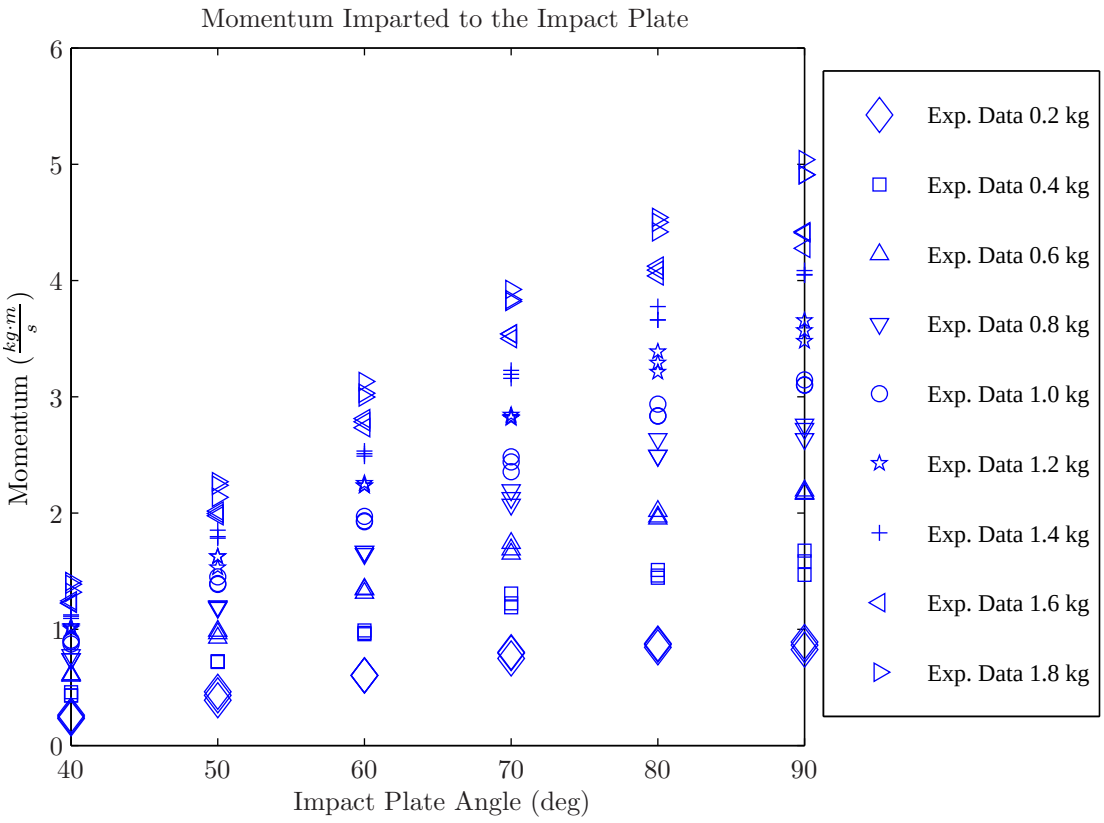

Figure 11.7: Data collected using discrete element modeling software displaying momentum imparted to the impact plate in the horizontal direction as a function of impact plate angle for varying amounts of mass.

Table 11.11: Summary of estimated model parameters from the regression algorithm performed for an impact plate orientation of 70°.

	μ	$\eta\left(\frac{kg}{m^3}\right)$	k_1	k_2
Initial guess	0.42	1,979	0.60	1.40
Final result	0.60	1,610	0.80	0.83

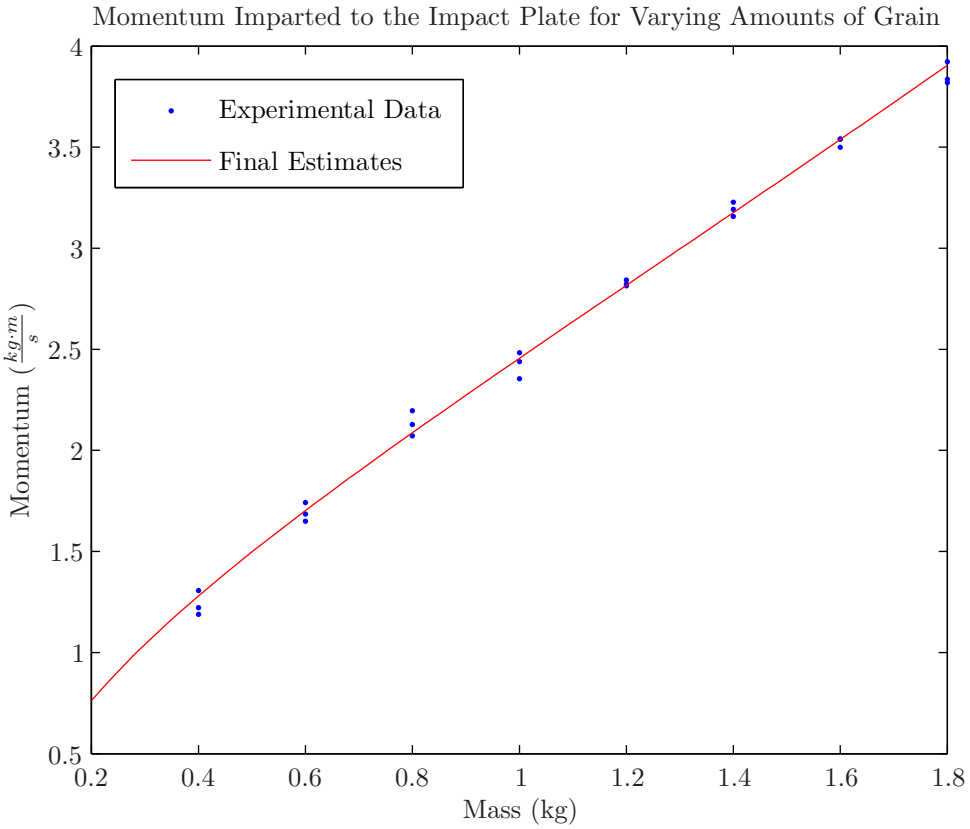

Figure 11.8: Momentum imparted to the impact plate for varying rates of mass flow. Discrete points represent data obtained from simulations using the discrete element modeling software, and the solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 70°.

Table 11.12: Summary of estimated grain mass returned from the open-loop estimation process using model parameters obtained from the system calibration routine performed for an impact plate orientation of 70°.

True Mass (kg)	Range of Estimated Mass (kg)
0.2	0.197 - 0.217
0.4	0.372 - 0.419
0.6	0.582 - 0.629
0.8	0.804 - 0.874
1.0	0.968 - 1.038
1.2	1.225 - 1.225
1.4	1.412 - 1.458
1.6	1.598 - 1.645
1.8	1.785 - 1.832

Table 11.13: Results of the closed loop estimation procedure for each amount of grain mass, using model parameters from the system calibration routine performed for an impact plate orientation of 70°. The percent error is calculated by comparing the estimated paddle mass with the actual paddle mass.

Actual paddle mass (kg)	Initial guess (kg)	Estimated paddle mass (kg)	% error
0.2	1.0	0.225	12.65
0.4	1.0	0.443	10.85
0.6	1.0	0.632	5.25
0.8	1.0	0.837	4.59
1.0	1.0	1.004	0.37
1.2	1.0	1.150	4.17
1.4	1.0	1.316	6.02
1.6	1.0	1.466	8.41
1.8	1.0	1.629	9.50

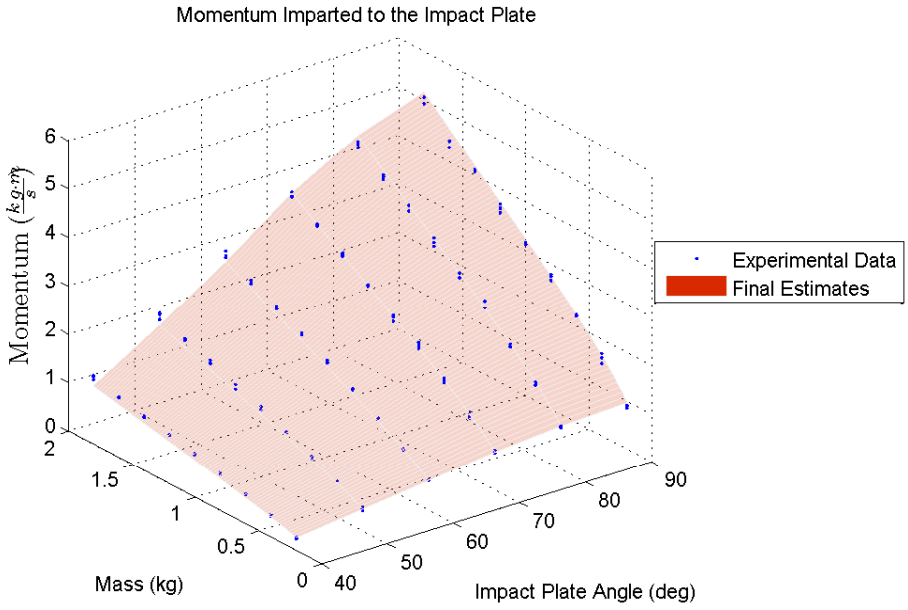

Figure 11.9: Comparison of the observed and predicted relationship between momentum imparted to the impact plate, mass, and impact plate orientation. Discrete points represent data obtained from discrete element modeling simulations, and the red surface represents predictions using model parameters obtained from the system calibration routine performed for an impact plate orientation of 70°.

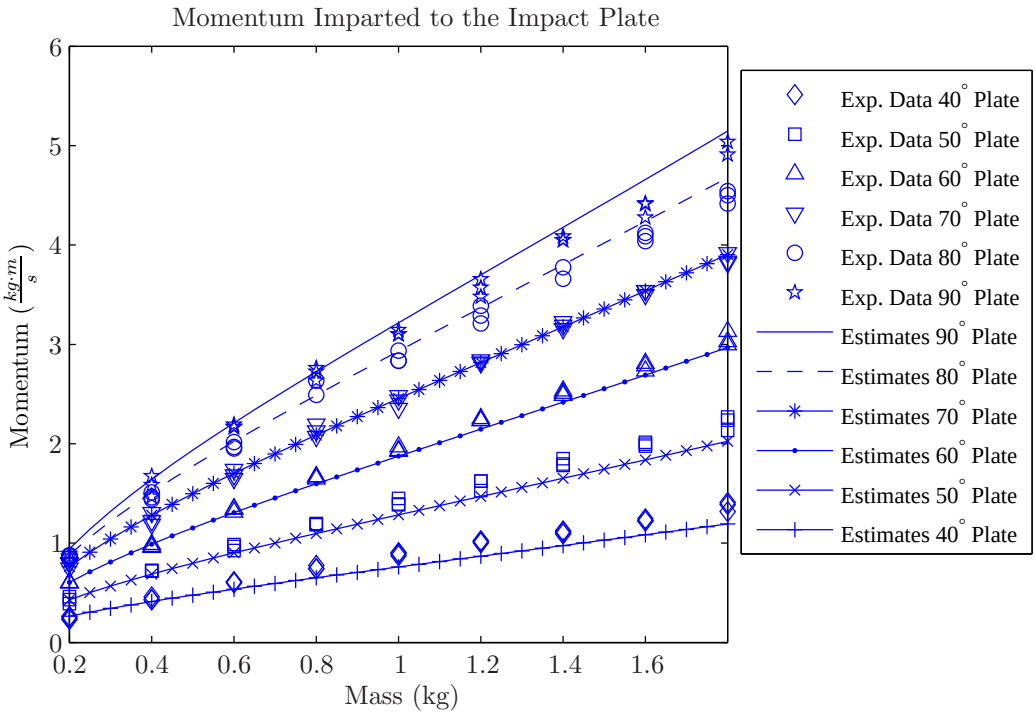

Figure 11.10: Comparison of the observed and predicted relationship between momentum and mass for varying impact plate orientations. The predicted relationships were obtained using the mathematical model and model parameters returned from the system calibration routine performed for an impact plate orientation of 70°.

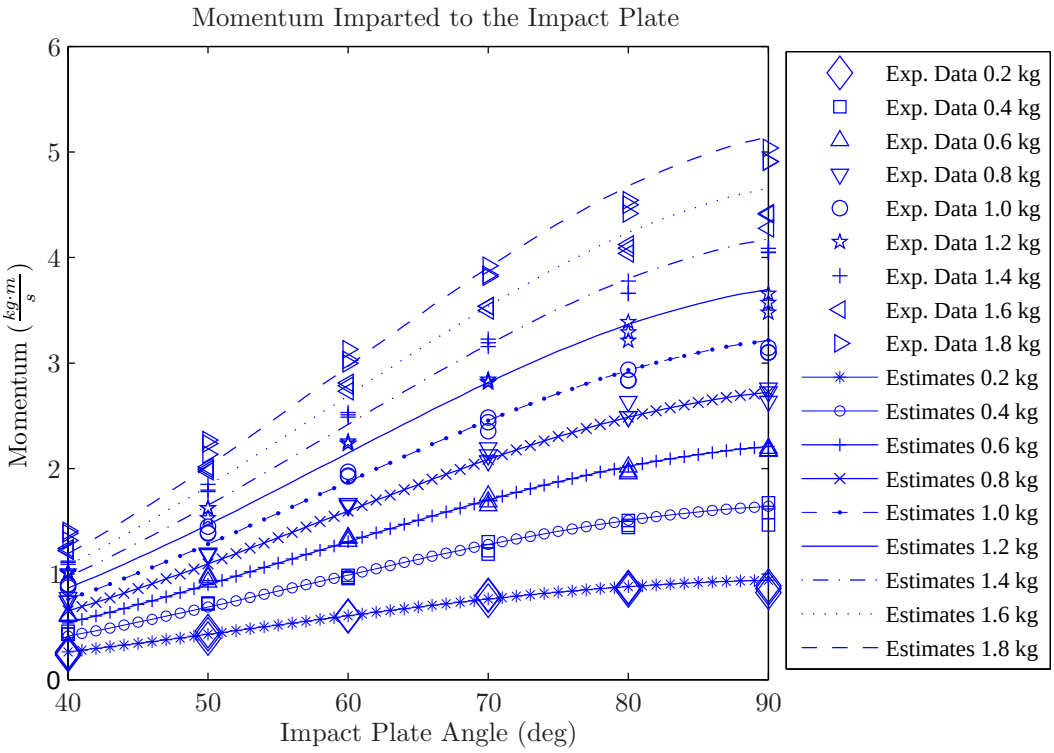

Figure 11.11: Comparison of the observed and predicted relationship between momentum and impact plate orientation for varying amounts of mass. The predicted relationships were obtained using the mathematical model and model parameters returned from the system calibration routine performed for an impact plate orientation of 70°.

Figure 11.12: Photographs of a testbench designed and developed for small-scale laboratory experiments.

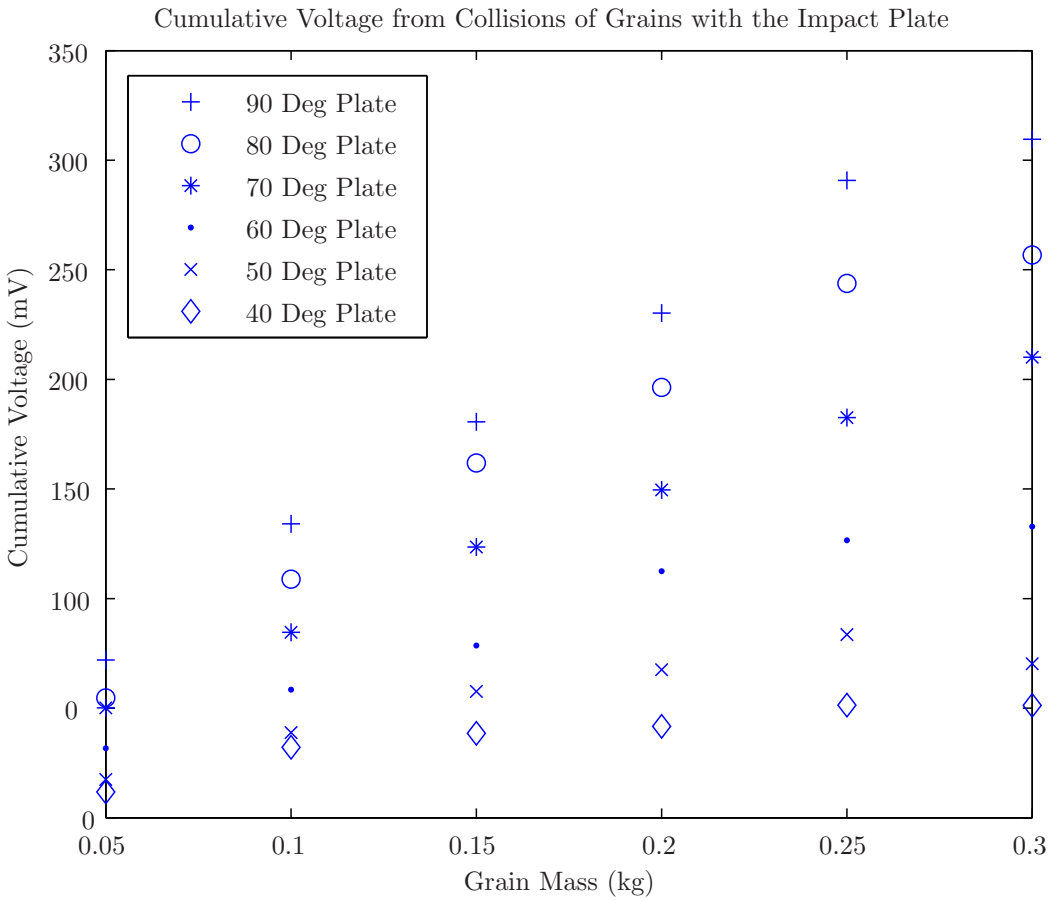

Figure 11.13: Cumulative sensor response as a function of mass for varying impact plate orientations. Discrete points are averages of data collected for each condition.

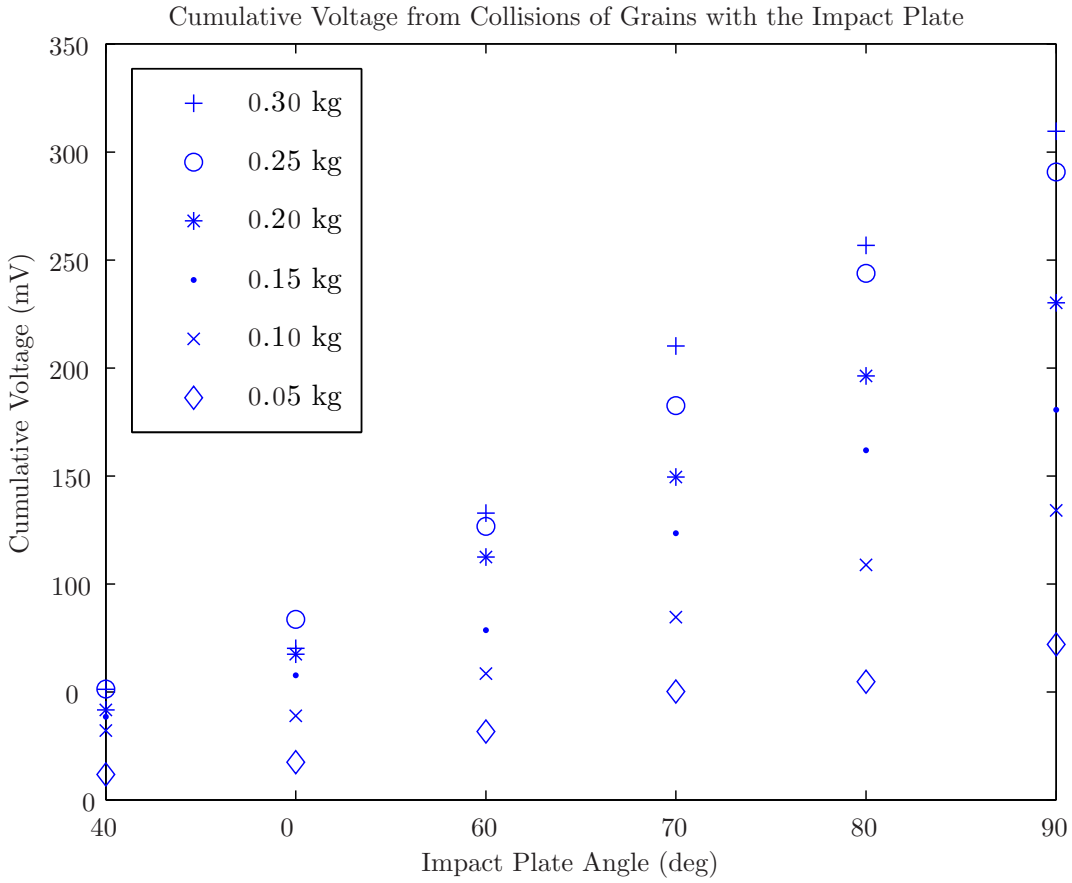

Figure 11.14: Cumulative sensor response as a function of impact plate angle for varying amounts of mass. Discrete points are averages of data collected for each condition.

Table 11.14: Summary of estimated model parameters returned from the system calibration process performed at an impact plate orienation of 70°.

	μ	$\eta\left(\frac{kg}{m^3}\right)$	k_1	k_2
Initial guess	0.08	883	177	385
Final result	0.43	836	394	202

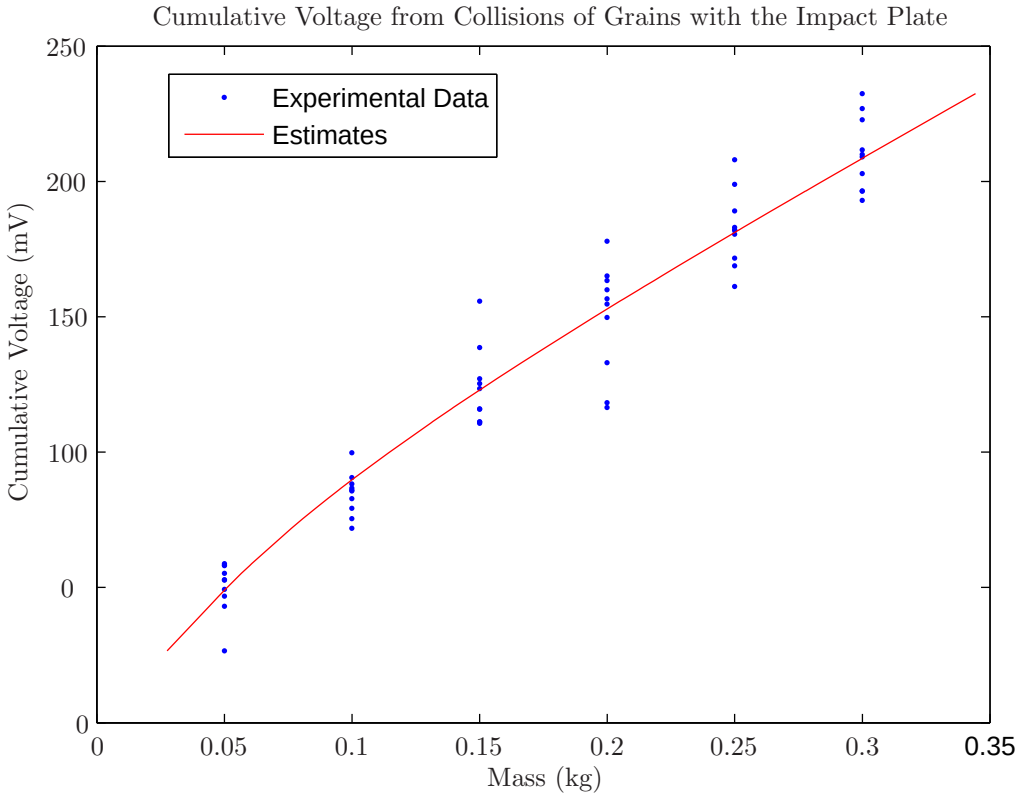

Figure 11.15: Cumulative sensor output for varying rates of mass flow for small-scale laboratory experiments. Here, the red curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 70°.

Table 11.15: Summary of estimated grain mass returned from the mass estimation process using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 70°.

True Mass (kg)	Range of Estimated Mass (kg)
0.05	0.028 - 0.061
0.1	0.076 - 0.114
0.15	0.131 - 0.205
0.2	0.140 - 0.244
0.25	0.215 - 0.299
0.3	0.272 - 0.344

Table 11.16: Summary of estimated mass flow rates returned from the closed loop estimation algorithm. The percent error is calculated by comparing the estimated grain mass with the true grain mass.

True Grain Mass (kg)	Initial Guess (kg)	Estimated Grain Mass (kg)	% Error
0.05	0.10	0.051	2.329%
0.10	0.10	0.103	2.708%
0.15	0.10	0.156	3.084%
0.20	0.10	0.218	8.808%
0.25	0.10	0.276	10.222%
0.30	0.10	0.294	1.914%

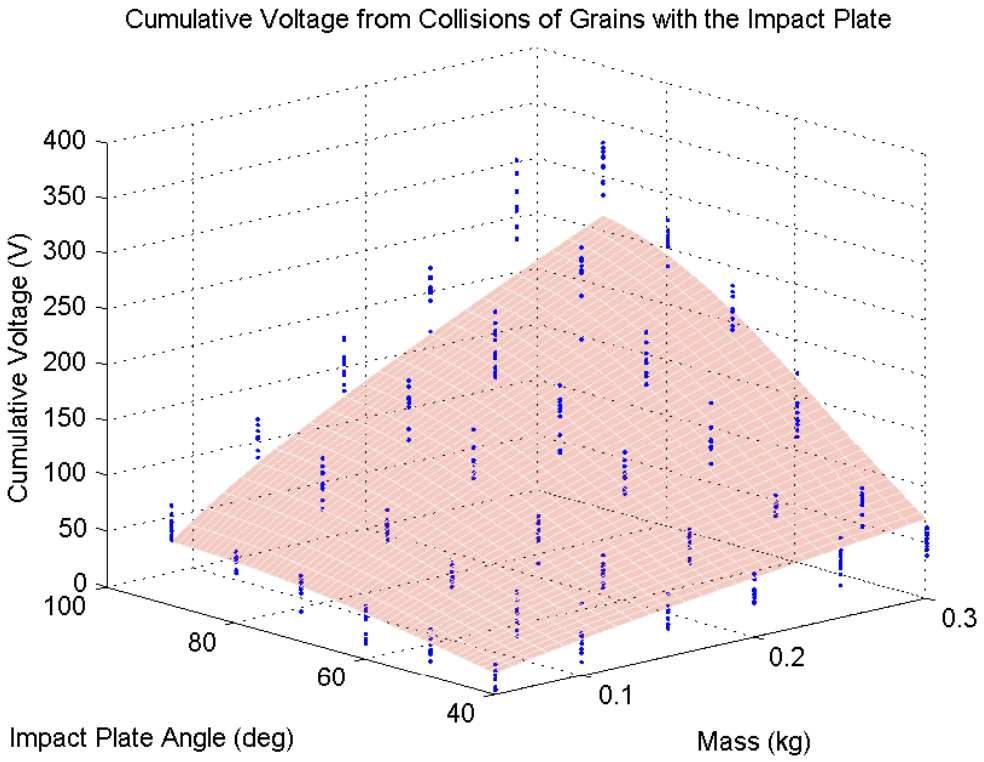

Figure 11.16: Comparison of the observed and predicted relationship between cumulative sensor response, mass, and impact plate orientation. Discrete points represent experimental data, and the red surface is the predicted sensor response from the mathematical model using model parameters obtained from the system calibration routine performed for an impact plate orientation of 70°.

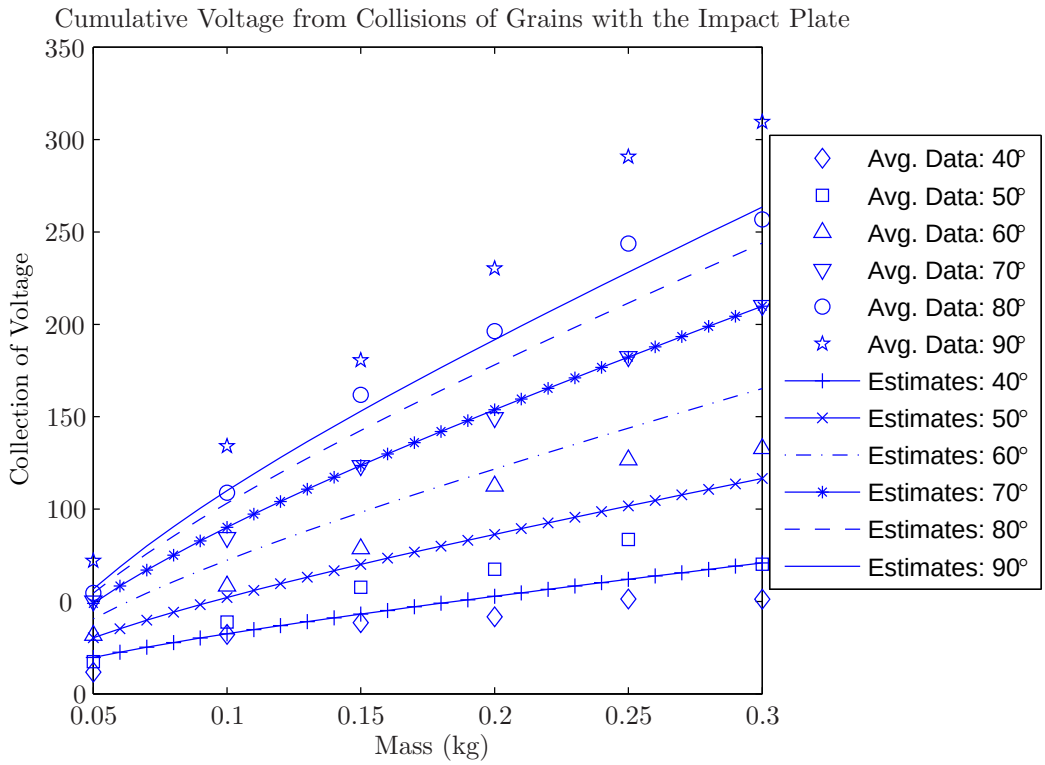

Figure 11.17: Comparison of the observed and predicted relationship between cumulative sensor response and mass for varying impact plate orientations. Discrete points represent the average of the data for each condition. The predicted relationships were obtained using the mathematical model and model parameters obtained from the system calibration routine performed for an impact plate orientation of 70°.

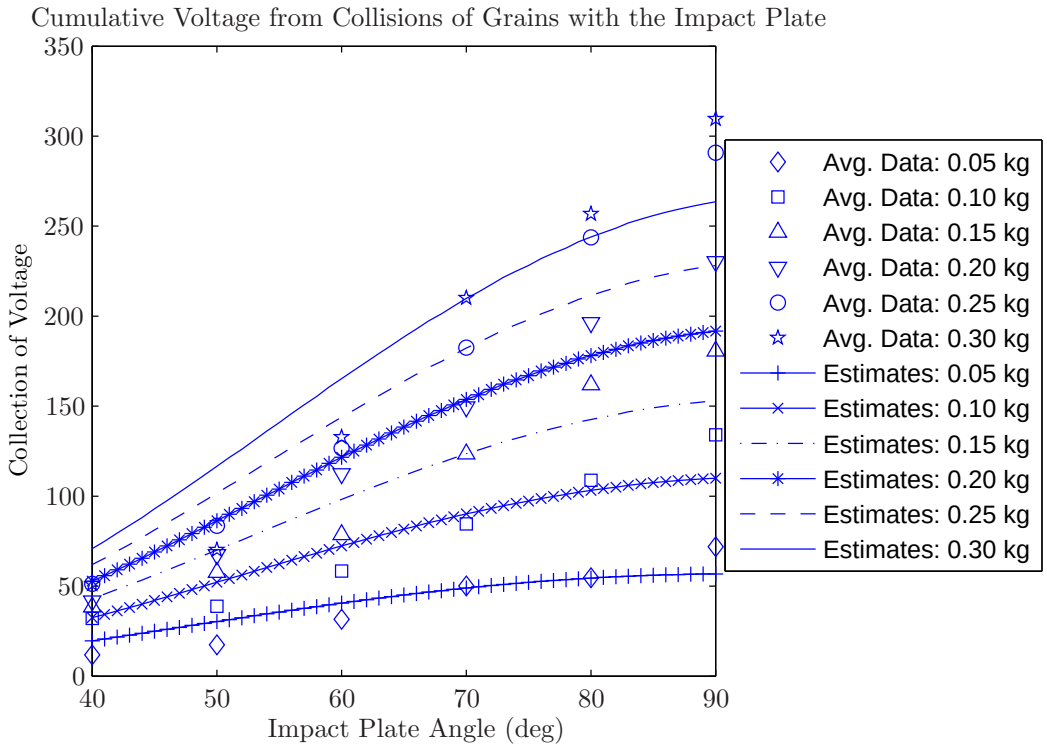

Figure 11.18: Comparison of the observed and predicted relationship between the cumulative sensor response and impact plate orientation for varying amounts of mass. Discrete points represent the average of the data for each condition. The predicted relationships were obtained using the mathematical model and model parameters obtained from the system calibration routine performed for an impact plate orientation of 70°.

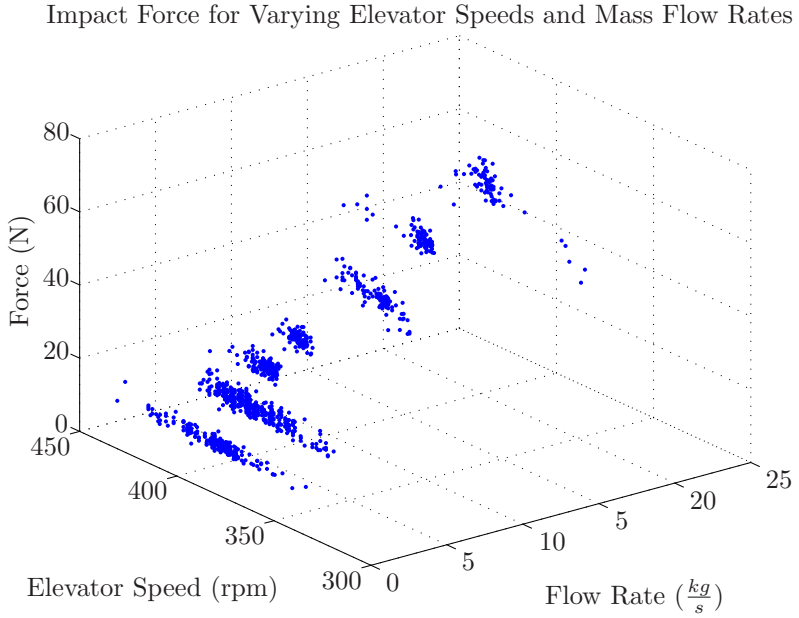

Figure 11.19: Experimental data displaying the relationship between measured force, elevator speed, and the rate of mass flow for corn at 14% moisture.

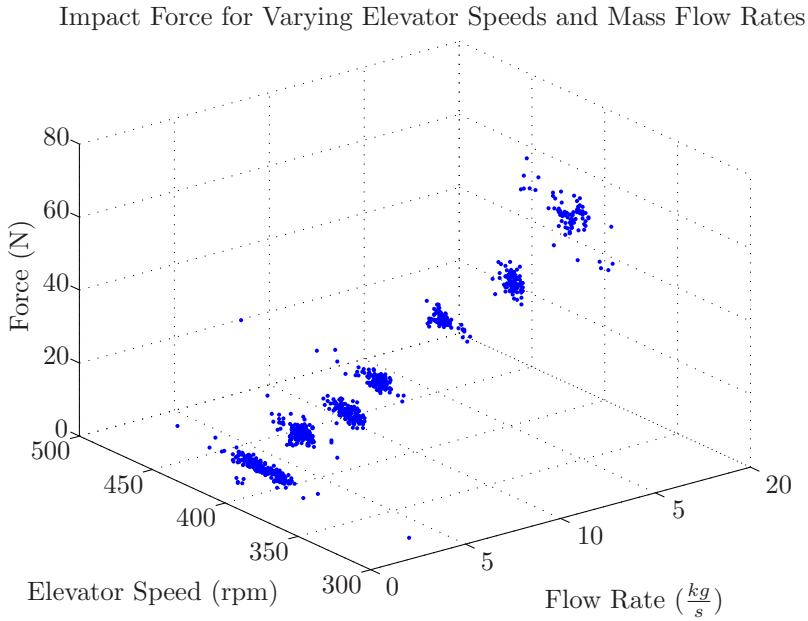

Figure 11.20: Experimental data displaying the relationship between measured force, elevator speed, and the rate of mass flow for corn at 21% moisture.

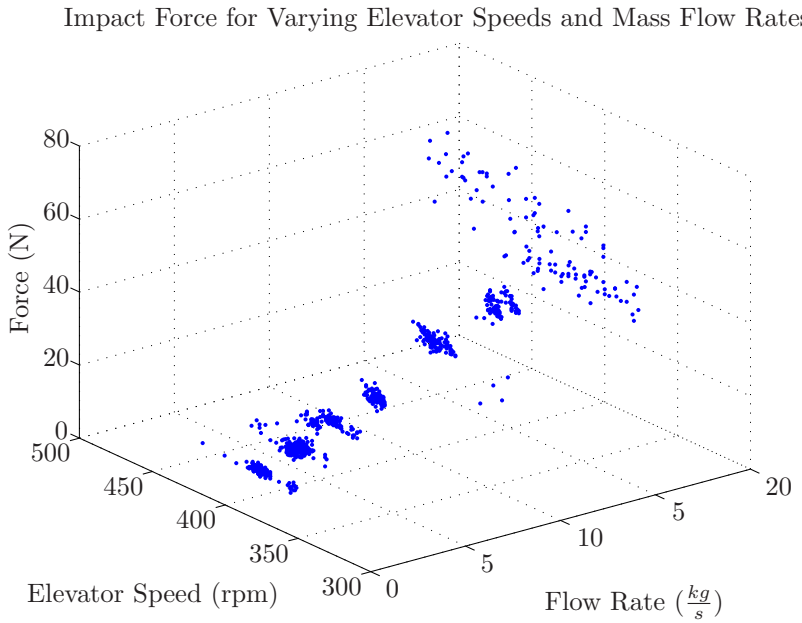

Figure 11.21: Experimental data displaying the relationship between measured force, elevator speed, and the rate of mass flow for corn at 26% moisture.

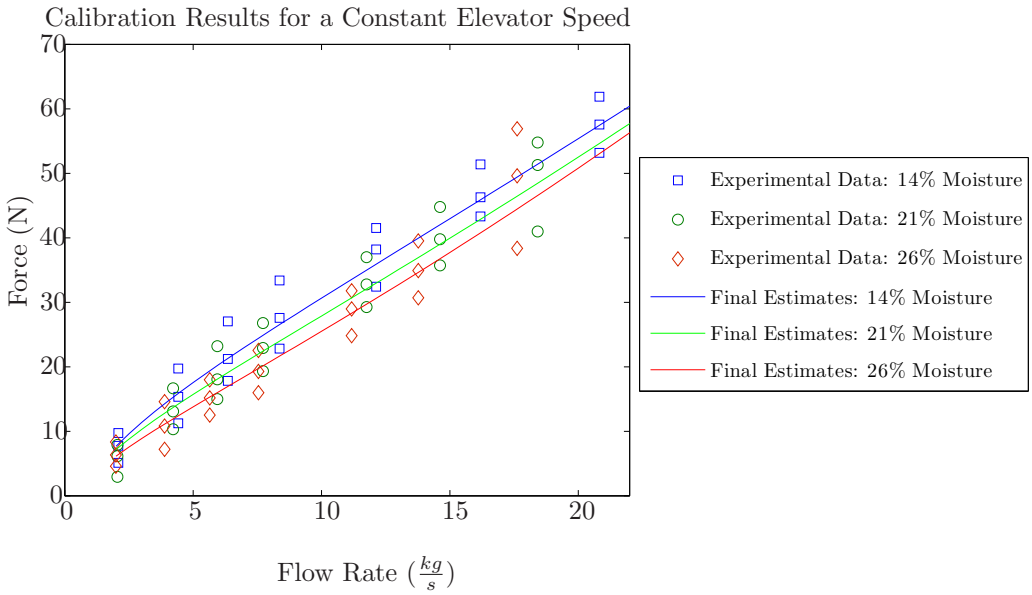

Figure 11.22: Results of the regression analysis for a constant elevator speed of 400 rpm and grain at moisture levels 14%, 21%, and 26%. The discrete points represent the maximum, minimum, and median measured force for each mass flow rate. The solid curves are model predictions, obtained using model parameters from the system calibration routine performed for an impact plate orientation of 68°.

Table 11.17: Summary of estimated model parameters returned from the regression algorithm, performed for an impact plate orientation of 68°.

	μ	$\eta\left(\frac{kg}{m^3}\right)$	k_1	k_2
Initial guess (14% moisture)	0.09	1,752	1.31	0.15
Final result (14% moisture)	0.08	1,751	1.65	0.61
Initial guess (21% moisture)	0.19	1,620	0.69	0.47
Final result (21% moisture)	0.15	1,601	2.01	0.57
Initial guess (26% moisture)	0.18	1,999	0.64	0.86
Final result (26% moisture)	0.28	1,999	3.06	0.48

Table 11.18: Summary of RMSR from the regression process. The NRMSR is calculated by dividing the RMSR by the maximum observed force for each experiment.

Corn Moisture Content	RMSR (N)	NRMSR
14%	1.94	3.09%
21%	1.74	2.78%
26%	1.67	2.64%

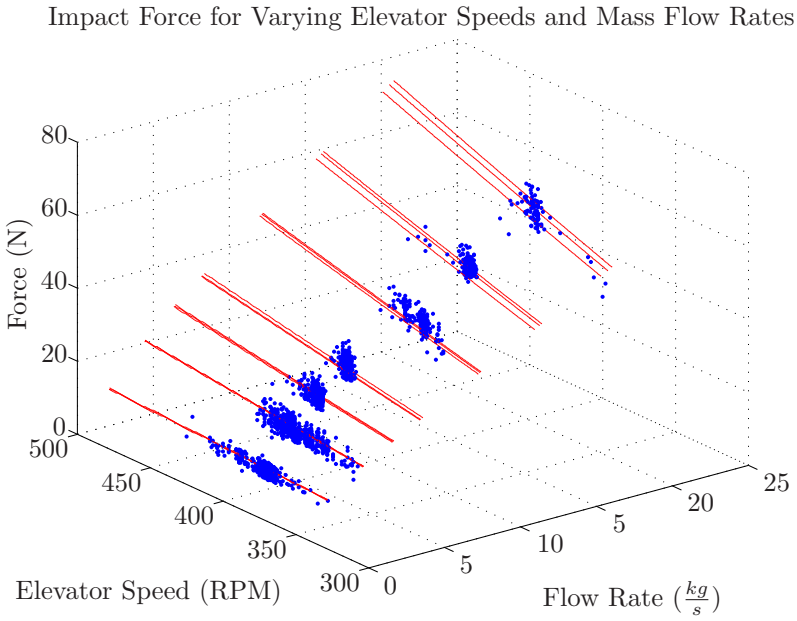

Figure 11.23: Results of the regression analysis for each mass flow rate for grain at 14% moisture. The solid curves are model predictions, obtained using model parameters from the system calibration routine performed for an impact plate orientation of 68°.

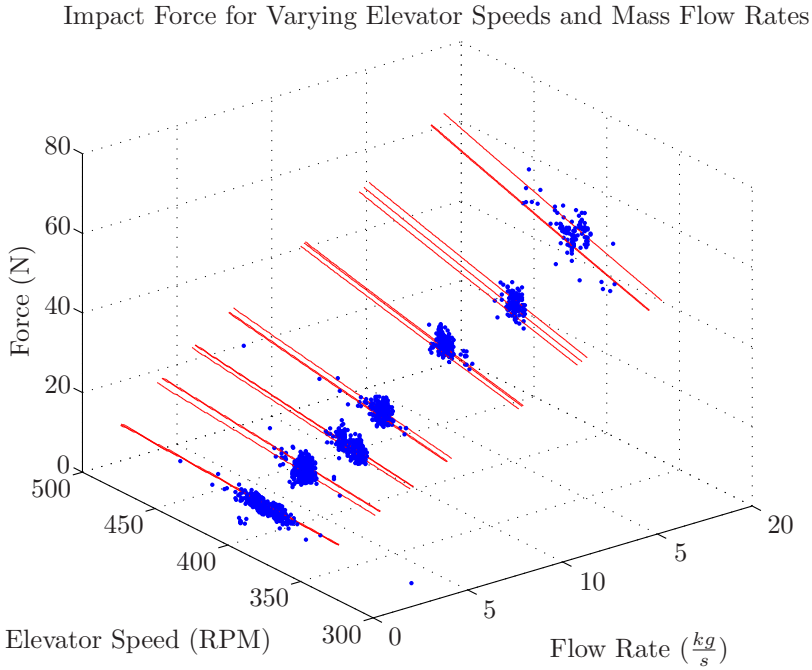

Figure 11.24: Results of the regression analysis for each mass flow rate for grain at 21% moisture. The solid curves are model predictions, obtained using model parameters from the system calibration routine performed for an impact plate orientation of 68°.

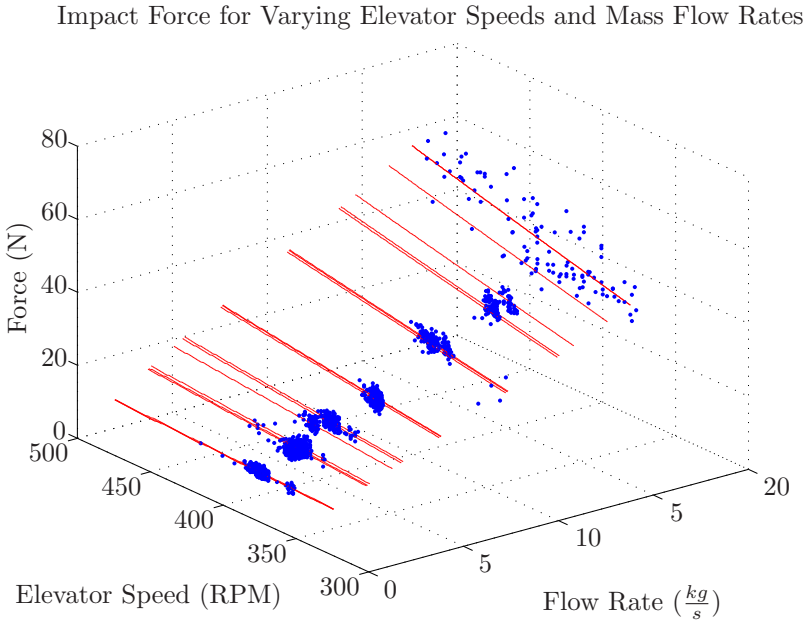

Figure 11.25: Results of the regression analysis for each mass flow rate for grain at 26% moisture. The solid curves are model predictions, obtained using model parameters from the system calibration routine performed for an impact plate orientation of 68°.

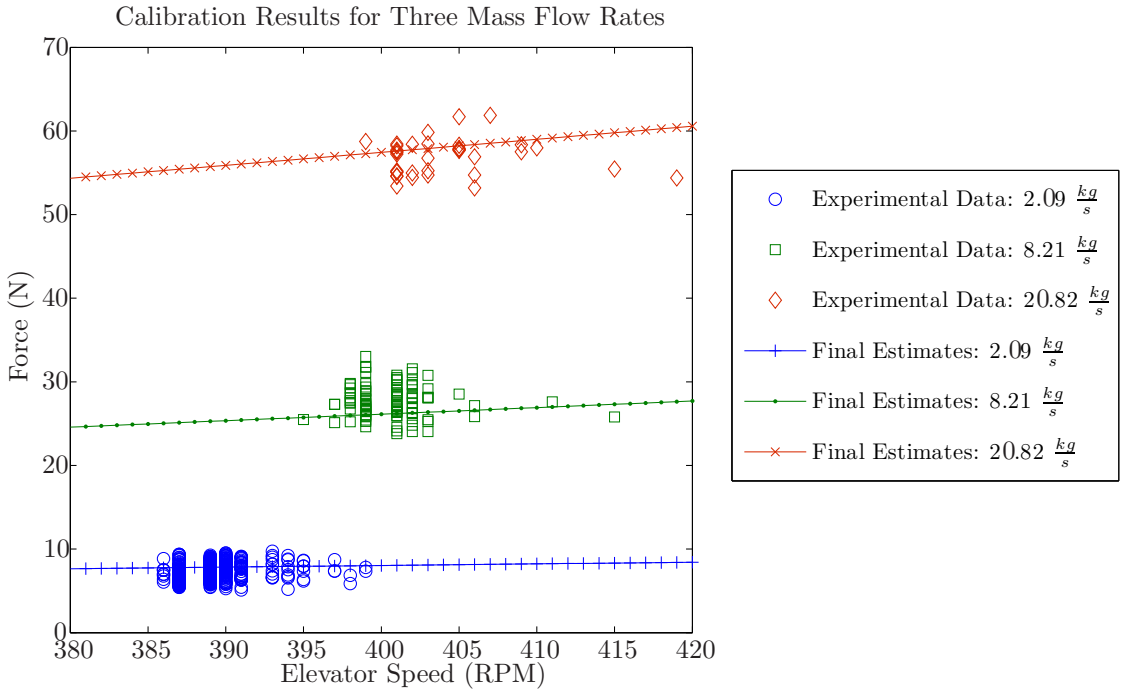

Figure 11.26: Results of the regression analysis as a function of elevator speed for grain at 14% moisture. The solid curves are model predictions, obtained using model parameters from the system calibration routine performed for an impact plate orientation of 68°.

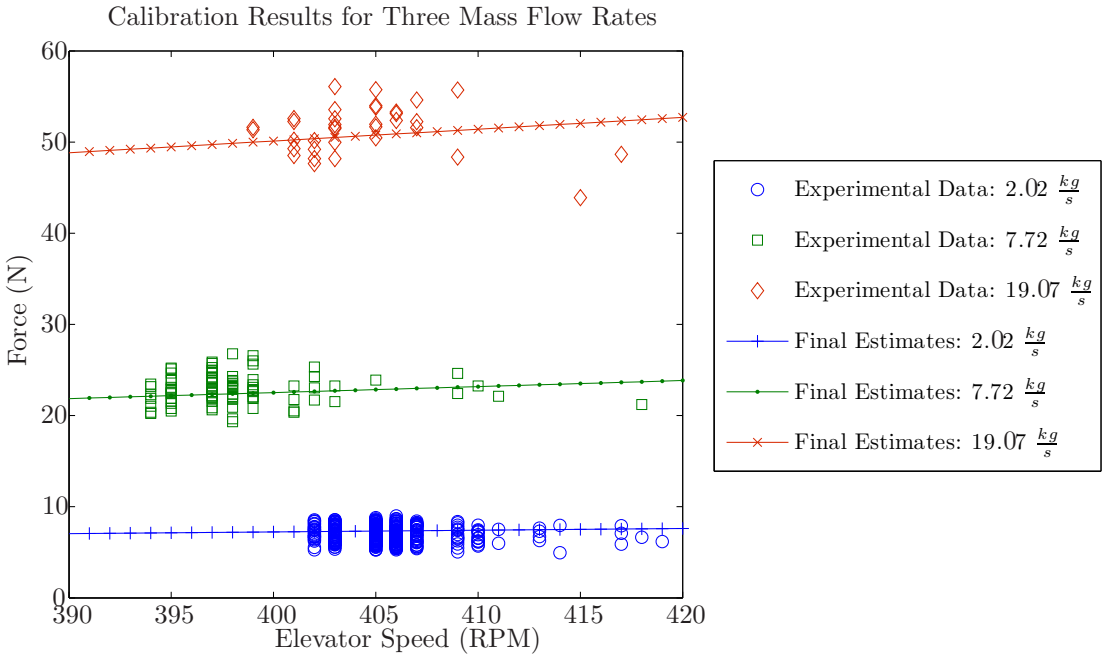

Figure 11.27: Results of the regression analysis as a function of elevator speed for grain at 21% moisture. The solid curves are model predictions, obtained using model parameters from the system calibration routine performed for an impact plate orientation of 68°.

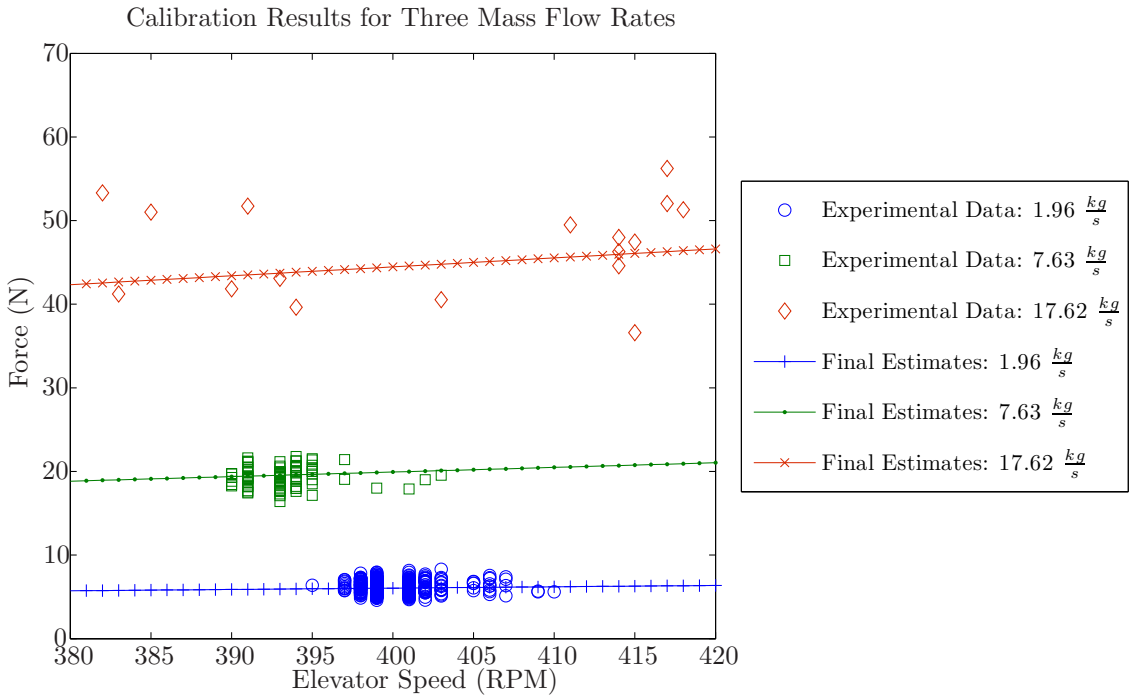

Figure 11.28: Results of the regression analysis as a function of elevator speed for grain at 26% moisture. The solid curves are model predictions, obtained using model parameters from the system calibration routine performed for an impact plate orientation of 68°.

Table 11.19: Summary of RMSR from the open-loop estimation process. The NRMSR is calculated by dividing the RMSR by the maximum observed mass flow rate for each experiment. The open-loop estimation procedure was performed using model parameters obtained from the system calibration procedure performed for an impact plate orientation of 68°.

Corn Moisture Content	RMSR $\left(\frac{kg}{s}\right)$	NRMSR
14%	0.6	2.89%
21%	0.57	2.97%
26%	0.71	4.02%

Table 11.20: Summary of mass flow estimates returned from the closed-loop estimation process.

Moisture Content	True $\dot{m}\left(\frac{kg}{s}\right)$	Estimated $\dot{m}\left(\frac{kg}{s}\right)$	Percent Error
14%	4.44	3.69	16.90%
14%	8.45	7.76	8.17%
14%	15.61	14.15	9.35%
21%	4.10	3.21	21.70%
21%	7.87	7.08	10.03%
21%	14.61	13.17	9.87%
26%	3.90	3.46	11.28%
26%	7.52	6.44	14.36%
26%	14.02	9.72	30.67%

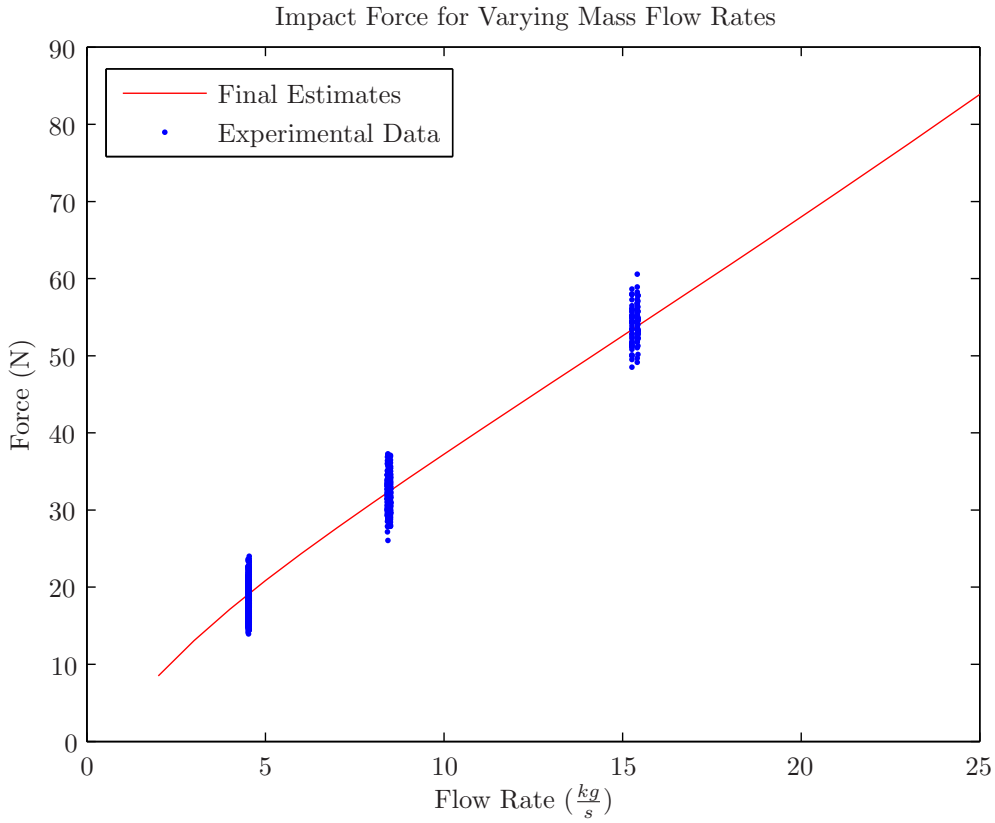

Figure 11.29: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 14% moisture using a 90° impact plate.

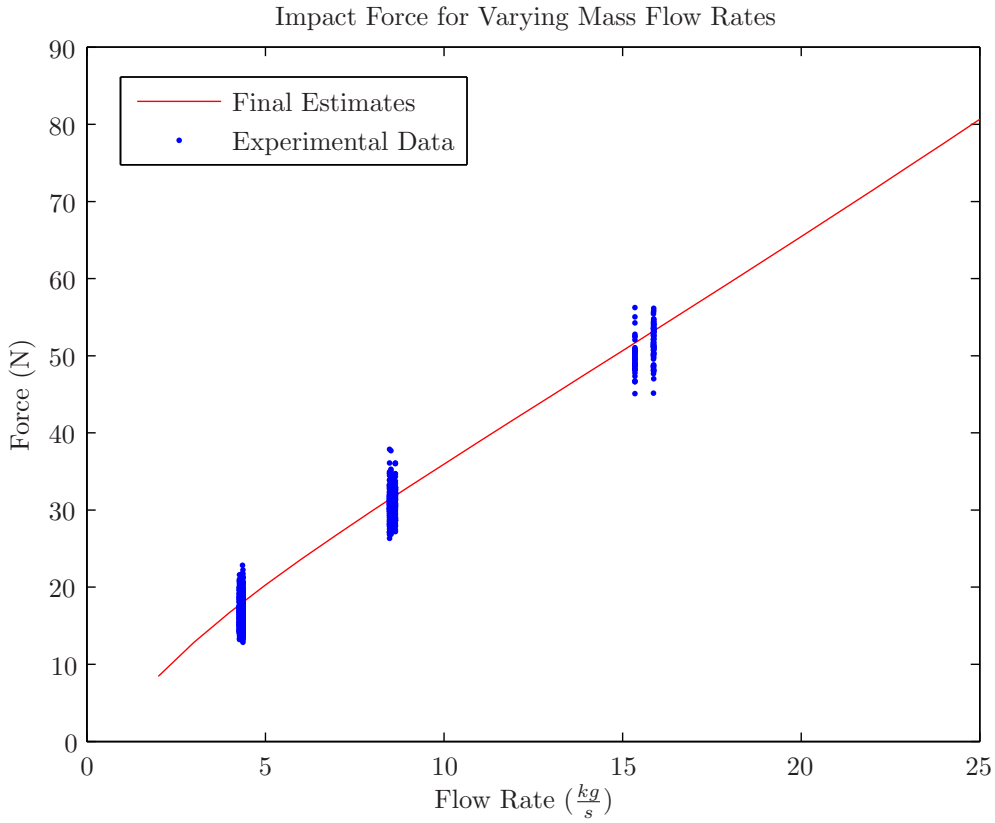

Figure 11.30: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 14% moisture using a 80° impact plate.

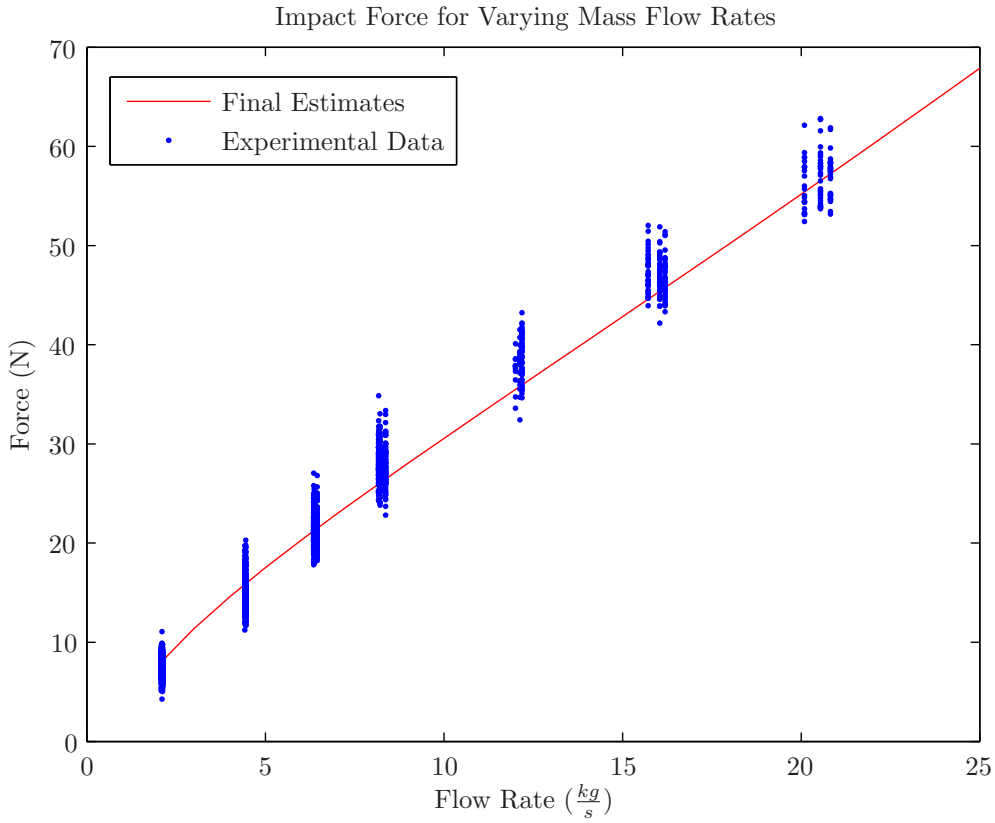

Figure 11.31: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 14% moisture using a 68° impact plate.

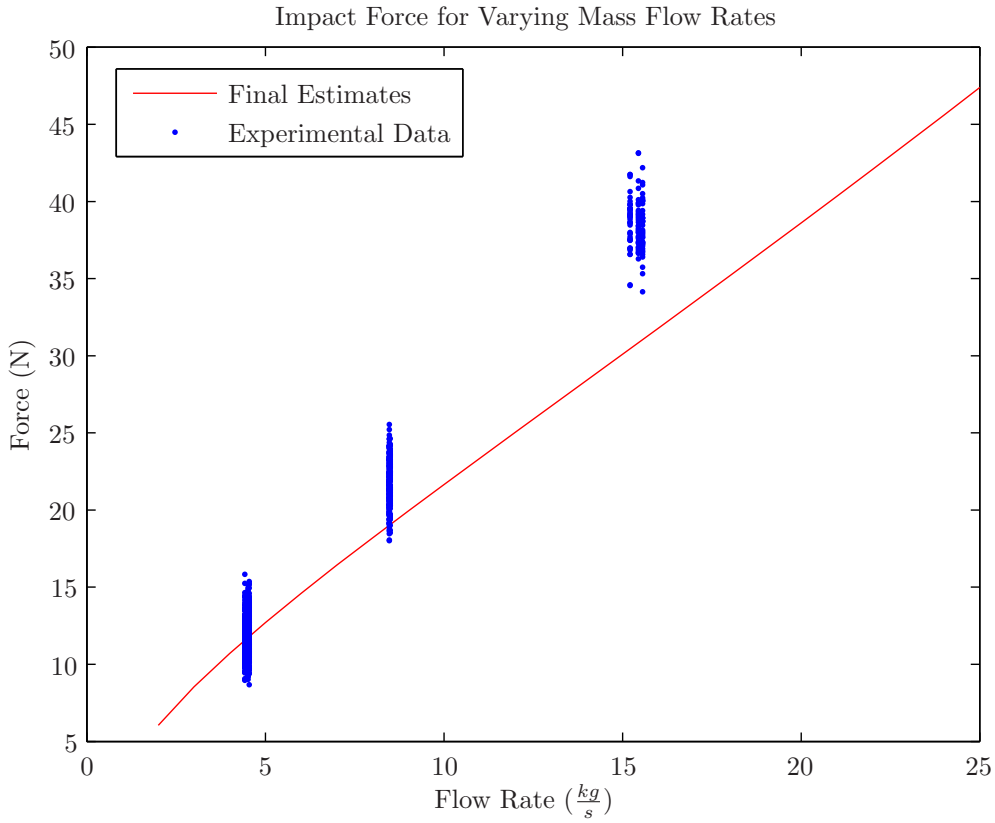

Figure 11.32: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 14% moisture using a 55° impact plate.

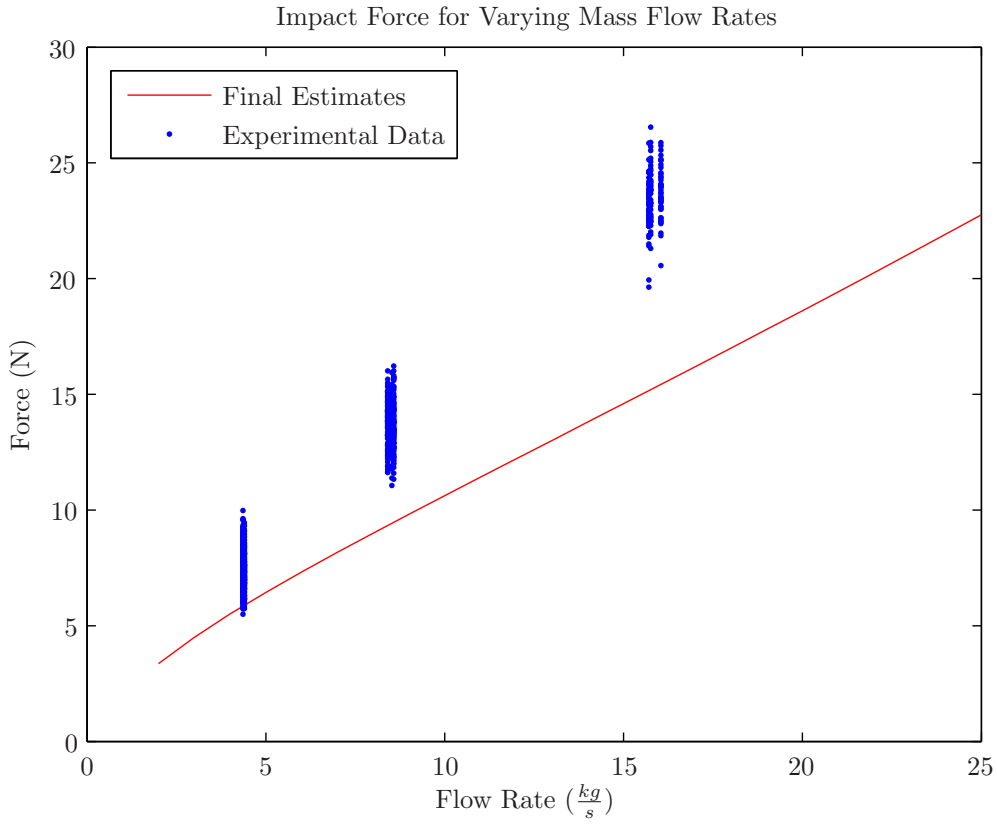

Figure 11.33: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 14% moisture using a 40° impact plate.

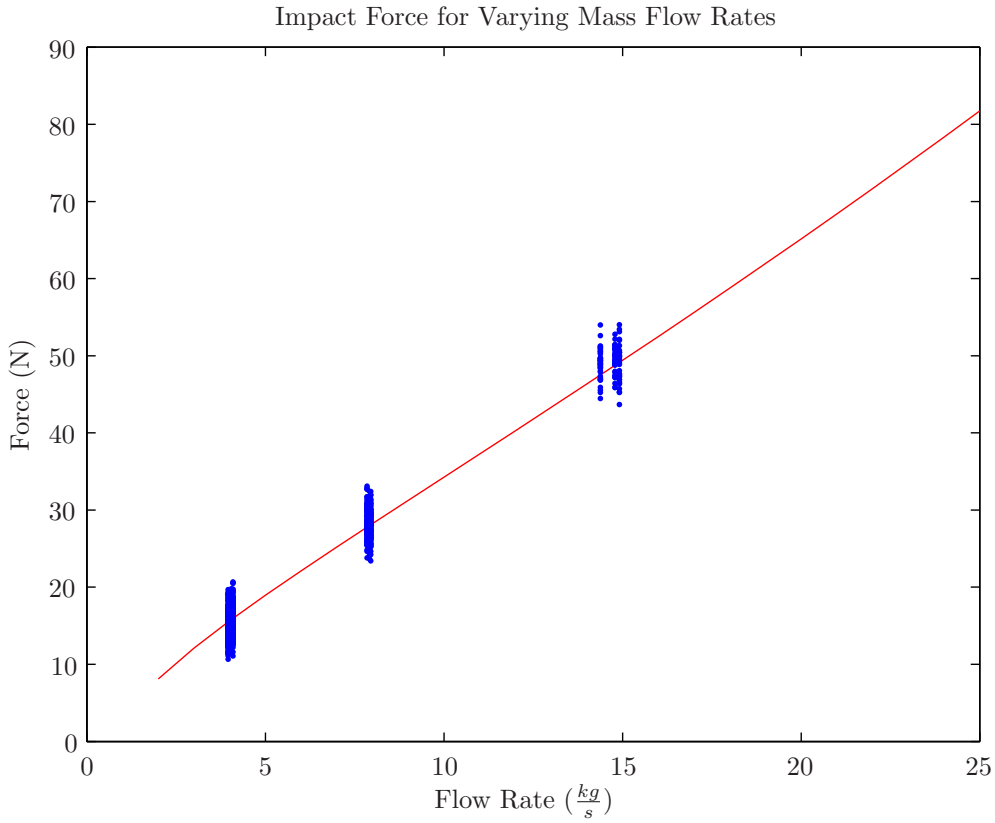

Figure 11.34: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 21% moisture using a 90° impact plate.

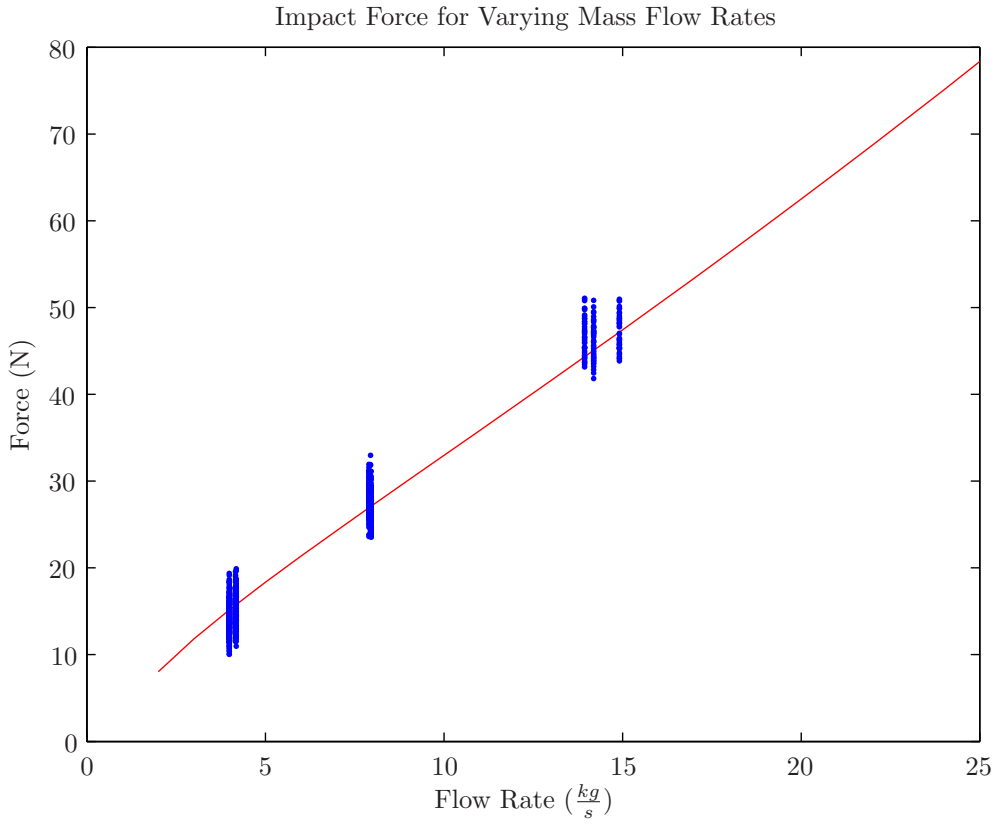

Figure 11.35: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 21% moisture using a 80° impact plate.

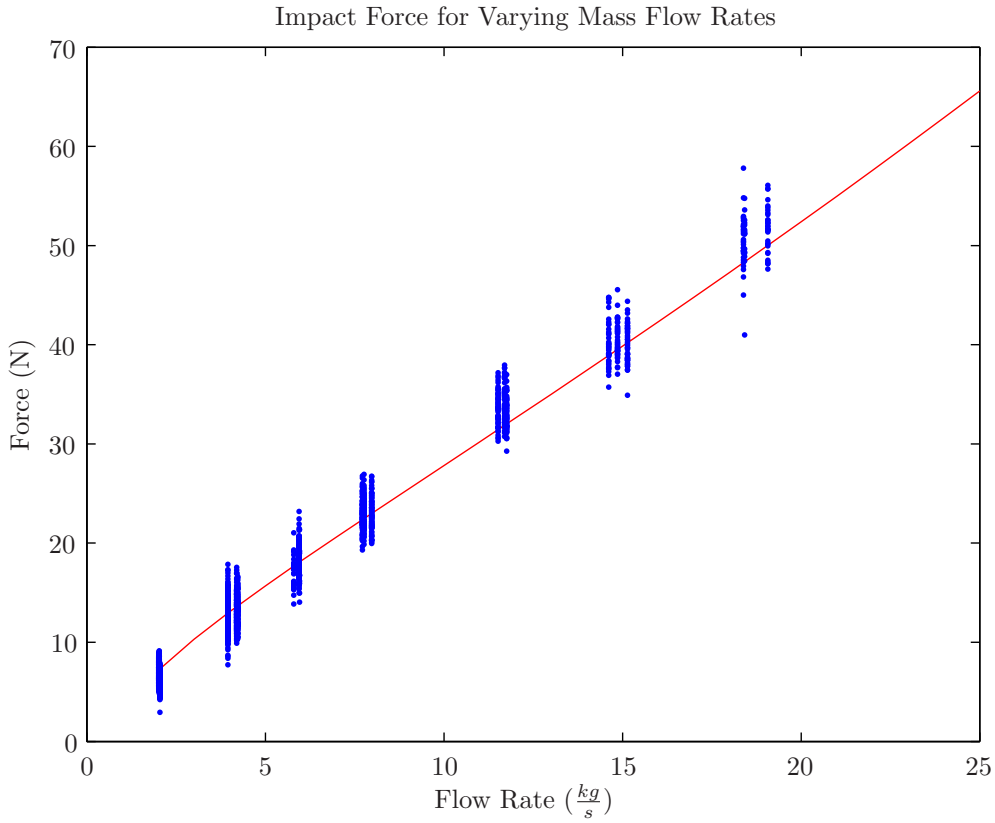

Figure 11.36: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 21% moisture using a 68° impact plate.

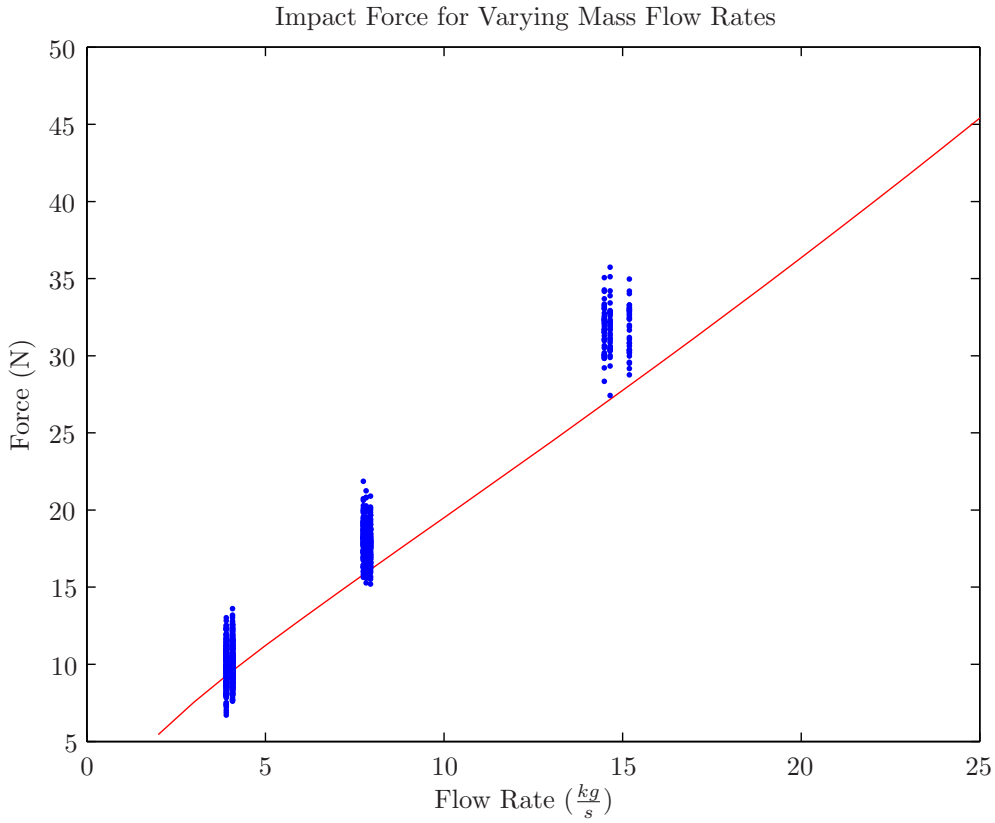

Figure 11.37: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 21% moisture using a 55° impact plate.

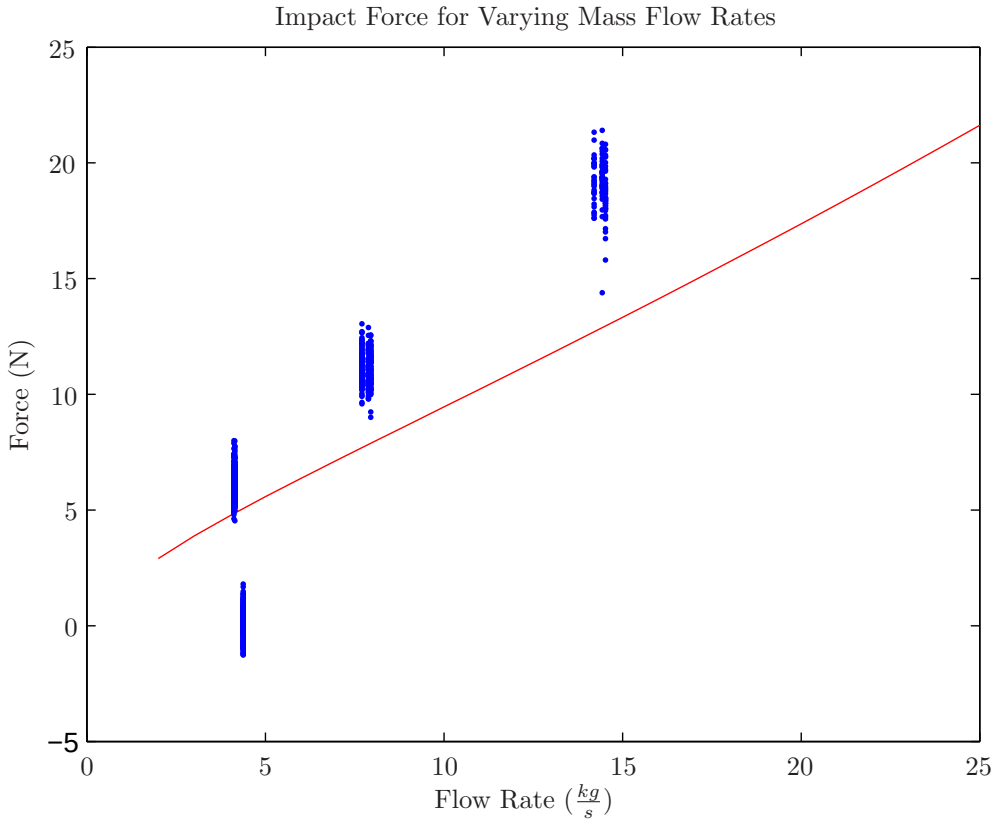

Figure 11.38: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 21% moisture using a 40° impact plate.

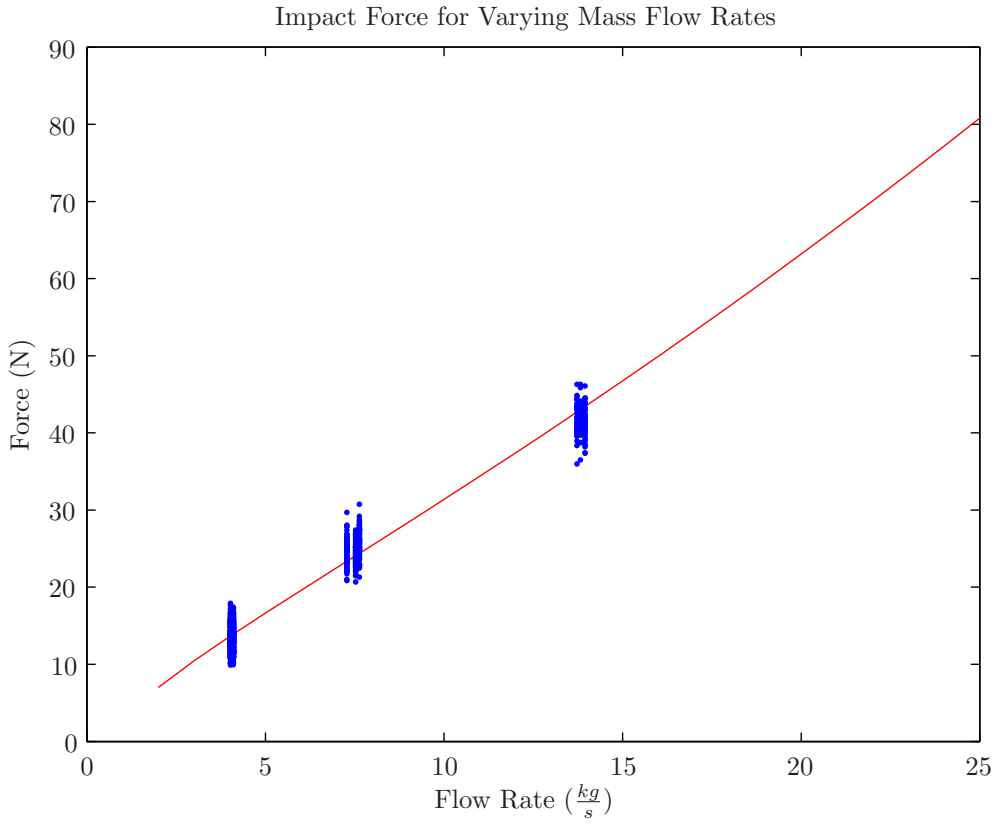

Figure 11.39: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 26% moisture using a 90° impact plate.

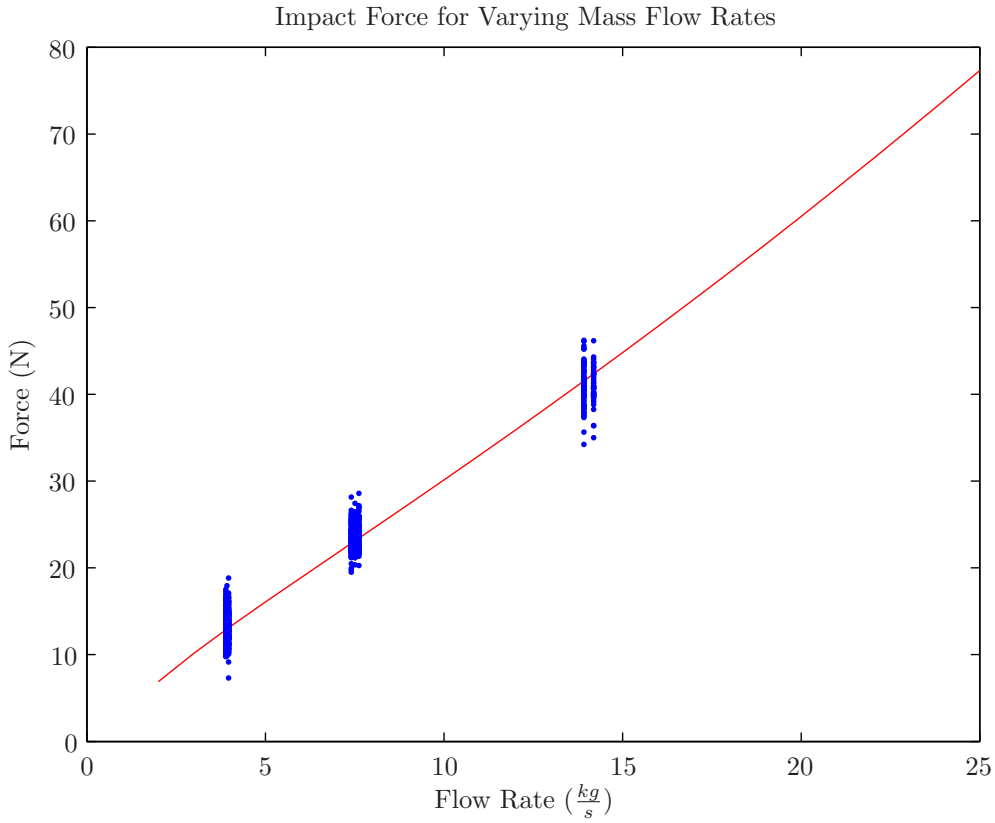

Figure 11.40: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 26% moisture using a 80° impact plate.

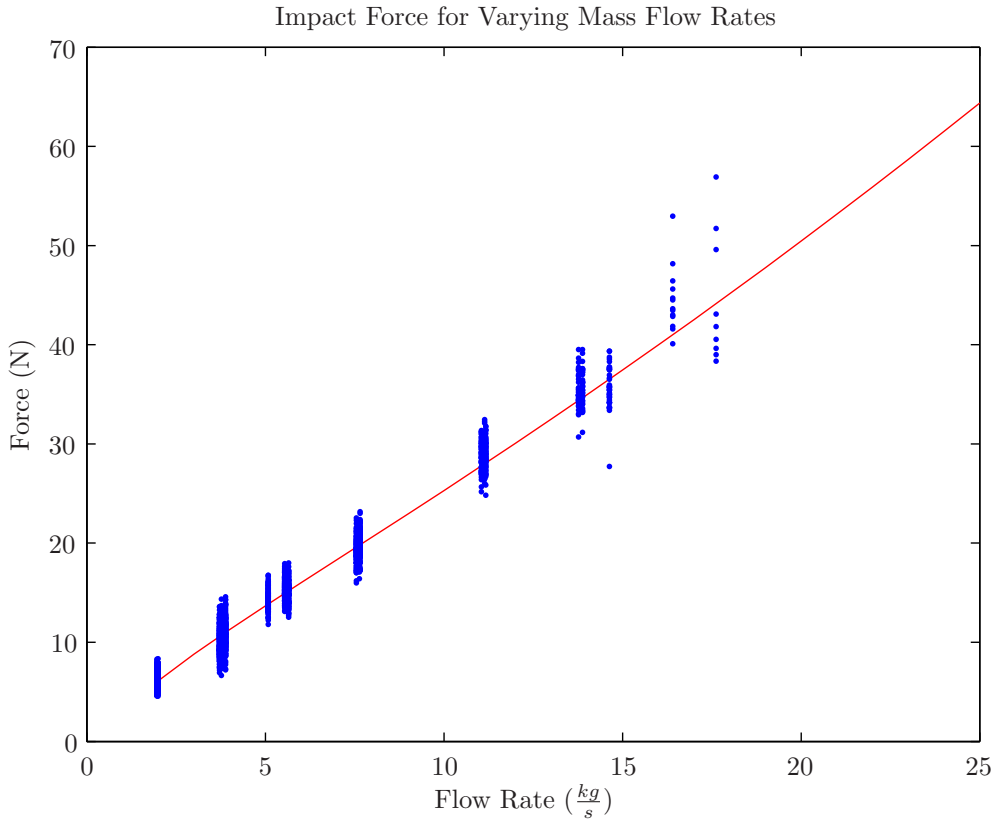

Figure 11.41: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 26% moisture using a 68° impact plate.

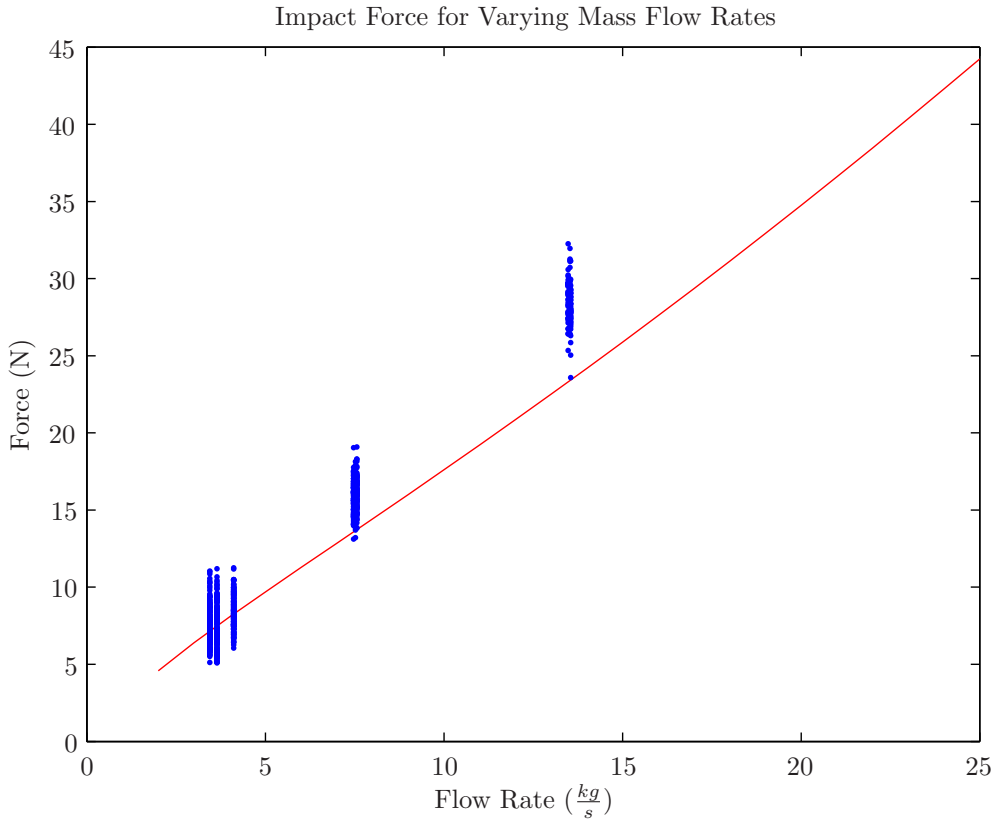

Figure 11.42: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 26% moisture using a 55° impact plate.

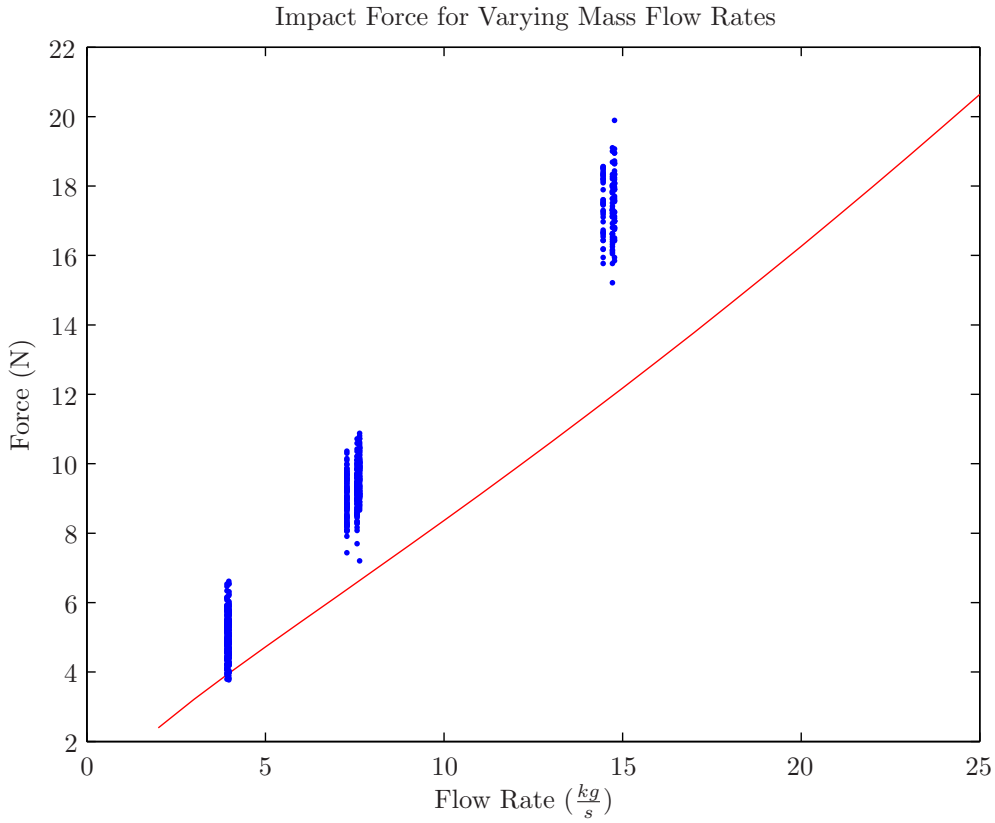

Figure 11.43: Impact force for varying rates of mass flow. The solid curve is the predicted dependence using the final estimates for model parameters obtained from the system calibration routine performed for an impact plate orientation of 68°. Discrete points represent data for corn at 26% moisture using a 40° impact plate.

Chapter 12

Dependence of the computational model on model parameters

The previous chapters have shown that a physics-based model, such as the one described above, in combination with a regression algorithm results in satisfactory agreement between mass flow rate estimations and experimental results. To further investigate the effects of changes in model parameters on the relationship between momentum imparted to the impact plate and mass flow rate, a study was conducted examining the dependence of the computational model on the model parameters μ, e, η, and h_f. This was accomplished by varying one parameter while fixing all other parameters at a typical value, and plotting the dependence of the predicted momentum on the free parameter. This analysis illustrates the sensitivity of the model to variations in each model parameter.

Typical values for each parameter are: $\mu = 0.5$, $e = 0.5$, $\eta = 1500\frac{kg}{m^3}$, and $h_f = 2$, with rates of mass flow between $2\frac{kg}{s}$ and $26\frac{kg}{s}$. To graph the dependence on μ, μ was varied between 0.1 and 2 while keeping all other parameters at their typical values. Similarly, in graphing the dependence on e, e was varied between 0.1 and 1 while keeping all other model parameters constant. The dependence on η was constructed by varying η between 750 $\frac{kg}{s}$ and 2000 $\frac{kg}{s}$, and the dependence on h_f was examined by varying h_f between 1.5 and 2.5. The results of this study are shown in Figures 13.1-13.4.

As seen from these results, the predicted momentum decreases in a non-linear fashion with increasing values of μ, increases linearly with increasing values of e, and increases in a non-linear fashion with increasing values of η and h_f. These trends exist for a number of reasons. Increases in μ contribute to more dissipation of the energy of the grain within the system, and in turn a larger value for the critical radius, ρ_{crit}, thereby lowering the resulting momentum imparted to the impact plate. Conversely, increases in e contribute to the efficiency of energy transfer from the grain to the impact plate, resulting in a higher transfer of momentum. Higher grain density, η, for constant rates of mass flow, leads to increases in kinetic energy of the grain which is ultimately transfered to the impact plate, thereby leading to higher momentum. Larger values of h_f contribute to a distribution of grain nearer to the tip of the paddle where the rotational velocity is higher, which ultimately leads to higher grain velocities for a larger number of grains. Additionally, grains at the furthest radial layers during the settling phase

are more likely to have a trajectory that will intersect with the impact plate and transfer momentum to the plate, rather than miss below the plate and transfer no momentum. In general, the momentum transferred to the impact plate depends more strongly on the model parameters μ and e than η and h_f.

Chapter 13

Figures and Tables

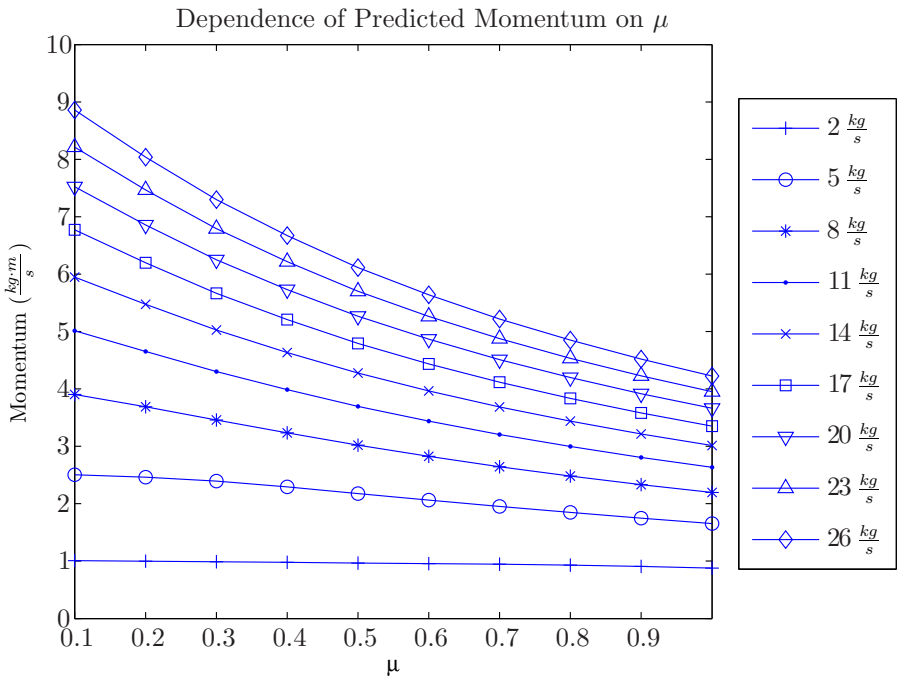

Figure 13.1: Dependence of the computational model on μ.

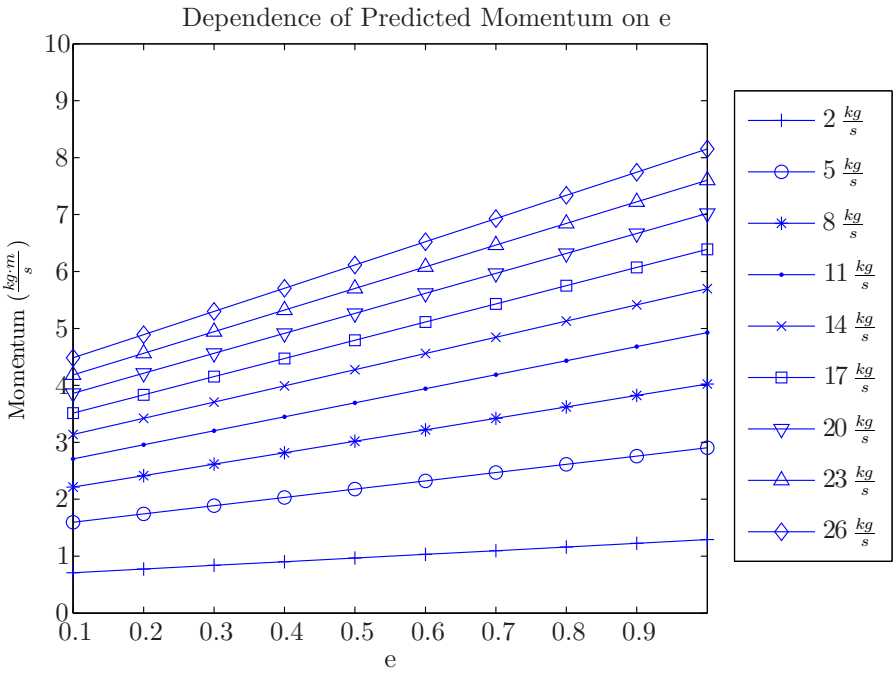

Figure 13.2: Dependence of the computational model on e.

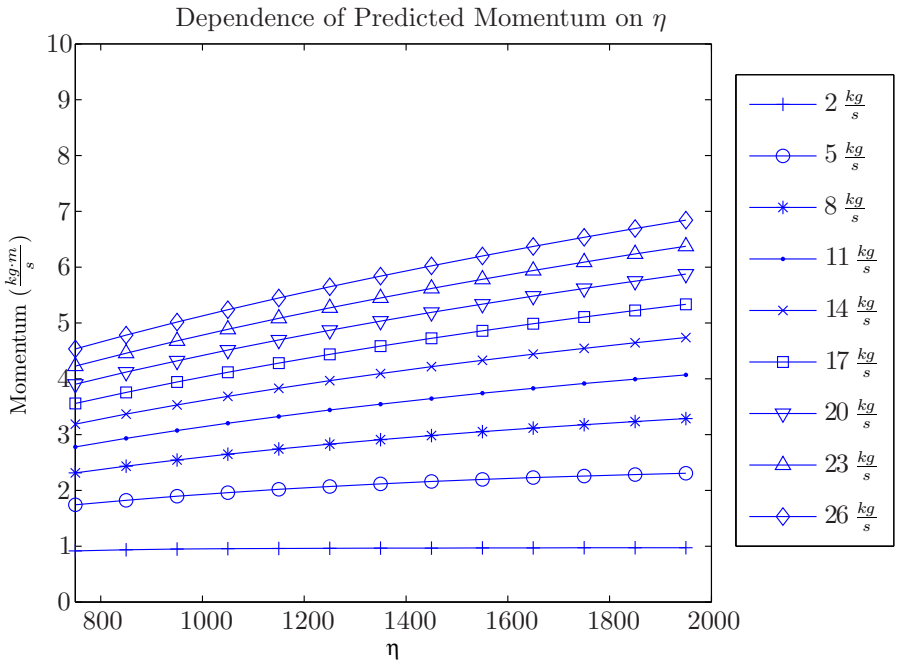

Figure 13.3: Dependence of the computational model on η.

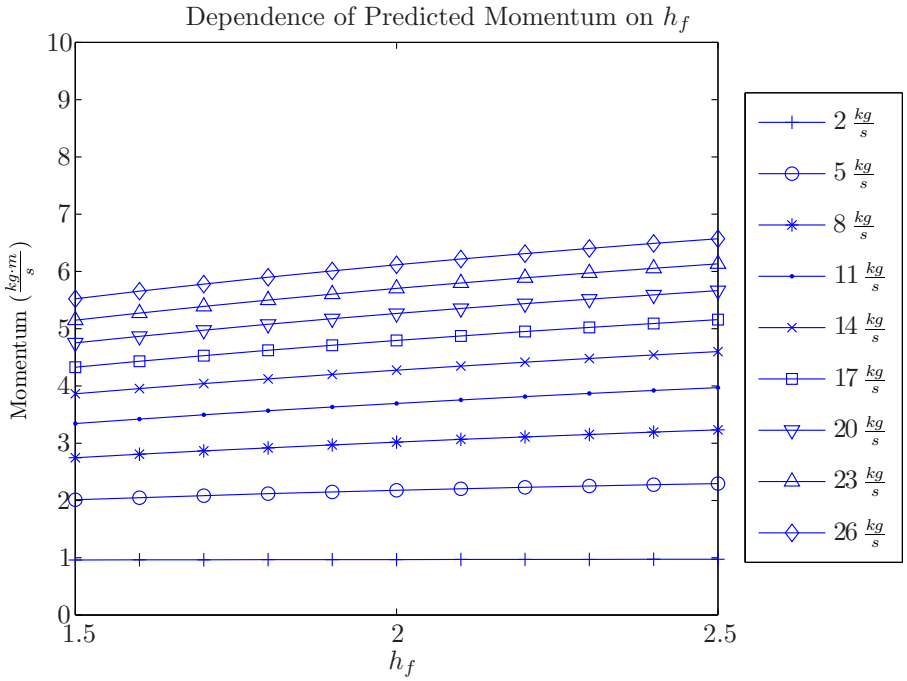

Figure 13.4: Dependence of the computational model on h_f.

Chapter 14

Summary and Conclusions

The work presented in this document serves as the thesis for the Master of Science degree in Mechanical Engineering. The topic *Self-Calibrating Mass Flow Sensor* was an effort in partnership with Deere and Co. to develop a method for improved accuracy and self-calibration for a mass flow sensor system on a combine. This effort has resulted in

- A physics-based mathematical model that expresses the relationship between the rate of mass flow through the combine and the measured force imparted to the impact plate in terms of mechanical properties of the grains and the interior geometry of the combine.
- A Matlab-based computational algorithm that implements the mathematical model and allows for prediction of the force imparted to the impact plate, given parameter values describing the grain properties, the combine geometry, and an input rate of mass flow.
- A nonlinear regression algorithm allowing for the estimation of model parameters for a known combine geometry and pairs of rates of mass flow and measured force.
- A proposed method of inducing known changes to the impact plate orientation as a means for self-calibration of the sensor system.
- A nonlinear regression algorithm allowing for the estimation of mass flow rate while simultaneously updating model parameters, resulting in a self-calibration procedure.
- A discrete element modeling realization of the relevant combine geometry as a simulation-based tool for validation of the constructed models.
- Design and development of a lab-sized testbench to simulate the operation of the sensor system to enable experimentation and data collection for validation of the mathematical model and self-calibration procedure.
- Experimental data collected via a replica of the sensor system at the Combine Yield Monitor Test Facility, using a lab-sized testbench, and through discrete element modeling simulations.

- Validation of the use of the mathematical model, regression methods, and self-calibration procedure, via data acquired from experiments and simulations, to enable accurate mass flow rate estimation and self-calibration of the sensor system.

Together, these deliverables establish a means for accurate mass flow rate estimation and self-calibration for a mass flow sensor on harvesting combines.

The physics-based mathematical model quantifies the relationship between mass flow rate and measured force imparted to the impact plate. This model was arrived at through a successive causal association between inputs and outputs of various stages of the physical process, in which the output of one stage constituted the input of the immediate subsequent stage. These relationships were composed of elements describing the mechanical properties of the grain and the geometric dimensions and operation of the mass flow sensing system.

The selection of the method of inducing modifications to the impact plate orientation as a means of achieving real-time calibration was based upon an observed strong dependence of the momentum imparted to the impact plate on the orientation of the impact plate. The viability of this concept was examined using data acquired through simulations and experimentation.

Computer simulations were conducted using discrete element modeling software, which allowed for strict control of grain properties and operation of the system, while enabling investigation into specific aspects of the grain dynamics throughout the operation of the system. Additionally, experimental data was collected via two distinct methods. Data acquired from experiments conducted at the Combine Yield Monitor Test Facility provided data relating the impact force, mass flow rate, elevator speed, impact plate orientation, and grain moisture levels for the clean grain elevator system commonly found on John Deere combines. Small-scale laboratory experiments allowed for strict control of machine operational characteristics and geometric system properties. These simulations and experiments facilitated validation of the mass flow rate estimation procedure, as well as the concept of using modifications to the impact plate orientation as a means of achieving self-calibration.

A realization of the physics-based model was implemented in MATLAB to facilitate numerical calculation of the computable relationship between mass flow rate and impact force given numerical values for mass flow rate, machine geometric and operational characteristics, and model parameters. Using this computational algorithm, three distinct forms of model-based estimation were achieved:

1. system calibration: The process of estimating internal model parameters to enable agreement between measured impact force and predicted impact force from the physics-based model, given input pairs of mass flow rate and measured impact force, and machine geometric parameters.

2. open-loop estimation: The process of estimating mass flow rate given values for internal model parameters, impact forces, and machine geometric parameters.

3. closed-loop estimation: The process of intentionally introducing known variations in machine parameters to generate a large sample of measured impact forces, while model parameters and mass flow rate remain constant, in order to simultaneously estimate values for internal model parameters and mass flow rate. This facilitates self-calibration of the mass flow sensing system.

These model-based estimation routines were successfully demonstrated using data acquired from simulations and experiments. In the case of system calibration, this equated to close agreement between measured and predicted impact forces through determination of internal model parameters. For open-loop estimation, this resulted in estimation of mass flow rate. Finally, for the case of closed-loop estimation, this resulted in an update of internal model parameters and an estimate of mass flow rate (for computer simulations and small-scale experiments) to enable close agreement between predicted and measured forces for known changes in system geometry. The results of the closed-loop estimation procedure applied to the data from large-scale experiments were less convincing, as a large spread in the experimental data resulted in significant residuals in the estimation of mass flow rate during this procedure.

Nevertheless, the results of this work demonstrate that the use of the mathematical model in combination with model-based regression is a promising approach in achieving accurate mass flow rate estimation and self-calibration of the mass flow sensing system. There is opportunity for the development of an improved model to capture the relationship between momentum imparted to the impact plate and the rate of mass flow, as well as to better capture the dependence on the orientation of the impact plate. This would facilitate improved mass flow rate estimation and self-calibration of the sensor system. Design and development of a prototype device capable of real-time variation of the impact plate orientation would be beneficial as well in order to field-test this technology and accelerate its progression toward commercialization.

Appendix A

Paddle filling process during linear motion for an arbitrary cross sectional auger opening

In the case that the auger deposits grain onto elevator paddles during the time the paddles move in a linear fashion across the auger opening (see Figure B.1), the amount of grain deposited onto a specific paddle of interest can be analyzed. The percentage opening of the auger area available to a specific paddle of interest is described by

$$P(t) = \begin{cases} \frac{s(t)}{A} & : t_1 < t < t_2 \\ 1 & : t_2 < t < t_3 \\ 1 - \frac{s(t)}{A} & : t_3 < t < t_4 \end{cases} \tag{A.1}$$

where $s(t)$ is the area of the auger exposed to the most-trailing elevator paddle available to the auger, and A is the total opening area of the auger. The total mass deposited on a single paddle equals

$$M = \dot{m} \int P \, dt. \tag{A.2}$$

It follows that

$$M = \dot{m} \int P \, dt = \dot{m} \left(\int_{t_1}^{t_2} \frac{s(t)}{A} \, dt + \int_{t_2}^{t_3} 1 \, dt + \int_{t_3}^{t_4} 1 - \frac{s(t)}{A} \, dt \right). \tag{A.3}$$

It is imperative to note that the elapsed time between t_1 and t_2 is equal to the elapsed time between t_3 and t_4, which is the time required for one paddle to traverse the auger opening. Furthermore, the percentage of the opening area of the auger available to the paddle of interest during these times is equivalent. It follows that

$$M = \dot{m} \int P \, dt = \dot{m} \left(\int_{t_1}^{t_2} \frac{s(t)}{A} \, dt + \int_{t_2}^{t_3} 1 \, dt + \int_{t_1}^{t_2} 1 - \frac{s(t)}{A} \, dt \right), \tag{A.4}$$

which simplifies to

$$M = \dot{m} \int P \, dt = \dot{m} \left(\int_{t_1}^{t_3} 1 \, dt \right). \tag{A.5}$$

Noting that the elapsed time between t_3 and t_1 is equivalent to $\frac{d}{V}$, the mass deposited onto a single paddle for any arbitrary cross-sectional area opening of the auger is described by

$$M = \dot{m} \int P \, dt = \dot{m} \frac{d}{V}. \tag{A.6}$$

Appendix B

Figures

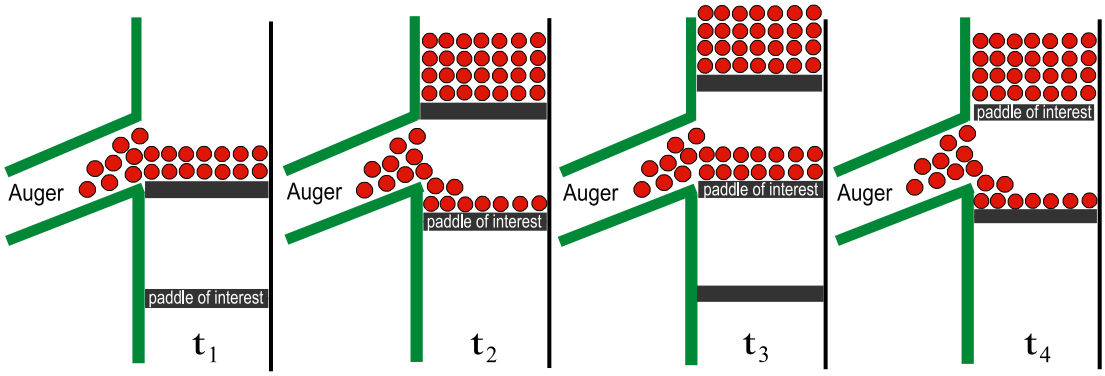

Figure B.1: Successive stages in the process of depositing grain exiting from an auger onto a single paddle and onto adjacent paddles as the paddles move in a circular fashion about the sprocket.

Appendix C

Paddle filling process during circular motion for an arbitrary cross sectional auger opening

In the case that the auger deposits grain onto elevator paddles during the time the paddles move in a circular fashion around the sprocket (see Figure D.1), the amount of grain deposited onto a specific paddle of interest can be analyzed. The percentage opening of the auger area available to a specific paddle of interest is described by

$$P(t) = \begin{cases} \frac{s(t)}{A} & : t_1 < t < t_2 \\ 1 & : t_2 < t < t_3 \\ 1 - \frac{s(t)}{A} & : t_3 < t < t_4 \end{cases} \tag{C.1}$$

where $s(t)$ is the area of the auger exposed to the most-trailing elevator paddle available to the auger, and A is the total opening area of the auger. The total mass deposited on a single paddle equals

$$M = \dot{m} \int P \, dt. \tag{C.2}$$

It follows that

$$M = \dot{m} \int P \, dt = \dot{m} \left(\int_{t_1}^{t_2} \frac{s(t)}{A} \, dt + \int_{t_2}^{t_3} 1 \, dt + \int_{t_3}^{t_4} 1 - \frac{s(t)}{A} \, dt \right). \tag{C.3}$$

It is imperative to note that the elapsed time between t_1 and t_2 is equal to the elapsed time between t_3 and t_4, which is the time required for one paddle to traverse the auger opening. Furthermore, the percentage of the opening area of the auger available to the paddle of interest during these times is equivalent. It follows that

$$M = \dot{m} \int P \, dt = \dot{m} \left(\int_{t_1}^{t_2} \frac{s(t)}{A} \, dt + \int_{t_2}^{t_3} 1 \, dt + \int_{t_1}^{t_2} 1 - \frac{s(t)}{A} \, dt \right), \tag{C.4}$$

which simplifies to

$$M = \dot{m} \int P \, dt = \dot{m} \left(\int_{t_1}^{t_3} 1 \, dt \right). \tag{C.5}$$

Noting that the elapsed time between t_3 and t_1 is equivalent to $\frac{\theta}{\Omega}$, the mass deposited onto a single paddle for any arbitrary cross-sectional area opening of the auger is described by

$$M = \dot{m} \int P \, dt = \dot{m} \frac{\theta}{\Omega}. \tag{C.6}$$

Appendix D

Figures

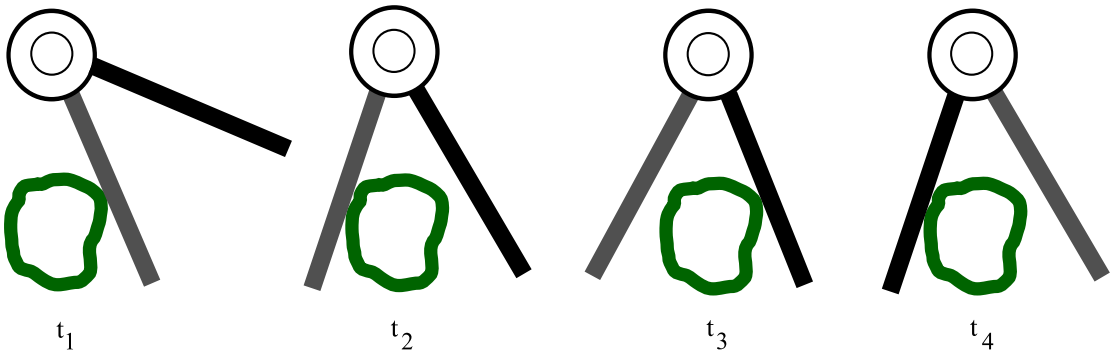

Figure D.1: Successive stages in the process of depositing grain exiting from an auger (green) onto a single paddle (black) and onto adjacent paddles (gray), as the paddles move in a circular fashion about the sprocket.

Appendix E

Derivation of the nonlinear shape function

Through DEM simulations it was found that the inner boundary of the distribution of grain was a function of the angular displacement from the paddle represented by $h(\psi)$ which corresponded to a straight line. Specifically, for a straight line

$$ax + by + c = 0 \tag{E.1}$$

where x and y are cartesian coordinates, and a, b, and c are arbitrary constants. With $x = h(\psi)\cos\psi$ and $y = h(\psi)\sin\psi$, it follows that

$$ah(\psi)\cos\psi + bh(\psi)\sin\psi + c = 0 \tag{E.2}$$

from which we obtain

$$h(\psi) = \frac{1}{\left(-\frac{a}{c}\right)\cos\psi + \left(-\frac{b}{c}\right)\sin\psi}. \tag{E.3}$$

Let h_0 denote the distance from the center of the sprocket to the inner boundary of the grain distribution for $\psi = 0$. Similarly, let ψ_f be the angle from the paddle at which the distance from the center of the sprocket to the inner boundary of the grain distribution equals the radius l of the housing. It follows that

$$h_0 = h(0) = \frac{1}{\left(-\frac{a}{c}\right)} \Rightarrow -\frac{a}{c} = \frac{1}{h_0} \tag{E.4}$$

and

$$l = h(\psi_f) = \frac{1}{\left(-\frac{a}{c}\right)\cos\psi_f + \left(-\frac{b}{c}\right)\sin\psi_f} \Rightarrow -\frac{b}{c} = \frac{\frac{1}{l} - \left(-\frac{a}{c}\right)\cos\psi_f}{\sin\psi_f} \tag{E.5}$$

and, consequently, that

$$h(\psi) = \left(\frac{1}{h_0}\cos\psi + \frac{\frac{1}{l} - \frac{1}{h_0}\cos\psi_f}{\sin\psi_f}\sin\psi\right)^{-1} \tag{E.6}$$

The function $h(\psi)$ is thus still parametrized by two parameters namely h_0 and ψ_f, both of which nominally are functions of the total mass of grain deposited onto the paddle. Based on observations

from numerical simulations, it is here chosen to replace ψ_f with the quantity h_f representing the ratio of the length of the azimuthal layer in contact with the housing and the thickness of the radial layer in contact with the paddle, such that

$$l\psi_f = h_f\left(l - h_0\right). \tag{E.7}$$

Specifically, it is assumed that h_f is independent of the total mass of grain deposited onto the paddle but possibly a function of the sprocket angular velocity.

The area of the circle sector between the paddle and the radial layer given by ψ_f equals

$$A_{sector} = \frac{l^2}{2}\psi_f = h_f\frac{l\left(l - h_0\right)}{2} \tag{E.8}$$

Similarly, the area of the triangular region within this sector that does not contain grain equals

$$A_{triangle} = \frac{h_0 l}{2}\sin\psi_f = \frac{h_0 l}{2}\sin\left(h_f\frac{l - h_0}{l}\right). \tag{E.9}$$

The total cross-sectional area of grain can then be calculated by subtraction

$$A_{grain} = A_{sector} - A_{triangle} = h_f\frac{l\left(l - h_0\right)}{2} - \frac{h_0 l}{2}\sin\left(h_f\frac{l - h_0}{l}\right) \tag{E.10}$$

Appendix F

Equations of motion during the release stage

Consider a cartesian coordinate system represented by the coordinates x and y, and the polar basis vectors $\mathbf{e}_\rho$ and $\mathbf{e}_\theta$, as shown in Figure G.1. The position of a single grain of mass δm is given by

$$\mathbf{r} = x\mathbf{e}_x + y\mathbf{e}_y = \rho\mathbf{e}_\rho$$

where $\mathbf{e}_x$ and $\mathbf{e}_y$ are cartesian basis vectors, such that

$$\mathbf{e}_\rho = \cos\theta\mathbf{e}_x + \sin\theta\mathbf{e}_y \tag{F.1}$$

$$\mathbf{e}_\theta = -\sin\theta\mathbf{e}_x + \cos\theta\mathbf{e}_y \tag{F.2}$$

The velocity of the particle is now given by

$$\mathbf{v} = \frac{d\mathbf{r}}{dt} = \dot{x}\mathbf{e}_x + \dot{y}\mathbf{e}_y = \dot{\rho}\mathbf{e}_\rho + \rho\dot{\mathbf{e}}_\rho = \dot{\rho}\mathbf{e}_\rho + \rho\dot{\theta}\mathbf{e}_\theta, \tag{F.3}$$

since

$$\dot{\mathbf{e}}_\rho = \dot{\theta}\left(-\sin\theta\mathbf{e}_x + \cos\theta\mathbf{e}_y\right) = \dot{\theta}\mathbf{e}_\theta \tag{F.4}$$

In particular,

$$\dot{x} = \dot{\rho}\cos\theta - \rho\dot{\theta}\sin\theta \tag{F.5}$$

$$\dot{y} = \dot{\rho}\sin\theta + \rho\dot{\theta}\cos\theta \tag{F.6}$$

Moreover, the acceleration vector is given by

$$\mathbf{a} = \frac{d^2\mathbf{r}}{dt^2} = \left(\ddot{\rho} - \rho\dot{\theta}^2\right)\mathbf{e}_\rho + \left(2\dot{\rho}\dot{\theta} + \rho\ddot{\theta}\right)\mathbf{e}_\theta, \tag{F.7}$$

since

$$\dot{\mathbf{e}}_\theta = -\dot{\theta}\mathbf{e}_\rho \tag{F.8}$$

In the analysis below, $\dot{\theta}$ is assumed to equal the angular speed of the rotating paddle Ω, such that $\ddot{\theta} = 0$.

Referring again to Figure G.1, the forces acting on an individual grain during this phase are described as

$$\mathbf{f} = (f_2 - f_1 - \mu N_1 + \mu N_2 - \delta m g \sin\theta)\,\mathbf{e}_\rho + (N_1 - N_2 - \delta m g \cos\theta)\,\mathbf{e}_\theta \tag{F.9}$$

where f_1 and f_2 represent interactions with neighboring grains in the same angular sector and N_1 and N_2 represent interactions with neighboring grains in nearby radial layers. From Newton's second law

$$\delta m \mathbf{a} = \mathbf{f} \tag{F.10}$$

it follows that

$$\left(\ddot{\rho} - \rho\Omega^2\right)\delta m = f_2 - f_1 - \mu N_1 + \mu N_2 - \delta m g \sin\theta \tag{F.11}$$

and

$$2\dot{\rho}\Omega\delta m = N_1 - N_2 - \delta m g \cos\theta \tag{F.12}$$

Solving for the net normal force, $N_1 - N_2$, yields:

$$N_1 - N_2 = (2\dot{\rho}\Omega + g\cos\theta)\,\delta m \tag{F.13}$$

Substituting $N_1 - N_2$ into the radial component of the force-balance equation yields

$$\left(\ddot{\rho} - \rho\Omega^2\right)\delta m = f_2 - f_1 - \mu\,(2\dot{\rho}\Omega + g\cos\theta)\,\delta m - \delta m g \sin\theta \tag{F.14}$$

Solving for the radial acceleration, $\ddot{\rho}$, yields the equation of motion

$$\ddot{\rho} = \rho\Omega^2 - 2\dot{\rho}\Omega\mu - g\,(\mu\cos\theta + \sin\theta) + \frac{f_2 - f_1}{\delta m} \tag{F.15}$$

Neglecting interactions between grains in the same radial layer layer, the equations of motion now become

$$\ddot{\rho} = \rho\Omega^2 - 2\dot{\rho}\Omega\mu - g\,(\mu\cos\theta + \sin\theta) \tag{F.16}$$

and, in terms of the non-dimensionalized variables $\tilde{\rho} = \rho/l$, $\tau = \Omega t$,

$$\tilde{\rho}'' = \tilde{\rho} - 2\mu\tilde{\rho}' + \tilde{g}(-\sin\tau - \mu\cos\tau) \tag{F.17}$$

where $\tilde{g} = g/\left(l\Omega^2\right)$.

Appendix G

Figures

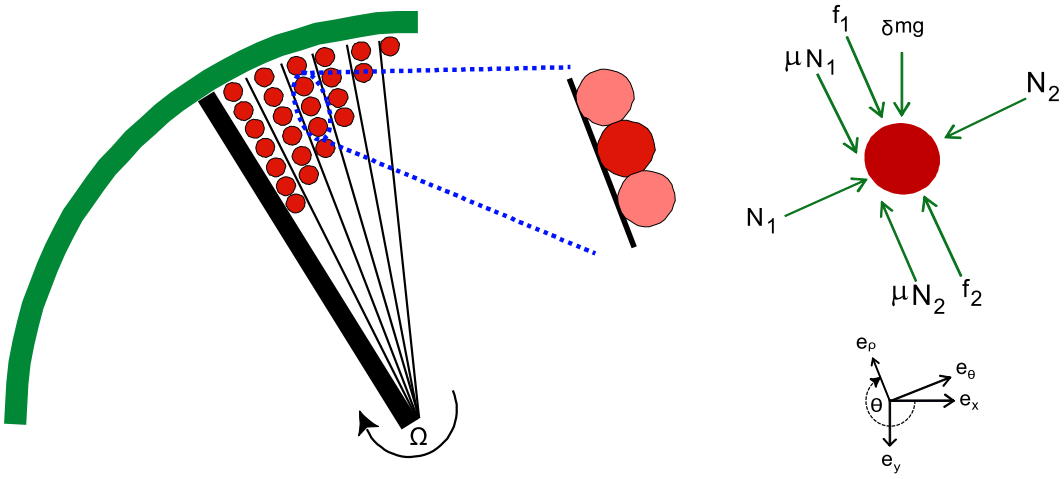

Figure G.1: Free body diagram representing the forces acting on an individual grain.

Appendix H

Approximate traveling time of grains to the tip of the paddle

The traveling time of a grain from its initial position to the tip of the paddle can be numerically computed by solving the nonlinear equation

$$\tilde{\rho}(\tau_d) - 1 = e^{(-\mu\tau_d)} \left(C_1 e^{\left(\sqrt{1+\mu^2}\right)\tau_d} + C_2 e^{\left(-\sqrt{1+\mu^2}\right)\tau_d} \right) - 1 = 0 \tag{H.1}$$

where

$$C_{1,2} = \frac{1}{2} \left(\tilde{\rho}_0 \mp \frac{\tilde{\rho}_0 \mu}{\sqrt{1+\mu^2}} \right) \tag{H.2}$$

In the limit that $\mu = 0$ and, consequently,

$$C_1 = C_2 = \frac{\tilde{\rho}_0}{2}$$

the equation reduces to

$$\tilde{\rho}_0 \cosh \tau_d - 1 = 0$$

such that

$$\tau_d = \cosh^{-1} \frac{1}{\tilde{\rho}_0}.$$

This can now be used as an initial guess for the solution in the case of $\mu \neq 0$.

Appendix I

Discharge velocity of grain from the paddle

The release time t_d is the elapsed time from the moment that the housing constraint is removed on a given radial layer of grains. This occurs when $\theta = 3\pi/2$. It follows that the discharge velocity is given by

$$\begin{aligned} \mathbf{v}_d &= \dot{\rho}\left(t_d\right)\mathbf{e}_\rho + l\Omega\mathbf{e}_\theta|_{\theta=3\pi/2+\Omega t_d} \\ &= \left(\dot{\rho}\left(t_d\right)\cos\left(3\pi/2+\Omega t_d\right) - l\Omega\sin\left(3\pi/2+\Omega t_d\right)\right)\mathbf{e}_x \\ &\quad + \left(\dot{\rho}\left(t_d\right)\sin\left(3\pi/2+\Omega t_d\right) + l\Omega\cos\left(3\pi/2+\Omega t_d\right)\right)\mathbf{e}_y \end{aligned} \tag{I.1}$$

where

$$\dot{\rho}\left(t_d\right) = l\Omega\tilde{\rho}'\left(\tau_d\right) \tag{I.2}$$

and $t_d = \tau_d/\Omega$.

Appendix J

Calculation of the critical radius

The equation to find if a particle hits or misses the impact plate is described as follows:

$$\lambda\left(v_x, v_y\right) = \left(d_x, d_y\right) + \kappa\left(\cos\phi, \sin\phi\right) \tag{J.1}$$

Here, d_x and d_y correspond to the distance from lower most points of the impact plate as related to the origin of paddle rotation. This equation provides two equations: one in the x-direction and one in the y-direction. These equations are shown as follows.

$$l\cos\theta_d + \lambda v_x = d_x + \kappa\cos\phi \tag{J.2}$$

$$l\sin\theta_d + \lambda v_y = d_y + \kappa\sin\phi \tag{J.3}$$

It follows that

$$\lambda = \frac{d_x + \kappa\cos\phi - l\cos\theta_d}{v_x} \tag{J.4}$$

such that

$$l\sin\theta_d + \frac{d_x + \kappa\cos\phi - l\cos\theta_d}{v_x} v_y = d_y + \kappa\sin\phi \tag{J.5}$$

from which

$$\kappa = \frac{d_y v_x - v_x l\sin\theta_d - v_y d_x + v_y l\cos\theta_d}{v_y\cos\phi - v_x\sin\phi} \tag{J.6}$$

and therefore

$$\lambda = \frac{d_x + \frac{d_y v_x - v_x l\sin\theta_d - v_y d_x + v_y l\cos\theta_d}{v_y\cos\phi - v_x\sin\phi}\cos\phi - l\cos\theta_d}{v_x} \tag{J.7}$$

$$= \frac{d_x\sin\phi + l\cos\phi\sin\theta_d - \cos\phi d_y - l\cos\theta_d\sin\phi}{v_x\sin\phi - v_y\cos\phi} \tag{J.8}$$

Appendix K

Momentum lost upon impact

Denote by $\mathbf{v}^-$ and $\mathbf{v}^+$ the velocities of a grain at the instant immediately prior to and immediately following a collision with the impact plate, respectively, such that

$$\mathbf{v}^- = v_x^- \mathbf{e}_x + v_y^- \mathbf{e}_y \tag{K.1}$$

and

$$\mathbf{v}^+ = v_x^+ \mathbf{e}_x + v_y^+ \mathbf{e}_y \tag{K.2}$$

(cf. Figure L.1).

Let

$$\mathbf{n} = -\sin\phi \mathbf{e}_x + \cos\phi \mathbf{e}_y \tag{K.3}$$

denote the unit vector normal to the impact plate. It follows that

$$\mathbf{v} \cdot \mathbf{n} = -v_x \sin\phi + v_y \cos\phi \tag{K.4}$$

equals the speed of the grain perpendicular to the plate. Similarly,

$$\mathbf{n} \times (\mathbf{v} \times \mathbf{n}) = \left(v_x \cos^2\phi + v_y \sin\phi\cos\phi\right)\mathbf{e}_x + \left(v_x \cos\phi\sin\phi + v_y \sin^2\phi\right)\mathbf{e}_y \tag{K.5}$$

is the component of the velocity tangential to the plate.

Assuming that the grain mass is negligible relative to that of the impact plate, using a kinematic coefficient of restitution e for the normal component of the velocity, it follows that

$$\mathbf{v}^+ \cdot \mathbf{n} = -e\mathbf{v}^- \cdot \mathbf{n} \tag{K.6}$$

and

$$\mathbf{n} \times \left(\mathbf{v}^+ \times \mathbf{n}\right) = \mathbf{n} \times \left(\mathbf{v}^- \times \mathbf{n}\right) \tag{K.7}$$

i.e.,

$$-v_x^+ \sin\phi + v_y^+ \cos\phi = -e\left(-v_x^- \sin\phi + v_y^- \cos\phi\right) \tag{K.8}$$

$$v_x^+ \cos^2\phi + v_y^+ \sin\phi\cos\phi = v_x^- \cos^2\phi + v_y^- \sin\phi\cos\phi \tag{K.9}$$

$$v_x^+ \cos\phi\sin\phi + v_y^+ \sin^2\phi = v_x^- \cos\phi\sin\phi + v_y^- \sin^2\phi \tag{K.10}$$

which implies that

$$v_x^+ = \frac{v_x^-\left((1-e)+(1+e)\cos 2\phi\right) + v_y^-\,(1+e)\sin 2\phi}{2} \tag{K.11}$$

$$= v_x^-\left(\cos^2\phi - e\sin^2\phi\right) + v_y^-\,(1+e)\sin\phi\cos\phi \tag{K.12}$$

$$v_y^+ = \frac{v_y^-\left((1-e)-(1+e)\cos 2\phi\right) + v_x^-\,(1+e)\sin 2\phi}{2} \tag{K.13}$$

$$= v_y^-\left(\sin^2\phi - e\cos^2\phi\right) + v_x^-\,(1+e)\sin\phi\cos\phi \tag{K.14}$$

The net loss of grain momentum therefore equals

$$\delta m\left(\mathbf{v}^- - \mathbf{v}^+\right) = \delta m\left[\begin{array}{c}\left(v_x^-\left(1-\cos^2\phi+e\sin^2\phi\right) - v_y^-\,(1+e)\sin\phi\cos\phi\right)\mathbf{e}_x \\ +\left(v_y^-\left(1-\sin^2\phi+e\cos^2\phi\right) - v_x^-\,(1+e)\sin\phi\cos\phi\right)\mathbf{e}_y\end{array}\right] \tag{K.15}$$

$$= \delta m\,(1+e)\left[\left(v_x^-\sin\phi - v_y^-\cos\phi\right)\sin\phi\mathbf{e}_x + \left(v_y^-\cos\phi - v_x^-\sin\phi\right)\cos\phi\mathbf{e}_y\right] \tag{K.16}$$

Appendix L

Figures

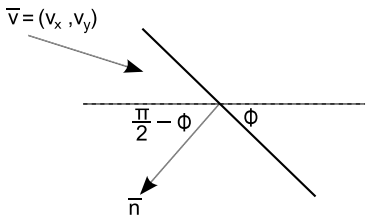

Figure L.1: Velocity vectors of grain before and after impact.

Appendix M

Calculation of the impact force

As previously described, grains with the same initial radial positions will have the same travel time until reaching the distal end of the paddle, and will thus have the same velocity upon release from the paddle, as well as upon impact with the impact plate. All grains located at an initial radial distance $\rho_0 > \rho_{crit}$ therefore lose the same amount of momentum in collisions with the impact plate. It follows that the total momentum lost by grains in a radial layer at radial distance $\rho_0 > \rho_{crit}$ of thickness $d\rho_0$ equals

$$(1+e)\left[\left(v_x^- \sin\phi - v_y^- \cos\phi\right)\sin\phi\mathbf{e}_x + \left(v_y^- \cos\phi - v_x^- \sin\phi\right)\cos\phi\mathbf{e}_y\right]\eta w\rho_0 h^{-1}\left(\rho_0\right)d\rho_0, \tag{M.1}$$

where η is the volume density, w is the paddle width, and $\rho h^{-1}\left(\rho\right)$ equals the arclength. Integrating over initial radial distance, keeping in mind that no collision occur for $\rho_0 < \rho_{crit}$ it follows that the total momentum lost in the horizontal direction due to the collisions with the impact plate of the entire mass deposited on a paddle equals

$$\Delta P = \begin{cases} \int_{\rho_{crit}}^{1}\left(v_x\sin\phi - v_y\cos\phi\right)\sin\phi\left(1+e\right)\rho h^{-1}\left(\rho\right)\eta w d\rho & \rho_{crit} > h_0 \\ \int_{h_0}^{1}\left(v_x\sin\phi - v_y\cos\phi\right)\sin\phi\left(1+e\right)\rho h^{-1}\left(\rho\right)\eta w d\rho & h_0 > \rho_{crit} \end{cases} \tag{M.2}$$

where v_x and v_y are the release velocities for a given initial radial distance ρ.

Appendix N

Matlab code used to generate data using the mathematical model

N.1 Primary function file used to generate data

The main function file used to generate data using the MATLAB realization of the forward computational model is shown below. This file allows the user to define certain geometric and model parameters, and input them to the function file "mymodel_generatedata.m" to perform subsequent calculations.

```
%% run_generatedata.m
% This file generates data using the mathematical model.
% The output is momentum (kg*m/s) transferred to the impact plate given
% inputs of mass flow rate (kg/s) and model parameter values for mu, e,
% density, and hf
%%
clc
clear
%close all
format long
%% Input Mass Flow Rate Data
actual_flow=25.*ones(1,11);
% actual_flow=[];
% for i = 2:0.1:25
%     actual_flow = [actual_flow;i]
% end
impact_plate_angle = [40;45;50;55;60;65;70;75;80;85;90];
%impact_plate_angle = 70.*ones(1,length(actual_flow));
impact_plate_angle = impact_plate_angle.*pi./180; %Convert angle to radians

%% Input parameter values
% mu = 0.2;
% e = 0.6;
% density = 1100;
```

```
% hf = 2;
% k1= 0.5;
% k2=1;
mu = 0.1952;
e = 0.636;
density = 1133;
hf = 2;
k1= 0.5;
k2=1;
%%
predicted_momentum = mymodel_generatedata(actual_flow,mu,e,density,...
    hf,impact_plate_angle,k1,k2)'
%% Plot
impact_plate_angle_plot = impact_plate_angle.*180./pi;
figure
plot(impact_plate_angle_plot,predicted_momentum,'.')
xlabel('Impact Plate Angle (degrees)')
ylabel('Momentum (kg*m/s)')
title('Momentum Transferred to the Impact Plate...
...for Varying ImpactPlate Angles')
legend('12 kg/s','Location','NorthWest')
```

N.2 Secondary function file used to generate data

The secondary function file "mymodel_generatedata.m", shown below, is called by the primary function file, "run_generatedata.m". This secondary file receives user inputs which are used in subsequent calculations of the computational model. Additional subsequent function files are called to complete the calculations required by the four stages of the physical process.

```
%% mymodel_generatedata.m
% This file is called by "run_generateddata" or
% by "run_openloopestimation_generate_data.m" and returns predicted momentum
% given inputs of mass flowrate and model parameters
%%
 function predicted_momentum=mymodel_generatedata(actual_flow,mu,e,density,...
     height_fraction,impact_plate_angle,k1,k2)
%% Machine Data
sprocket_speed = 400; %Rotations per minute
```

```
omega=2*pi*sprocket_speed/60; %Rotational speed of paddles (radians/sec)
sprocket_radius = 0.0541; % Pitch Radius of sprocket (8 teeth) - Units of meters
%There is also a high capacity sprocket with 10 teeth and a pitch radius of 67
%mm
%paddle_frequency = 34/2.6; %Number of paddles per second
paddle_width = 0.21;  %width of paddle (meters)
paddle_distance = 0.18635; %distance between paddles (meters) -
                           %alternates between 0.1656 meters and 0.207 meters;
                           %Here, an average distance is used
paddle_length = 0.16; %Paddle length (meters)
linear_velocity = (omega*sprocket_radius);
paddle_frequency = linear_velocity/paddle_distance; % Units of paddles/second
%% Display Parameters on each iteration
%params
%% Define simulation constants
rho_step_size_hat = 0.0005; %Represents non-dimensionalized size of
                            %incremental radial positions of curved rows
rho_step_size = rho_step_size_hat*paddle_length; %Represents dimensionalized
                            %size of incremental radial positions of curved rows
%% HEIGHT METHOD
h0_save=zeros(length(actual_flow),1); %h0(phi=0)=h0_0
for i=1:length(actual_flow)
    %paddle_mass(i) = actual_flow(i)./paddle_frequency;
    paddle_mass(i) = actual_flow(i).*paddle_distance./linear_velocity;
    paddle_area(i) = paddle_mass(i)./ (density*paddle_width);
    % Triangular area of grain on a paddle (m^2) - from a side view
    xa = 0;
    xb = paddle_length;        % search interval
    x_error=100;
    while (x_error >0.00000001)
        % mid point of interval
        xtest = xa + (xb-xa)/2;
        % left-interval function value
        fa = ( (paddle_length/2).*height_fraction.*(paddle_length - xa) ) -...
            ( (0.5).*xa.*paddle_length.*sin(height_fraction.*...
            (paddle_length-xa)./paddle_length) ) -paddle_area(i);
        % right-interval function value
        fb = ( (paddle_length/2).*height_fraction.*(paddle_length - xb) ) -...
            ( (0.5).*xb.*paddle_length.*sin(height_fraction.*...
            % mid-point function value
            (paddle_length-xb)./paddle_length) ) -paddle_area(i);
```

```
        ftest = ( (paddle_length/2).*height_fraction.*(paddle_length - xtest) ) -...
            ( (0.5).*xtest.*paddle_length.*sin(height_fraction.*...
            (paddle_length-xtest)./paddle_length) ) -paddle_area(i);
        if sign(fa)*sign(ftest)<0       % if zero in left half
            xb = xtest;                 %   take left half of interval
        elseif sign(ftest)*sign(fb)<0   % if zero in right half
            xa = xtest;                 %   take right half of interval
        elseif ftest ==0                % if zero at mid-point
            break                       %   this is the zero
        else                            %
            % may have no zero or multiple zeros
            error('multiple roots or no root')
        end
        x_error = abs(ftest-0); %perform error calculation between midpoint...
                                %value and desired value
    end
    h0_save(i)=xtest; %Initial height of grain on a paddle (meters)
end
%% Non-dimensionalization of initial grain heights, h0
h0_hat=h0_save./paddle_length; % dimensionless inital heights, h0
%% Evaluate discharge velocities, time to discharge, and radial position...
%of each row of grains
% These quantities are dimensionless
[vx_hat, vy_hat,tau_d_save, rho0_hat_save] = v_hat( mu, rho_step_size_hat);
%% Rescaling procedure
theta_d = (3*pi/2) + tau_d_save;    %Dimensionalized paddle travel angle...
                                    %until discharge (radians)
vx = vx_hat.*(paddle_length*omega); %Dimensionalized discharge x-velocity
vy = vy_hat.*(paddle_length*omega); %Dimensionalized discharge y-velocity
rho = rho0_hat_save.*paddle_length; %Dimensionalized incremental values of rho -
                                    %represents radial position of each row
                                    %of grains
%% Evaluate critical radius for each flow rate and its vector index value
for i = 1:length(impact_plate_angle)
    phi = impact_plate_angle(i);
    dx = 0.4 + 0.1887*cos(phi);
    dy = 0.1887*sin(phi) - 0.16;
    %critical radius between grains with impact and without impact
    [rho_cr,iteration_rho_cr]=critical_radius(mu, vx', vy', theta_d,...
        rho, phi, dx, dy, paddle_length);
    critical_radius_location(i) = rho_cr;
```

```
    iteration_critical_radius(i) = iteration_rho_cr;
end
%% Evaluate velocity*mass in a curved row
for i = 1:length(actual_flow)
    rho_cr = critical_radius_location(i);
    iteration_rho_cr = iteration_critical_radius(i);
    [vx_dm_A_save_temp,vy_dm_A_save_temp]=mv(h0_save, vx', vy', rho_cr,...
        rho_step_size,rho', iteration_rho_cr,actual_flow,paddle_length,...
        density,paddle_width,height_fraction); %evaluate mv_hat
    vx_dm_A_save(i) = vx_dm_A_save_temp(i);
    vy_dm_A_save(i) = vy_dm_A_save_temp(i);
    size(vx_dm_A_save)
end
%% Evalutate change in momentum in x-direction
% only x-direction momentum is needed because sensor only measures in
% x-direction. Note: this still includes x and y velocities
% NOTE: the term "(1+e)" has been replaced by "e" on Aug.21,2009
% to account for relationship between change in momentum and force, which
% are proportional.
full_paddle_mass = (height_fraction/2)*density*paddle_width*...
    (paddle_length^2);
for i = 1:length(vx_dm_A_save)
    predicted_momentum(i) = ( vx_dm_A_save(i).*sin(impact_plate_angle(i))-...
        vy_dm_A_save(i).*cos(impact_plate_angle(i)) ).*...
        sin(impact_plate_angle(i)).*(1+e);
end
```

N.3 Function file for computation of grain discharge velocities

The tertiary function file "v_hat.m", shown below, is called by the secondary function file. This tertiary function file computes the velocity vectors of grains at the conclusion of the release stage. This file also calls two additional function files, "tauf.m" and "velocity.m", which compute the travel time of a grain to the distal end of the paddle, and the radial velocity of grains upon discharge from the paddle, respectively.

```
%% v_hat.m
% This file is called by mymodel.m and returns non-dimensionalized...
% velocities, travel time of the grain from to the tip of the paddle,...
```

```
% and incremental radial positions along the paddle
function [vx_hat, vy_hat, tau_d_save, rho0_hat_save]=v_hat(mu,...
    rho_step_size_hat)
%% Initial value for rho0_hat
rho0_hat=rho_step_size_hat;
%% Pre-allocate matricies
tau_d_save = zeros(2000,1);
rho0_hat_save = zeros(2000,1);
vx_hat = zeros(1,2000);
vy_hat = zeros(1,2000);
%% Evaluate dimensionless travel time to tip of paddle
i=1;
while (rho0_hat<1.00)
    rho0_hat_save(i)=rho0_hat; %Non-dimensionalized radial position of each
                               %row of grains
    xa = 0;
    xb = 1000;        % search interval
    x_error=100;
    first_iter=1;
    while (x_error >0.000001)           % loop
        xtest = xa + (xb-xa)/2;         % mid point of interval
        fa = tauf(xa, mu, rho0_hat);        % left-interval function value
        fb = tauf(xb, mu, rho0_hat);        % right-interval function value
        ftest = tauf(xtest, mu, rho0_hat);  % mid-point function value
        %For each value of rho0_hat, the time of flight of grains at the
        %current value of rho0_hat is close to the value of the time of
        %flight for grains at the previous value of rho0_hat - the
        %following section of code therefore uses partial derivatives to
        %obtain a closer initial guess for taud using the previous value of
        %taud
        %-------------------------
        %if this is the firstiteration for this valueof rho and this is...
        %not the lowest value of rho
        if (first_iter == 1 && rho0_hat~=rho_step_size_hat)
            dF_drho = 0.5*(1-(mu/sqrt(1+(mu^2))))*...
                exp((-mu-sqrt(1+(mu^2)))*tau_d) + 0.5*...
                (1+(mu/sqrt(1+(mu^2))))*exp((-mu+sqrt(1+(mu^2)))*tau_d);
            dF_dtaud = 0.5*(rho0_hat-((rho0_hat*mu)/sqrt(1+(mu^2))))*...
                exp((-mu-sqrt(1+(mu^2)))*tau_d) + 0.5*...
                (rho0_hat+((rho0_hat*mu)/sqrt(1+(mu^2))))*...
                exp((-mu+sqrt(1+(mu^2)))*tau_d);
```

```
            taud_prime = -dF_drho/dF_dtaud;
            xtest = tau_d + rho_step_size_hat*taud_prime;
            ftest = tauf(xtest, mu, rho0_hat);
        end
        if sign(fa)*sign(ftest)<0       % if zero in left half
            xb = xtest;                 %   take left half of interval
        elseif sign(ftest)*sign(fb)<0   % if zero in right half
            xa = xtest;                 %   take right half of interval
        elseif ftest ==0                % if zero at mid-point
            break                       %   this is the zero
        else                            %
            % may have no zero or multiple zeros
            error('multiple roots or no root')
        end
        first_iter=0;
        x_error = abs(ftest-0);
    end
    tau_d=xtest;
    tau_d_save(i)=tau_d;
    %tau_d_save=[tau_d_save; tau_d];
    rho0_hat=rho0_hat+rho_step_size_hat;
    i=i+1;
end
%% Evaluate radial discharge velocity
vr_hat=velocity(tau_d_save, mu, rho0_hat_save);
%% Evaluate discharge velocities in x and y directions
%dimensionless velocity in x-direction; vx_hat=vx /l/omega
vx_hat=[(vr_hat.*cos(tau_d_save + (3*pi/2)) - sin(tau_d_save + (3*pi/2)))'];
%dimensionless velocity in y-direction; vy_hat=vy/l/omega
vy_hat=[(vr_hat.*sin(tau_d_save + (3*pi/2)) + cos(tau_d_save + (3*pi/2)))'];
```

N.4 Function file for computation of grain travel times

The quaternary function file "tauf.m", shown below, is called by the tertiary function file "v_hat.m". This file computes the travel times of grains from their initial location on the paddle to the distal end of the paddle.

```
%% tauf.m
% This file is called by v_hat.m and returns the travel time for a grain
```

```
% from its initial position to the tip of the paddle
function f=tauf(taud, mu, rho0_hat) %tauf=travel time from r0 to l
C1 = (1/2)*(rho0_hat - ( (rho0_hat*mu) / sqrt(1+(mu^2)) ) );
C2 = (1/2)*(rho0_hat + ( (rho0_hat*mu) / sqrt(1+(mu^2)) ) );
%Exponential that C1 is multiplied by
C1exp = exp((-mu - sqrt(1 + mu^2))* taud);
%Exponential that C2 is multiplied by
C2exp = exp((-mu + sqrt(1 + mu^2))* taud);
f = C1*C1exp + C2*C2exp - 1;
```

N.5 Function file for computation of radial velocities of grains

The quaternary function file "velocity.m", shown below, is called by the tertiary function file "v_hat.m". This file computes the radial velocities of grains upon their discharge from the paddle.

```
%% velocity.m
% this file is called by v_hat.m and returns the radial velocity of a
% grain upon reaching the tip of the paddle
function [vr_hat]=velocity(taud, mu, rho0_hat_save)
C1 = ( (1/2).*(rho0_hat_save - ( (rho0_hat_save.*mu) ./...
    sqrt(1+(mu^2)) ) ) )';
C2 = ( (1/2).*(rho0_hat_save + ( (rho0_hat_save.*mu) ./...
    sqrt(1+(mu^2)) ) ) )';
%Exponential that C1 is multiplied by
C1exp = exp((-mu - sqrt(1 + mu^2)).* taud);
%Exponential that C2 is multiplied by
C2exp = exp((-mu + sqrt(1 + mu^2)).* taud);
vr_hat = (C1'.*C1exp).*( -mu - sqrt(1+(mu^2)) ) + (C2'.*...
    C2exp).*(-mu + sqrt(1+(mu^2)));
```

N.6 Function file for computation of the critical radius

The tertiary function file "critical_radius.m", shown below, is called by the secondary function file. This file computes the critical radius of grains on the paddle at the conclusion of the settling stage.

```
%% critical_radius.m
% This file is called by mymodel.m and returns the critical radius and the
```

```
% location of the critical radius within the Matlab code
function [rho_cr,iteration_rho_cr]=critical_radius(mu, vx, vy, theta_d,...
    rho,phi, dx, dy, paddle_length) %critical radius between grains with...
                                    %impact and without impact
%Note - dx and dy have dimension
%% Preallocate vector
kappa=zeros(length(theta_d),1);
%% Calcuate value of kappa for each row of grains
% kappa = 0 corresponds to the critical radius
% kappa < 0 corresponds to grains that will hit the plate
% kappa > 0 corresponds to grains that will miss the plate
[kappa, lambda] = yf_zero(theta_d, phi, vx, vy, dx, dy,...
    paddle_length); %Compute yf=0
%% Find theta_d greater than 2*PI
% theta_d is sorted from largest to smallest
low = length(theta_d);
high = 1;
while( low > high+1)  %Stop when the elements are separated by only 1 index
   mid =ceil((low+high)/2); %If (low+high)/2 is a decimal then round up
   if theta_d(mid) > 2*pi
       high = mid;
   elseif theta_d(mid) < 2*pi
       low = mid;
   else
       period_iter = mid;
       break
   end
end
period_iter=mid;
if theta_d(mid)>2*pi    %if index is on the high side of...
    %the border then add one
    period_iter=period_iter+1;
end
%% Find the region where lambda is positive (for grains nearest paddle tip)
% only searching in the interval where theta_d is less than 360 degrees
low = period_iter;
high = length(lambda);
while( high > low+1)  %Stop when the elements are separated by only 1 index
    mid = ceil((low+high)/2);  %If (low+high)/2 is a decimal then round up
    if lambda(mid) > 0
        high = mid;
```

```
    elseif lambda(mid)<0
        low = mid;
    else
        lambda_iter = mid;
        break
    end
end
lambda_iter = mid;
if lambda(mid)<0   %if index is on the high side of the border then add one
    lambda_iter=lambda_iter+1;
end
%% Find the critical radius and its index location within the vector
% Uses a binary seach algorithm
low = lambda_iter; %Index of first matrix element - ...
%grains nearest center of paddle rotation
high = length(kappa); %Index of last matrix element - nearest paddle tip
 while ( high > low+1)%Stop when the elements are separated by only 1 index
     mid = ceil((low + high)/2);%If (low+high)/2 is a decimal then round up
    if kappa(mid) < 0
        high = mid;
    elseif kappa(mid) > 0
        low = mid;
    else
        iteration_rho_cr = mid;
        break;
    end
 end
iteration_rho_cr = mid;
if kappa(mid)>0
    iteration_rho_cr = mid+1;
end
rho_cr=rho(iteration_rho_cr);
```

N.7 Function file to facilitate the computation of the critical radius

The function file "yf_zero.m", shown below, is called by the function file "critical_radius.m". This file computes the values of κ and λ for computation of the critical radius.

```
%% yf_zero.m
% This file is called by critical_radius.m and returns the values of kappa
% and lambda, which indicate if the grain will hit or miss the impact plate
function [kappa, lambda]=yf_zero(theta_d, phi, vx, vy, dx, dy,...
    paddle_length)
%% Calcuate value of kappa for each row of grains
% kappa = 0 corresponds to the critical radius
% kappa < 0 corresponds to grains that will hit the plate
% kappa > 0 corresponds to grains that will miss the plate
kappa = ( dy.*vx - vx.*paddle_length.*sin(theta_d) - dx.*vy + vy.*...
    paddle_length.*cos(theta_d) ) ./ ( vy.*cos(phi) - vx.*sin(phi) );
lambda = ( dx + kappa.*cos(phi) - paddle_length.*cos(theta_d) )./ vx;
```

N.8 Function file to compute the momentum of the bulk mass of grains

The function file "mv.m", shown below, is called by the function file "mymodel_generatedata.m". It computes the momentum for the entire bulk mass of grains after release from the elevator paddle.

```
%% mv.m
% This file is called by mymodel.m and returns the momentum transferred to
% the impact plate in the x-direction and in the y-direction
function [vx_dm_A_save,vy_dm_A_save]= mv(h0, vx, vy, rho_cr, rho_step_size,...
    rho, iteration_rho_cr,actual_flow,paddle_length,density,paddle_width,...
    height_fraction)
%h0 represents initial heights of grain for each flowrate (m)
%vx represents dimensionalized discharge x-velocity (m/s)
%vy represents dimensionalized discharge y-velocity (m/s)
%rho represents dimensionalized incremental values of rho which represents ...
%radial position of each row of grains
%% Pre-allocate vectors
%h0_save=[];
vx_save=zeros(length(actual_flow),1);
vy_save=zeros(length(actual_flow),1);
%% Calculate velocity*mass for each curved row
% velocity*mass is calculated here because there are unique values of
% mass and velocities for each curved row, and this combination must be
% kept track of
```

```
iii=iteration_rho_cr;
for i=1:length(h0)
    psi_f = height_fraction*(paddle_length - h0(i))/paddle_length;
    for j=1:length(rho)
        xa = 0;
        xb = psi_f;        % search interval
        x_error=100;
        while (x_error >0.000001)
            if (rho(j)<=h0(i))  %If rho is less than the initial ...
                %height then psi is zero
                xtest=0;
                break
            end
            xtest = xa + (xb-xa)/2;         % mid point of interval
            %The following equations can be found in the appendix of the
            %paper describing the mathematical model
            % left-interval function value
            fa = ( 1 / (    (cos(xa)/h0(i)) + ( ((1/paddle_length)-...
                (cos(psi_f)/h0(i))) / sin(psi_f)*sin(xa) ) ) ) - rho(j);
            % right-interval function value
            fb = ( 1 / (    (cos(xb)/h0(i)) + ( ((1/paddle_length)-...
                (cos(psi_f)/h0(i))) / sin(psi_f)*sin(xb) ) ) ) - rho(j);
            % mid-point function value
            ftest = ( 1 / (    (cos(xtest)/h0(i)) + ( ((1/paddle_length)-...
                (cos(psi_f)/h0(i))) / sin(psi_f)*sin(xtest) ) ) ) - rho(j);
            if sign(fa)*sign(ftest)<0       % if zero in left half
                xb = xtest;                 %   take left half of interval
            elseif sign(ftest)*sign(fb)<0   % if zero in right half
                xa = xtest;                 %   take right half of interval
            elseif ftest ==0                % if zero at mid-point
                break                       %   this is the zero
            else                            %
                error('multiple roots or no root')  % may have...
                %no zero or multiple zeros
            end
            x_error = abs(ftest-0); %perform error calculation between...
                                    %midpoint value and desired value
        end
        psi(j)=xtest; % Azimuthal angle for each value of rho
    end
    dm = (rho_step_size.*rho.*psi.*density.*paddle_width)'; % Mass of...
```

```
    %grain in a curved row (kg) – Note that the arc length here...
    %(rho*psi) is equivalent to the expression for arc length
    %as given in the paper describing the mathematical model. In the paper
    %the expression for arclength is rho * the inverse function of h(rho)
    %Note: that expression is not equal to 1/h(rho)
    if rho_cr<h0(i)  % h0_hat(i)>rho_cr => integratae from h0_hat...
        %to paddle_length_hat
        %Find the index location of rho that corresponds to h0
       low = 1;
       high = length(rho);
       while (low < high–1)
           mid = round((low + high)/2);
           if h0(i) > rho(mid)
               low = mid;
           elseif h0(i)<rho(mid)
               high = mid;
           else
               ii = mid;
               break;
           end
       end
       ii = mid;
       %All grains in a curved row will have the same velocity
       %Sum (velocity*massofgrain) for an entire curved row of grain
       vx_dm_A=sum(vx(ii:end).*dm(ii:end));
       vy_dm_A=sum(vy(ii:end).*dm(ii:end));
       vx_dm_A_save(i)=vx_dm_A;
       vy_dm_A_save(i)=vy_dm_A;
    else     %h0_hat(i)<rho_cr => integrate from rho_cr to l_hat
       vx_dm_A=sum(vx(iii:end).*dm(iii:end));
       vy_dm_A=sum(vy(iii:end).*dm(iii:end));
       vx_dm_A_save(i)=vx_dm_A;
       vy_dm_A_save(i)=vy_dm_A;
    end
end
```

Appendix O

Matlab code used in the system calibration estimation process

O.1 Primary function file used in the system calibration process

The function file "run_calibration_generated_data.m" is the primary function file used in the system calibration process. This file defines pairs of mass flow rate and momentum imparted to the impact plate, as well as machine geometric parameters. It then calls the *lsqnonlin* function and a subsequent function file, "mymodel_calibration_generated_data.m', to facilitate the calibration process.

```
%% run_calibration_generated_data.m
% This file performs "System Calibration" and returns estimates for
% internal model parameters given inputs of mass flow rate and measured
% momentum values
tic
clc
clear
close all
old_resnorm = 10000;
mu_save=[10000];
e_save=[10000];
density_save=[10000];
hf_save=[10000];
max_error=[100000];
avg_error=[100000];
angle=[40 45 50 55 60 65 70 75 80 85 90];
actual_flow = [2    5   9   12  15  18  21  25];
% The following data is momentum values. The ROWS are impact plate angles
% from 40 degrees to 90 degrees.  The COLUMNS are for mass flow rates of
% 2, 5, 9, 12, 15, 18, 21, 25 kg/s
data_summary=[0.269242  0.591450    0.874724    1.042850    1.190011...
```

```
    1.323006    1.445756    1.597702
            0.329125    0.757577    1.141269    1.367851    1.565763...
            1.744381    1.909080    2.112784
            0.389532    0.928932    1.428843    1.722274    1.977972...
            2.208398    2.420644    2.682924
            0.448628    1.096114    1.725491    2.092243    2.411001...
            2.697793    2.961668    3.287450
            0.504616    1.250340    2.022145    2.467735    2.853863...
            3.200653    3.519351    3.912433
            0.555795    1.384641    2.307490    2.835657    3.291768...
            3.700604    4.075834    4.538161
            0.600611    1.502836    2.566531    3.176552    3.701369...
            4.170824    4.601121    5.130743
            0.637702    1.602100    2.793051    3.483944    4.075710...
            4.603841    5.087230    5.681533
            0.665940    1.679416    2.977798    3.746457    4.401344...
            4.984311    5.517061    6.171274
            0.684469    1.732436    3.110760    3.948516    4.658031...
            5.287949    5.862708    6.567683
            0.692724    1.759548    3.188131    4.083337    4.836486...
            5.503339    6.110867    6.855199];
%%
%measured_momentum = data_summary(7,(1:8));
measured_momentum = [1.601630   4.007562    6.844082    8.470806...
    9.870317    11.122196   12.269657   13.681981];
%impact_plate_angle = angle(7);
impact_plate_angle=70
%% Perform curvefitting using Kentucky Data
successful_random_parameters = 0;
for i=1:100 % Number of initial guesses to try in lsqnonlin
    while(successful_random_parameters == 0)
        mu_guess = 0 + (1-0).*rand(1);
        e_guess = 0 + (2-0).*rand(1);
        density_guess = 0.75 + (2-0.75).*rand(1);
        %hf_guess = 2.4 + (3.1-2.4).*rand(1);
        hf_guess = 2;
        for j=1:length(actual_flow)
            MassOverEtaWLsquared(j) = ( actual_flow(j)*0.18635/( (2*...
                pi*400/60) *0.0541) )/(density_guess*1000*0.21*...
                (0.16^2));  %paddle_width=0.18635 and paddle_length=0.127
        end
```

```
        for j=1:length(actual_flow)
            if (MassOverEtaWLsquared(j) < (hf_guess/2)) &&...
                    (MassOverEtaWLsquared(j) > 0)
                successful_random_parameters = 1;
            else
                successful_random_parameters = 0;
                break
            end
        end
    end
    successful_random_parameters = 0;
    mu_guess_save(i) = mu_guess;
    e_guess_save(i) = e_guess;
    density_guess_save(i) = density_guess;
    hf_guess_save(i) = hf_guess;
%%
    mu_e_density_guess=[.18 .6 0.917];
    %mu_e_density_guess=[mu_guess e_guess density_guess];
    options=optimset('TolFun',1E-6,'TolX',1E-3,'DiffMaxChange',5,...
        'DiffMinChange',0.001);
    lb=[0 0 0.750];
    ub=[1 2 2.000];
    [Estimates,resnorm,residual]=lsqnonlin(@...
        mymodel_calibration_generated_data, mu_e_density_guess,lb,ub,...
        options, actual_flow, measured_momentum,impact_plate_angle);
    resnorm_history(i)=resnorm;
    residual_history(i,1:length(residual)) = residual;
    max_error_history(i)=100*max(abs(residual./measured_momentum));
    avg_error_history(i)=100*mean(abs(residual./measured_momentum));
    %% Convert parameters back to original
    mu = Estimates(1)
    e = Estimates(2)
    density = Estimates(3)*1000
    %hf = Estimates(4)
    mu_history(i)=mu;
    e_history(i)=e;
    density_history(i)=density;
    %hf_history(i)=hf;
%% Calculate Errors
    if resnorm==old_resnorm
        mu_save(length(mu_save)+1) = mu
```

```
        e_save(length(e_save)+1) = e
        density_save(length(density_save)+1) = density
        %hf_save(length(hf_save)+1) = hf
        max_error(length(max_error+1)) = 100*max(abs(residual./...
            measured_momentum))
        avg_error(length(avg_error+1)) = 100*mean(abs(residual./...
            measured_momentum))
        old_resnorm = resnorm;
    elseif  resnorm<old_resnorm
        mu_save(length(mu_save)) = mu
        e_save(length(e_save)) = e
        density_save(length(density_save)) = density
        %hf_save(length(hf_save)) = hf
        max_error(length(max_error)) = 100*max(abs(residual./...
            measured_momentum))
        avg_error(length(avg_error)) = 100*mean(abs(residual./...
            measured_momentum))
        old_resnorm = resnorm;
    end
    i=i+1
end
toc
summary=[resnorm_history' max_error_history' avg_error_history'...
    mu_history' e_history' density_history']
```

O.2 Secondary function file used in the system calibration process

The function file "mymodel_calibration_generated_data.m", shown below, is called by the primary function file, "run_calibration_generated_data.m". This secondary file receives user inputs which are used in subsequent calculations of the computational model. Additional subsequent function files (see Appendix N) are called to complete the calculations required by the four stages of the physical process.

```
%% mymodel_calibration_generated_data.m
% This file is called by "run_calibration_generated_data.m" and returns the
% error in the predicted momentum
function Error_Vector=mymodel_calibration_generated_data(params,...
```

```
    actual_flow, measured_momentum, impact_plate_angle)
%function sse=mymodel(params, actual_flow, measured_momentum) %params=[mu,
%e], l=paddle length, density=bulk density
%% Parameters
mu=params(1);   %friction coefficient
e=params(2);    %efficiency of impact
density=params(3)*1000;
%height_fraction=params(4);
height_fraction = 2;
phi = impact_plate_angle.*pi./180;
dx = 0.4 + 0.1887*cos(phi);
dy = 0.1887*sin(phi) - 0.16;
%% Machine Data
sprocket_speed = 400; %Rotations per minute
omega=2*pi*sprocket_speed/60; %Rotational speed of paddles (radians/sec)
sprocket_radius = 0.0541; %Pitch Radius of sprocket (8 teeth)...
%-Units of meters
%There is also a high capacity sprocket with 10 teeth...
%and a pitch radius of 67mm
%phi=58*pi/180; %Impact plate angle (radians) measured CW from the x-axis
%paddle_frequency = 34/2.6; %Number of paddles per second
paddle_width = 0.21;  %width of paddle (meters)
paddle_distance = 0.18635; %distance between paddles (meters) -...
%alternates between 0.1656 meters and 0.207 meters; Here, an...
%average distance is used
paddle_length = 0.16; %Paddle length (meters)
%dx = 0.5; %Horizontal distance between center of paddle rotation...
%and lower most point of plate - has dimension
%dy = 0; %Vertical distance between center of paddle rotation and...
%lower most point of plate - has dimension
linear_velocity = (omega*sprocket_radius);
paddle_frequency = linear_velocity/paddle_distance; %paddles/second
%% Display Parameters on each iteration
%params
%% Define simulation constants
rho_step_size_hat = 0.0005; %Represents non-dimensionalized...
%size of incremental radial positions of curved rows
rho_step_size = rho_step_size_hat*paddle_length; %Represents...
%dimensionalized size of incremental radial positions of curved rows
%% HEIGHT METHOD
h0_save=zeros(length(actual_flow),1); %h0(phi=0)=h0_0
```

```
bisect_error=0;
% a flag to determine if an error occured during bisection algorithm
for i=1:length(actual_flow)
    %paddle_mass(i) = actual_flow(i)./paddle_frequency;
    paddle_mass(i) = actual_flow(i).*paddle_distance./linear_velocity;
    paddle_area(i) = paddle_mass(i)./ (density*paddle_width);
    xa = 0;
    xb = paddle_length;        % search interval
    x_error=100;
    while (x_error >0.0000000001)
        xtest = xa + (xb-xa)/2;         % mid point of interval
        fa = ( (paddle_length/2).*height_fraction.*...
            (paddle_length - xa) ) - ( (0.5).*xa.*paddle_length.*...
            sin(height_fraction.*(paddle_length-xa)....
            /paddle_length) ) -paddle_area(i);%left-interval function value
        fb = ( (paddle_length/2).*height_fraction.*...
            (paddle_length - xb) ) - ( (0.5).*xb.*paddle_length.*...
            sin(height_fraction.*(paddle_length-xb)./paddle_length) )...
            -paddle_area(i);          % right-interval function value
        ftest = ( (paddle_length/2).*height_fraction.*(paddle_length -...
            xtest) ) - ( (0.5).*xtest.*paddle_length.*...
            sin(height_fraction.*(paddle_length-xtest)./paddle_length) )...
            -paddle_area(i); % mid-point function value
        if sign(fa)*sign(ftest)<0       % if zero in left half
            xb = xtest;                 %   take left half of interval
        elseif sign(ftest)*sign(fb)<0   % if zero in right half
            xa = xtest;                 %   take right half of interval
        elseif ftest ==0                % if zero at mid-point
            break                       %   this is the zero
        else                            %
            bisect_error=1;
            break
            %error('multiple roots or no root')  % may have no zero or
            %multiple zeros
        end
        x_error = abs(ftest-0); %perform error calculation between...
        %midpoint value and desired value
    end
    h0_save(i)=xtest; %Initial height of grain on a paddle (meters)
end
%% Non-dimensionalization of initial grain heights, h0
```

```
h0_hat=h0_save./paddle_length; % dimensionless inital heights, h0
%% Evaluate discharge velocities, time to discharge,...
%and radial position of each row of grains
% These quantities are dimensionless
[vx_hat, vy_hat,tau_d_save, rho0_hat_save] = v_hat( mu,...
    rho_step_size_hat);% evaluate v_hat
%% Rescaling procedure
theta_d = (3*pi/2) + tau_d_save;    %Dimensionalized...
%paddle travel angle until discharge (radians)
vx = vx_hat.*(paddle_length*omega); %Dimensionalized discharge x-velocity
vy = vy_hat.*(paddle_length*omega); %Dimensionalized discharge y-velocity
rho = rho0_hat_save.*paddle_length; %Dimensionalized incremental...
%values of rho - represents radial position of each row of grains
%% Evaluate critical radius for each flow rate and its vector index value
[rho_cr,iteration_rho_cr]=critical_radius(mu, vx', vy', theta_d,...
    rho, phi, dx, dy, paddle_length); %critical radius between grains...
%with impact and without impact
%% Evaluate velocity*mass in a curved row
[vx_dm_A_save,vy_dm_A_save]=mv(h0_save, vx', vy', rho_cr,...
    rho_step_size, rho', iteration_rho_cr,actual_flow,paddle_length,...
    density,paddle_width,height_fraction); %evaluate mv_hat
%% Evaluate change in momentum in x-direction
% only x-direction momentum is needed because sensor only measures in
% x-direction. Note: this still includes x and y velocities
% NOTE: the term "(1+e)" has been replaced by "e" on Aug.21,2009
% to account for relationship between change in momentum and force, which
% are proportional.
predicted_momentum = ( vx_dm_A_save.*sin(phi)-vy_dm_A_save.*cos(phi) )....
    *sin(phi).*(1+e);
%% Evaluate predicted force
% NOTE: This was commented out on Oct.5, 2009 becuase the sensor really
% captures momentum rather than force
%predicted_force = paddle_frequency.*momentum;
%predicted_force = predicted_force';
%% Compare predicted force with measure experimental force
%Error_Vector = predicted_force - measured_force;
Error_Vector = predicted_momentum - measured_momentum;
if bisect_error==1
    Error_Vector=10000000.*measured_momentum;
    bisect_error=0;
end
```

```
%% PLOT
hold off
figure(1), plot( actual_flow, predicted_momentum,'xr'),...
    title('Momentum Imparted to the Impact Plate for Varying...
Rates of Mass Flow')
hold on
plot(actual_flow,measured_momentum,'.')
legend('Estimated Data','Generated Data','Location','Northwest')
xlabel('Flow Rate (kg/s)')
ylabel('Momentum (kg*m/s)')
```

Appendix P

Matlab code used in the open loop estimation process

The function file "run_openloopestimation_generated_data.m" is the primary function file used in the open-loop estimation process. This file defines mass flow rates and model parameters for computation of momentum imparted to the impact plate. It then calls a subsequent function file (cf. Appendix N) to facilitate the open-loop estimation process.

```
%% run_openloopestimation_generated_data.m
% This file performs "Open Loop Estimation".  It estimates mass flow rate
% given values for measured force and parameter estimates (obtained from
% system calibration)
clc
clear
close all
%% Input estimated parameter values
mu = 0.19;
e = 0.60;
density = 1084;
hf = 2;
%angle=[40 45 50 55 60 65 70 75 80 85 90];
flow_rate = [2  5   9   12  15  18  21  25];
% The following data is momentum values. The rows are impact plate angles
% from 40 degrees to 90 degrees.  The columns are for mass flow rates of
% 2, 5, 9, 12, 15, 18, 21, 25 kg/s
% NOTE THE DATA IS TRANSPOSED FROM THE FORMAT NORMALLY USED
data_summary=[0.72      1.58    2.33    2.78    3.17    3.53    3.86 4.26
              0.88    2.02    3.04    3.65    4.18    4.65    5.09 5.63
              1.04    2.48    3.81    4.59    5.27    5.89    6.46 7.15
              1.2     2.92    4.6     5.58    6.43    7.19    7.9  8.77
              1.35    3.33    5.39    6.58    7.61    8.54    9.38 10.43
              1.48    3.69    6.15    7.56    8.78    9.87    10.87 12.1
              1.6     4.01    6.84    8.47    9.87    11.12   12.27 13.68
```

```
                1.7     4.27    7.45    9.29    10.87   12.28   13.57 15.15
                1.78    4.48    7.94    9.99    11.74   13.29   14.71 16.46
                1.83    4.62    8.3     10.53   12.42   14.1    15.63 17.51
                1.85    4.69    8.5     10.89   12.9    14.68   16.3 18.28
                ];
%% Flow Estimates
% Use bisection algorithm to estimate mass flow
%impact_plate_angle = [70,70,70,70,70,70,70,70,];
impact_plate_angle = 70*pi/180;
measured_momentum = data_summary(7,1:8)
%measured_momentum = 37.2518;
for i=1:length(measured_momentum)
    flow_guess_low = 1;
    flow_guess_high = 30;
    flow_error=100;
    while(flow_error>0.001)
        flow_guess_test = flow_guess_low + (flow_guess_high -...
            flow_guess_low)/2;
        estimated_momentum_low = mymodel_generatedata(flow_guess_low,...
            mu,e,density,hf,impact_plate_angle);
        estimated_momentum_high = mymodel_generatedata(flow_guess_high,...
            mu,e,density,hf,impact_plate_angle);
        estimated_momentum_test = mymodel_generatedata(flow_guess_test,...
            mu,e,density,hf,impact_plate_angle);
        if estimated_momentum_test < measured_momentum(i)
                flow_guess_low = flow_guess_test;
            elseif estimated_momentum_test > measured_momentum(i)
                flow_guess_high = flow_guess_test;
            elseif estimated_momentum_test == measured_momentum(i)
                break
        else
                error('multiple roots or no root')
        end
        %perform error calculation between midpoint value and desired value
        flow_error = 100*abs(estimated_momentum_test -...
            measured_momentum(i))/measured_momentum(i);
    end
    estimated_momentum(i)=estimated_momentum_test
    flow_estimate(i) = flow_guess_test
end
flow_estimate(i)
```

```
max_pct_error=100.*(max(abs((flow_rate-flow_estimate)./flow_rate)))
avg_pct_error=100.*(mean(abs(flow_rate-flow_estimate)./flow_rate))
%% PLOT
hold off
figure(1), plot(flow_estimate, measured_momentum,'xr'),...
    title('Momentum vs. Flowrate')
hold on
plot(flow_rate,measured_momentum,'.')
legend('Predicted Data','Experimental Data','Location','Northwest')
xlabel('Flow Rate (kg/s)')
ylabel('Momentum (kg*m/s)')
```

Appendix Q

Matlab code used in the closed-loop estimation process

Q.1 Primary function file used in the closed-loop estimation process

The function file "run_closedloopestimation_generated_data.m" is the primary function file used in the system calibration process. This file defines pairs of mass flow rate and momentum imparted to the impact plate, as well as machine geometric parameters. It then calls the *lsqnonlin* function and a subsequent function file, "mymodel_closedloopestimation_generated_data.m', to facilitate the closed-loop estimation process.

```
%% run_closedloopestimation_generated_data.m
% This file performs "Closed Loop Estimation" with Matlab generated data.
% It estimates mass flow rate and internal model parameters given values
% for measured force and impact plate angles, and parameter estimates
clc
clear
close all
tic
%% Input estimated parameter values
mu = 0.19;
e = 0.60;
density = 1.084;
hf = 2;
%% Input Original Mass Flow Rate Data (Pretend we don't know this)
% This will be used for comparison purposes
actual_flow=2;
%% Impact plate angle for each flow rate (degrees)
impact_plate_angle = [40;45;50;55;60;65;70;75;80;85;90];
impact_plate_angle = impact_plate_angle.*pi./180; %Convert the angle to...
%radians
```

```
%% data summary WITH NOISE
%To generate this data the following parameter values were used
% mu = 0.1952;
% e = 0.636;
% density = 1133;
% hf = 2;
data_summary=[0.735 1.630   2.419   2.886   3.295   3.665   4.005   4.427
0.898   2.086   3.158   3.789   4.341   4.838   5.296   5.862
1.063   2.552   3.949   4.767   5.479   6.120   6.710   7.439
1.224   3.005   4.768   5.793   6.683   7.483   8.218   9.125
1.377   3.416   5.580   6.826   7.904   8.871   9.758   10.852
1.516   3.779   6.353   7.827   9.098   10.236  11.279  12.563
1.638   4.101   7.062   8.769   10.234  11.543  12.742  14.215
1.739   4.370   7.674   9.608   11.261  12.733  14.080  15.733
1.816   4.580   8.164   10.313  12.140  13.763  15.244  17.062
1.866   4.724   8.515   10.860  12.839  14.593  16.192  18.150
1.888   4.797   8.712   11.227  13.334  15.194  16.887  18.958]
%%
measured_momentum=data_summary(1:11,8);
%% Perform curvefitting
mu_low = 0.2*.9;
mu_high = 0.2*1.1;
e_low = 0.6*.9;
e_high = 0.6*1.1;
density_low = 1.084*.9;
density_high = 1.084*1.1;
flow=15;
mu_e_density_guess=[mu e density flow];
options=optimset('TolFun',1E-6,'TolX',1E-6,'DiffMinChange',0.001,...
    'DiffMaxChange',5,'MaxIter',100);
% lb=[0.9*mu 0.9*e 0.9*density 0.9*hf 0.0001*actual_flow]
% ub=[1.1*mu 1.1*e 1.1*density 1.1*hf 2*actual_flow]
%lb=[0.1818 0.54 0.997 0.1];
%ub=[0.2222 0.66 1.218 30];
lb=[mu_low e_low density_low 0.1];
ub=[mu_high e_high density_high 30];
[Estimates,resnorm,residual]=lsqnonlin(@...
    mymodel_closedloopestimation_generated_data,mu_e_density_guess,lb,...
    ub, options, measured_momentum, impact_plate_angle)
toc
```

Q.2 Secondary function file used in the closed-loop estimation process

The function file "mymodel_closedloopestimation_generated_data.m", shown below, is called by the primary function file, "run_closedloopestimation_generated_data.m". This secondary file receives user inputs which are used in subsequent calculations of the computational model. Additional subsequent function files (cf. Appendix N) are called to complete the calculations required by the four stages of the physical process.

```
%% mymodel_closedloopestimation_generated_data.m
% This file is called by "run_closedloopestimation_generated_data" and
% returns the error in the momentum estimate
function Error_Vector=mymodel_closedloopestimation_generated_data...
    (params, measured_momentum, impact_plate_angle)
%% Parameters
mu=params(1);   %friction coefficient
e=params(2);
density=params(3)*1000;
%height_fraction=params(4);
height_fraction=2;
actual_flow=params(4);
%% Machine Data
sprocket_speed = 400; %Rotations per minute
omega=2*pi*sprocket_speed/60; %Rotational speed of paddles (radians/sec)
sprocket_radius = 0.0541;%Pitch Radius of sprocket(8 teeth)—Units of meters
%There is also a high capacity sprocket with 10 teeth and a pitch...
%radius of 67mm
%paddle_frequency = 34/2.6; %Number of paddles per second
paddle_width = 0.21;  %width of paddle (meters)
paddle_distance = 0.18635; %distance between paddles (meters) — alternates...
                           %between 0.1656 meters and 0.207 meters;
                           %Here, an average distance is used
paddle_length = 0.16; %Paddle length (meters)
%dx = 0.5; %Horizontal distance between center of paddle rotation and lower
           %most point of plate — has dimension
%dy = 0; %Vertical distance between center of paddle rotation and lower
         %most point of plate — has dimension
linear_velocity = (omega*sprocket_radius);
```

```
paddle_frequency = linear_velocity/paddle_distance; %Units ofpaddles/second
%% Display Parameters on each iteration
params;
%% Define simulation constants
rho_step_size_hat = 0.0005; %Represents non-dimensionalized size of
                            %incremental radial positions of curved rows
rho_step_size = rho_step_size_hat*paddle_length; %Represents dimensionalized
                                                 %size of incremental radial
                                                 %positions of curved rows
%% HEIGHT METHOD 2
h0_save=zeros(length(measured_momentum),1); %h0(phi=0)=h0_0
bisect_error=0; % a flag to determine if an error occured during bisection...
%algorithm
for i=1:length(measured_momentum)
    %paddle_mass(i) = actual_flow(i)./paddle_frequency;
    paddle_mass = actual_flow.*paddle_distance./linear_velocity;
    paddle_area = paddle_mass./ (density*paddle_width);
    % Triangular area of grain on a paddle (m^2) - from a side view
    xa = 0;
    xb = paddle_length;        % search interval
    x_error=100;
    while (x_error >0.0000000001)
        xtest = xa + (xb-xa)/2;         % mid point of interval
        fa = ( (paddle_length/2).*height_fraction.*...
            (paddle_length - xa) ) -( (0.5).*xa.*paddle_length.*...
            sin(height_fraction.*...
            (paddle_length-xa)./paddle_length) ) -paddle_area;
        fb = ( (paddle_length/2).*height_fraction.*...
            (paddle_length - xb) ) -( (0.5).*xb.*paddle_length.*...
            sin(height_fraction.*...
            (paddle_length-xb)./paddle_length) ) -paddle_area;
        ftest = ( (paddle_length/2).*height_fraction.*...
            (paddle_length - xtest) ) - ( (0.5).*xtest.*paddle_length.*...
            sin(height_fraction.*(paddle_length-xtest)./...
            paddle_length) ) -paddle_area; % mid-point function value
        if sign(fa)*sign(ftest)<0       % if zero in left half
            xb = xtest;                 %   take left half of interval
        elseif sign(ftest)*sign(fb)<0   % if zero in right half
            xa = xtest;                 %   take right half of interval
        elseif ftest ==0                % if zero at mid-point
            break                       %   this is the zero
```

```
        else                            %
            bisect_error=1;
            break
            %error('multiple roots or no root')  % may have no zero or
            %multiple zeros
        end
        %perform error calculation between midpoint value and desired value
        x_error = abs(ftest-0);
    end
    h0_save(i)=xtest; %Initial height of grain on a paddle (meters)
end
%% Non-dimensionalization of initial grain heights, h0
h0_hat=h0_save./paddle_length; % dimensionless inital heights, h0
%% Evaluate discharge velocities, time to discharge, and radial position...
%of each row of grains
% These quantities are dimensionless
[vx_hat, vy_hat,tau_d_save, rho0_hat_save] = v_hat( mu,...
    rho_step_size_hat);% evaluate v_hat
%% Rescaling procedure
theta_d = (3*pi/2) + tau_d_save;  %Dimensionalized paddle...
%travel angle until discharge (radians)
vx = vx_hat.*(paddle_length*omega); %Dimensionalized discharge x-velocity
vy = vy_hat.*(paddle_length*omega); %Dimensionalized discharge y-velocity
rho = rho0_hat_save.*paddle_length;
%Dimensionalized incremental values of rho - represents radial...
%position of each row of grains
%% Evaluate critical radius for each flow rate and its vector index value
for i = 1:length(impact_plate_angle)
    phi = impact_plate_angle(i);
    dx = 0.4 + 0.1887*cos(phi);
    dy = 0.1887*sin(phi) - 0.16;

    [rho_cr,iteration_rho_cr]=critical_radius(mu, vx', vy', theta_d, rho,...
        phi, dx, dy, paddle_length); %critical radius between grains with
                                     %impact and without impact
    critical_radius_location(i) = rho_cr;
    iteration_critical_radius(i) = iteration_rho_cr;
end
%% Evaluate velocity*mass in a curved row
for i = 1:length(critical_radius_location)
    rho_cr = critical_radius_location(i);
```

```
    iteration_rho_cr = iteration_critical_radius(i);
    h0_save_temp = h0_save(i);
    [vx_dm_A_save_temp,vy_dm_A_save_temp]=mv(h0_save_temp, vx',...
        vy', rho_cr,rho_step_size, rho', iteration_rho_cr,actual_flow,...
        paddle_length,density,...
        paddle_width,height_fraction); %evaluate mv_hat
    vx_dm_A_save(i) = vx_dm_A_save_temp;
    vy_dm_A_save(i) = vy_dm_A_save_temp;
end
%% Evaluate change in momentum in x-direction
% only x-direction momentum is needed because sensor only measures in
% x-direction. Note: this still includes x and y velocities
% NOTE: the term "(1+e)" has been replaced by "e" on Aug.21,2009
% to account for relationship between change in momentum and force, which
% are proportional.
for i = 1:length(vx_dm_A_save)
    predicted_momentum(i) = ( vx_dm_A_save(i).*...
        sin(impact_plate_angle(i))-...
        vy_dm_A_save(i).*cos(impact_plate_angle(i)) ).*...
        sin(impact_plate_angle(i)).*(1+e);
end
%% Compare predicted force with measure experimental force
%Error_Vector = predicted_force - measured_force;
Error_Vector = predicted_momentum' - measured_momentum;
if bisect_error==1
    Error_Vector=10000000.*measured_momentum;
    bisect_error=0;
end
%% Convert Impact Plate Angle Back to Degrees For Plotting
impact_plate_angle_plot = impact_plate_angle.*180./pi;
%% PLOT
hold off
figure(1), plot( impact_plate_angle_plot, predicted_momentum,'xr'),...
    title('Momentum vs. Flowrate')
hold on
plot(impact_plate_angle_plot,measured_momentum,'.')
legend('Predicted Data','Generated Data','Location','Northwest')
title('Momentum Imparted to the Impact Plate for Varying Impact...
Plate Angles')
xlabel('Impact Plate Angle ')
ylabel('Momentum (kg*m/s)')
```

Appendix R

Numerical results of discrete element modeling simulations

Table R.1: Summary of simulation results of discrete element modeling simulations.

Impact Plate Angle (deg)	Mass (kg)	Trial-1 Momentum $\left(\frac{kg \cdot m}{s}\right)$	Trial-2 Momentum $\left(\frac{kg \cdot m}{s}\right)$	Trial-3 Momentum $\left(\frac{kg \cdot m}{s}\right)$
40	0.2	0.232	0.262	0.245
40	0.4	0.432	0.459	0.430
40	0.6	0.611	0.608	0.599
40	0.8	0.742	0.776	0.743
40	1.0	0.873	0.894	0.905
40	1.2	1.014	1.004	1.024
40	1.4	1.123	1.093	1.111
40	1.6	1.245	1.222	1.226
40	1.8	1.409	1.387	1.318
50	0.2	0.431	0.390	0.461
50	0.4	0.724	0.717	0.724
50	0.6	0.926	0.965	0.986
50	0.8	1.184	1.191	1.198
50	1.0	1.394	1.450	1.386
50	1.2	1.625	1.528	1.623
50	1.4	1.796	1.851	1.782

Continued on Next Page...

Table R.1 – Continued

Impact Plate Angle (deg)	Mass kg	Trial-1 Momentum $\left(\frac{kg \cdot m}{s}\right)$	Trial-2 Momentum $\left(\frac{kg \cdot m}{s}\right)$	Trial-3 Momentum $\left(\frac{kg \cdot m}{s}\right)$
50	1.6	2.015	1.976	1.996
50	1.8	2.135	2.270	2.238
60	0.2	0.603	0.604	0.601
60	0.4	0.957	0.969	0.989
60	0.6	1.312	1.348	1.342
60	0.8	1.667	1.647	1.667
60	1.0	1.924	1.970	1.932
60	1.2	2.235	2.233	2.255
60	1.4	2.488	2.532	2.509
60	1.6	2.811	2.734	2.785
60	1.8	3.131	2.996	3.029
70	0.2	0.803	0.794	0.749
70	0.4	1.307	1.222	1.189
70	0.6	1.742	1.650	1.685
70	0.8	2.196	2.072	2.128
70	1.0	2.355	2.483	2.439
70	1.2	2.822	2.842	2.814
70	1.4	3.157	3.227	3.193
70	1.6	3.539	3.499	3.539
70	1.8	3.819	3.922	3.835
80	0.2	0.845	0.867	0.878
80	0.4	1.509	1.461	1.442
80	0.6	2.016	1.968	1.952
80	0.8	2.635	2.492	2.492
80	1.0	2.837	2.833	2.936
80	1.2	3.211	3.388	3.289

Continued on Next Page...

Table R.1 – Continued

Impact Plate Angle (deg)	Mass kg	Trial-1 Momentum $\left(\frac{kg \cdot m}{s}\right)$	Trial-2 Momentum $\left(\frac{kg \cdot m}{s}\right)$	Trial-3 Momentum $\left(\frac{kg \cdot m}{s}\right)$
80	1.4	3.660	3.660	3.776
80	1.6	4.089	4.039	4.122
80	1.8	4.418	4.500	4.542
90	0.2	0.890	0.864	0.825
90	0.4	1.585	1.675	1.467
90	0.6	2.172	2.193	2.164
90	0.8	2.636	2.761	2.725
90	1.0	3.098	3.144	3.103
90	1.2	3.477	3.569	3.655
90	1.4	4.047	4.050	4.086
90	1.6	4.420	4.276	4.410
90	1.8	4.909	4.910	5.038

Appendix S

Numerical results of experiments using a lab-sized testbench

Table S.1: Summary of simulation results of experiments using a lab-sized testbench.

		Sensor Output (mV)									
Impact Plate Angle (deg)	Mass (kg)	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10
40	0.05	5.24	-4.79	6.51	13.97	21.83	27.99	3.56	18.01	4.51	20.90
40	0.10	35.24	12.29	23.90	29.16	28.69	37.94	64.90	32.59	35.97	20.58
40	0.15	25.17	28.62	33.95	33.84	55.78	39.62	41.76	39.57	44.76	41.89
40	0.20	52.47	31.83	37.38	35.26	42.24	38.65	30.62	51.41	56.33	40.67
40	0.25	44.15	61.07	54.56	46.84	71.16	60.28	38.73	50.20	57.44	28.87
40	0.30	56.31	62.30	46.22	37.89	53.00	43.05	48.65	62.98	58.32	43.34
50	0.05	14.37	-53.2	23.09	14.78	24.80	43.14	29.28	34.98	22.12	21.00
50	0.10	44.36	39.84	24.91	51.80	39.78	19.99	58.78	32.07	43.73	33.51
50	0.15	44.98	56.52	60.87	64.68	56.32	49.26	67.03	44.97	74.40	57.55
50	0.20	73.53	76.26	53.59	79.95	66.54	73.23	67.68	63.54	49.93	70.75
50	0.25	93.36	93.23	84.38	82.96	83.73	76.68	84.88	77.99	74.35	83.82
50	0.30	69.81	74.86	75.96	47.74	73.18	75.72	77.12	82.14	58.33	67.23
60	0.05	35.62	15.29	37.43	22.41	13.73	18.47	47.04	45.92	41.79	38.86
60	0.10	71.76	52.08	66.56	53.72	60.86	56.71	59.98	59.63	55.75	47.18
60	0.15	50.98	78.14	73.75	70.97	74.15	89.29	80.94	86.58	88.38	92.87

Continued on Next Page...

Table S.1 – Continued

		Sensor Output (mV)									
Impact Plate Angle (deg)	Mass (kg)	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10
60	0.20	98.00	120.53	114.59	110.16	101.04	110.81	114.77	95.85	125.72	132.93
60	0.25	127.60	119.03	126.64	138.58	120.82	121.02	105.27	126.10	160.69	120.55
60	0.30	124.05	143.38	134.49	169.87	118.82	112.77	140.54	128.47	128.18	127.33
70	0.05	55.18	52.90	49.28	58.09	46.77	52.74	58.78	26.58	43.03	58.33
70	0.10	88.21	79.23	99.76	86.67	86.00	82.77	71.87	90.56	75.43	85.66
70	0.15	110.63	125.36	111.11	115.99	123.47	111.18	127.08	155.73	138.67	115.81
70	0.20	133.02	116.44	165.08	156.67	163.42	154.69	177.93	118.25	159.99	149.73
70	0.25	189.07	198.94	182.99	208.04	180.52	161.18	181.94	182.12	171.67	168.78
70	0.30	202.93	222.77	192.97	209.04	196.52	232.43	211.64	196.37	226.91	209.94
80	0.05	59.67	56.01	47.44	44.97	58.86	58.15	47.33	54.44	63.73	56.06
80	0.10	84.62	103.14	92.37	117.76	116.80	106.31	130.07	107.22	106.94	122.99
80	0.15	163.36	130.00	166.81	177.85	159.66	163.13	139.90	183.46	168.10	166.52
80	0.20	173.51	207.32	186.62	202.14	217.64	169.32	190.74	227.81	177.79	210.42
80	0.25	247.38	251.36	245.41	267.91	224.58	244.93	185.28	256.43	256.31	258.40
80	0.30	265.70	251.23	260.51	254.35	275.46	255.82	258.36	261.34	251.24	233.96
90	0.05	71.84	68.62	71.36	74.17	81.29	62.78	58.85	66.54	89.09	74.98
90	0.10	120.99	120.89	139.56	149.73	134.37	133.05	114.75	149.96	132.28	144.80
90	0.15	157.99	181.70	187.86	175.54	183.66	173.49	205.41	172.73	204.11	163.56
90	0.20	242.94	233.57	193.98	241.54	221.79	250.85	228.43	228.52	229.28	231.31
90	0.25	285.92	302.95	289.21	259.60	270.07	289.17	319.53	288.63	271.28	331.16
90	0.30	282.49	294.69	308.93	317.22	329.45	315.65	292.98	321.59	324.98	307.64

Appendix T

Computer control code used in the control of solenoids for lab experiments

The following C code was used to control the action of the solenoids used in the laboratory experiments. This code was downloaded to the microcontroller, which was wired to the solenoids and the optical sensor. Signals from the sensor were monitored via the microcontroller and C code to facilitate the control of the action of the solenoids.

```
//*****************************************************************************
//*****************************************************************************
#include <stdarg.h>
#include <stdio.h>
#include  "msp430x22x2.h"

int UART_printf(const char *format, ...);

char newprint = 0;
long timeint = 0;
int release_flag = 0;
int paddle_count = 0;
long solenoid_off_count = 0;
int debounce_count = 0;

void main(void)
{

WDTCTL = WDTPW + WDTHOLD;                 // Stop WDT

if (CALBC1_16MHZ ==0xFF || CALDCO_16MHZ == 0xFF)
  {
    while(1);                               // If calibration constants erased
                                            // do not load, trap CPU!!
  }
```

```
  DCOCTL = CALDCO_16MHZ;                    // Set uC to run at approximately 16 Mhz
  BCSCTL1 = CALBC1_16MHZ;

//  if (CALBC1_1MHZ ==0xFF || CALDCO_1MHZ == 0xFF)
// {
//    while(1);                               // If calibration constants erased
//                                            // do not load, trap CPU!!
//  }

//  BCSCTL1 = CALBC1_1MHZ;                    // Set DCO
//  DCOCTL = CALDCO_1MHZ;

  P3SEL = 0x30;                             // P3.4,5 = USCI_A0 TXD/RXD
  UCA0CTL1 |= UCSSEL_2;                     // SMCLK
  UCA0BR0 = 0x82;                           // 16MHz 9600
  UCA0BR1 = 0x6;                            // 16MHz 9600
  UCA0MCTL = UCBRS_6;                       // Modulation UCBRSx = 6
  UCA0CTL1 &= ~UCSWRST;                     // **Initialize USCI state machine**
//  IE2 |= UCA0RXIE;                          // Enable USCI_A0 RX interrupt

  P1DIR |= 0x80;                            // P1.7 as output

  ////////////////////////////////////////////////////////////////////
  // Phototransistor register setup
  // P1.0 is the phototransistor
  P2SEL &= ~0x01;                           // P2.0 GPIO
  P2DIR &= ~0x01;                           // P2.0 as input
  P2IE &= ~0x01;                            // P2.0 interrupt disabled
  P2IES |= 0x01;                            // P2.0 Hi/lo edge
  P2IFG &= ~0x01;                           // P2.0 IFG cleared
  P2IE |= 0x01;                             // P2.0 interrupt enabled
  ////////////////////////////////////////////////////////////////////

  ////////////////////////////////////////////////////////////////////
  //Solenoid register setup
  // P1.1 is the solenoid
  P1DIR |= 0x02;                            // P1.1 as output
  P1OUT &= 0xFD;                            // P1.1 initially off
  ////////////////////////////////////////////////////////////////////
```

```
  CCTL0 = CCIE;                             // CCR0 interrupt enabled
  CCR0 = 16000;                              // 0.001s
  TACTL = TASSEL_2 + MC_2;                  // SMCLK, contmode

   _BIS_SR(GIE);                 // Enable global interrupt

  while(1) {



    if (newprint == 1)  { // newprint set to one in timer ISR



      //P1OUT ^= 0x01;  // toggle P1.0 on and off

      UART_printf("Hello %d\n\r",timeint);  // Print to UART
      newprint = 0;  // reset flag back to zero

    }
  }

}


// only 15 characters can be sent at a time
#define UART_PRINTF_SIZE 15
char txbuff[UART_PRINTF_SIZE];  // global character buffer set in the
...function UART_printf and the output in the TX ISR function
signed char txcount = 0;
signed char currentindex = 0;
signed char senddone = 1;


// This function assumes txbuff has already been filled with the characters to send.
// It initializes txcount to the number of chars to send and then sets
...senddone = 0 so that txbuff is sent out UCATX
int sendchars(int size) {
```

```
  if (senddone == 1) {  // Only setup txcount if previous transmission complete
    if (size < UART_PRINTF_SIZE) {
      txcount = size;
    } else {
      txcount = UART_PRINTF_SIZE;
    }

    currentindex = 1;
    senddone = 0;  // signal that a new transmission should occur.
    UCA0TXBUF = txbuff[0];
    IE2 |= UCA0TXIE;                        // Enable USCI_A0 TX interrupt

    return(0);
  } else {
    return(-1); // error
  }

}

int UART_printf(const char *format, ...)
{
  // the "va" and "v" functions handle the variable argument ...
      ...in the function parameters
  va_list ap;
  int error;

  va_start(ap, format);      /* Variable argument begin */
  error = sendchars(vsprintf(txbuff, format, ap));   // fill txbuff
  ...with the format string and the pass the size of txbuff to the sendchars function
  va_end(ap);                /* Variable argument end */
  return error;
}

// Timer A0 interrupt service routine
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{

  CCR0 += 16000;   // .001 second                          // Add Offset to
  ...CCR0   because timer 0 has been told to continuously count up.
  timeint++; // Keep track of time for main while loop.
```

```
  if ((P2IE & 0x01) == 0x00)  // If interrupt is disabled (done in ISR)
  {
    debounce_count++;         // This counts ever 0.001 seconds
    if(debounce_count≥10){   // If reached 0.01 seconds
      debounce_count=0;       // Debounce count reset to zero
      P2IE |= 0x01;           // P2.6 interrupt ensabled
    }
  }

  if(release_flag==1){    //Paddle has rotated enough times and solenoid
      ...activated
    solenoid_off_count++;
  }

  //If solenoid has been ON for 0.5 seconds...
  if(solenoid_off_count ≥1000){
    P1OUT &= ¬0x02;                //Turn solenoid off on P1.1
  }

  if ((timeint%500) == 0) {
    newprint = 1;  // flag main while loop that .5seconds have gone by.
  }
}


#pragma vector=USCIAB0TX_VECTOR
__interrupt void USCI0TX_ISR(void)
{
  UCA0TXBUF = txbuff[currentindex];                // TX next character
  currentindex++;
  if (currentindex == txcount) {
          senddone = 1;
          IE2 &= ¬UCA0TXIE;                       // Disable USCI_A0 TX interrupt
  }

}

// Port 2 interrupt service routine
#pragma vector=PORT2_VECTOR
```

```
__interrupt void Port_2(void)
{
  if ((P2IE & 0x01) == 0x00)  //IS THIS CASE NECESSARY?????
  {
    P2IFG &= ~0x01;           //Clear flag
  }

  //Interrupt for photoresistor
   if ((P2IFG & 0x01) == 0x01){        //If flag is set

     paddle_count = paddle_count + 1;  //Count number of paddle rotations

     if(paddle_count >= 40){
       P1OUT |= 0x02;                //Turn ON solenoid on P1.1
       release_flag=1;
     }

     //P1IFG &= ~0x20;                   // Clear interrupt flag on P1.5
     //P1IFG &= ~0x10;                        // P1.4 IFG cleared for 1st piezo
     //P1IE |= 0x10;                   // P1.4 (1st Piezo) interrupt enabled
     //x=timeint;
     //i = i + 1;

     P2IE &= ~0x01;                    // P2.6 interrupt disabled
     P2IFG &= ~0x01;                   // P2.6 IFG cleared for photoresistor
    }

}
```

References

[1] Ambuel, J.R., *Knowledge and decision support for variable rate application of materials in precision agriculture*, Ph.D dissertation. Iowa State University, Ames, IA, USA, 1995.

[2] Anthonis, J., Strubbe, G., Maertens, K., De Baerdemaeker, J., and Ramon, H. "Design of a friction independent mass flow sensor by force measurement on a circular chute," *Biosystems Engineering* **84(2)**, pp. 127-136, 2003.

[3] Arslan, S., and Colvin, T.S., "Laboratory performance of a yield monitor," *Applied Engineering in Agriculture* **15(3)**, pp. 189-195, 1999.

[4] Arslan, S., Inanc, F., Gray, J.N., and Colvin, T.S. "Grain flow measurements with X-ray techniques," *Computers and Electronics in Agriculture* **26(1)**, pp. 65-80, 2000.

[5] De Baerdemaeker, J., Delcroix, R., and Lindemans, P. "Monitoring grain flow on combines," in *Proceedings of the Agri-mation 1 Conference & Exposition*, pp. 329-337, 1985.

[6] Birrell S.J., Sudduth K.A., and Borgelt S.C. "Comparison of sensors and techniques for crop yield mapping," *Computers and Electronics in Agriculture* **14(2-3)**, pp. 215-233, 1996.

[7] Borgelt, S.C., "Sensing and measurement technologies for site specific management," in *Soil Specific Crop Management*, Madison: American Society of Agronomy, Inc., pp. 141-157, 1993.

[8] Burks, T.F., Shearer, S.A., Sobolik, C.J., and Fulton, J.P., "Combine Yield Monitor Test Facility Development," in *Proceedings of 2000 ASAE Annual Intenational Meeting*, pp. 2559-2573, 2000.

[9] Burks, T.F., Shearer, S.A., Fulton, J.P., and Sobolik, C.J., "Combine yield monitor test facility development and initial monitoring test," *Applied Engineering in Agriculture* **19(1)**, pp. 5-12, 2003.

[10] Burks, T.F., Shearer, S.A., Fulton, J.P., and Sobolik, C.J., "Effects of time-varying inflow rates on combine yield monitor accuracy," *Applied Engineering in Agriculture* **20(3)**, pp. 269-275, 2004.

[11] Chaplin, J., Hemming, N., and Hetchler, B. "Comparison of impact plate and torque-based grain mass flow sensors," *Transactions of the American Society of Agricultural Engineers* **47(4)**, pp. 1337-1345, 2004.

[12] Colvin, T.S. "Automated weighing and moisture sampling for a field-plot combine," *Applied Engineering in Agriculture* **6(6)**, pp. 713-714, 1990.

[13] Colvin, T., and Arslan, S. "'Yield Monitor Accuracy," *Site-Specific Management Guidelines.* Potash & Phosphate Institute, Print. 2009.

[14] Grisso, R., Alley, M., and McClellan, P. "Precision Farming Tools: Yield Monitor," College of Agriculture and Life Sciences, Virginia Polytechnic Institute and State University. Publication 442-502. Print. 2009.

[15] Hemming, N., and Chaplin, J. "Precision of real time grain yield data," in *Proceedings of 2004 ASAE Annual International Meeting*, pp. 747-756, 2004.

[16] Hennens, D., Baert, J, Broos, B., Ramon, H., and De Baerdemaeker, J., "Development of a flow model for the design of a momentum type beet mass flow sensor," *Biosystems Engineering* **85(4)**, pp. 425-437, 2003.

[17] Jasa, P.J., Grisso, R.D., and Wilcox, J.C. "Yield monitor accuracy at reduced flow rates," *Proceedings of 2000 ASAE Annual Intenational Meeting*, pp. 2575-2586, 2000.

[18] Phelan, J. Personal communications. September, 2008.

[19] Reyns, P., Missotten, B, Ramon, H., and De Baerdemaeker, J., "Review of combine sensors for precision farming," *Precision Agriculture* **3(2)**, pp. 169-182, 2002.

[20] Schrock, M.D., Kuhlman, D.K., Hinnen, R.T., Oard, D.L., and Pringle, J.L., "Sensing grain yield with a triangular elevator," in *Proceedings of Site-specific Management for Agricultural Systems* pp. 637-650, 1995.

[21] Schrock, M.D., Oard, D.L., Taylor, R.K., Eisele, E.L., Zhang, N., Suhardjito, and Pringle, J.L., "Diaphragm impact sensor for measuring combine grain flow," *Applied Engineering in Agriculture* **15(6)**, pp. 639-642, 1999.

[22] Strubbe, G.J., Missotten, B., and De Baerdemaeker, J., "Mass flow measurement with a curved plate at the exit of an elevator," in *Proceedings of the Third International Conference on Precision Agriculture* pp. 703-712, 1996.

[23] Strubbe, G.J., Missotten, B., and De Baerdemaeker, J., "Performance evaluation of a three-dimensional optical volume flow meter," *Applied Engineering in Agriculture* **12(4)**, pp. 403-409, 1996.

[24] Tijskens, E., Ramon, H, and De Baerdemaeker, J., "Discrete element modeling for process simulation in agriculture," *Journal of Sound and Vibration* **266(3)**, pp. 493-514, 2003.

[25] Wagner L.E., *Development of an auger-based grain flow meter for use in a yield mapping system,* Ph.D dissertation. Iowa State University, Ames, IA, USA, 1988.

[26] Wang, B., and Li, M. "Development of a grain volumetric-flow sensor based on photoelectrical principle," in *Proceedings of SPIE - The International Society for Optical Engineering*, art. no. 71571L, 2009.

[27] Zandonadi, R.S., Stombaugh, T.S., Shearer, S.A., Sama, M.M., and Queiroz, D.M., "Laboratory performance of a low cost mass flow sensor for combines," in *Proceedings of the ASABE Annual International Meeting 2008*, pp. 3843-3858, 2008.