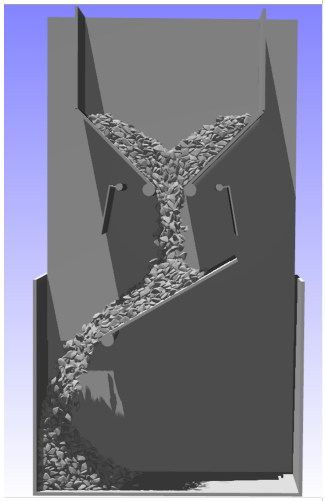
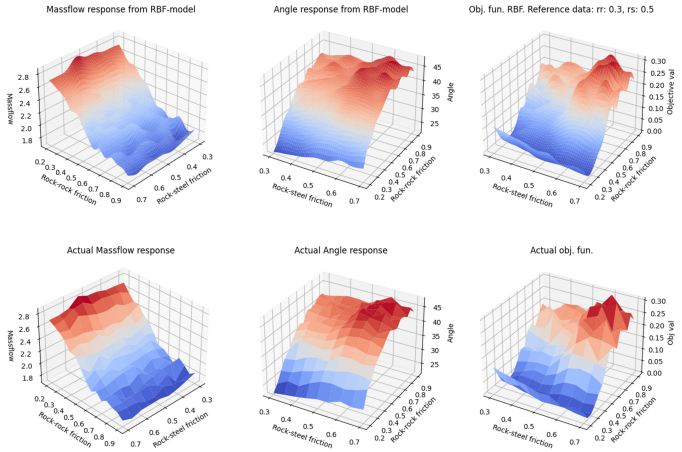
CHALMERS
UNIVERSITY OF TECHNOLOGY





Parallel optimization for model calibration of large scale particle systems

An optimization framework for calibration of the discrete element method

Master's thesis in Engineering Mathematics and Computational Science

Alexander Samuelsson

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Parallel optimization for model calibration of large scale particle systems

An optimization framework for calibration of the discrete element method

Alexander Samuelsson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Fraunhofer Chalmers Centre
Chalmers University of Technology
Gothenburg, Sweden 2023

Parallel optimization for model calibration of large scale particle systems
An optimization framework for calibration of the discrete element method
Alexander Samuelsson

Supervisor: Johannes Quist, Fraunhofer Chalmers Centre
Christoffer Cromvik, Fraunhofer Chalmers Centre
Examiner: Anders Logg, Department of Mathematical Sciences

Master's Thesis 2023
Department of Mathematical Sciences
Fraunhofer Chalmers Centre
Chalmers University of Technology
SE-412 96 Gothenburg

Cover: A DEM simulation model and surfaces of its responses when varying the friction.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Parallel optimization for model calibration of large scale particle systems
A framework for calibration of the discrete element method
Alexander Samuelsson
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

In this thesis we present an optimization framework for model calibration of Discrete Element Method (DEM) simulations. The DEM is a method for simulating large scale particle systems. Numerical complexity makes calibration of models computationally expensive and time consuming. We approach the procedure by viewing it as a black box optimization problem and apply a parallel optimization algorithm to perform calibration.
The algorithm is a surrogate optimization method using Radial Basis Functions. We first sample the objective function with a Symmetric Latin Hypercube serving as a design of experiments. In the iterative part of the algorithm we implement the Metric Stochastic Response surface method proposed by Regis and Shoemaker. We choose new evaluation points based on a weighted score of the expected objective value of the point and its distance from previously evaluated points. The weight decides whether to focus on exploitation by trying to minimize the objective value of the next point, or exploration by evaluating points in areas of the sample space where we have less information. To reduce the wall time required in finding a solution the framework allows for asynchronous parallelism by simply evaluating many points at the same time. We also implement a method for early termination.
The algorithm is applied to two DEM optimization problems. The first one is a calibration problem involving a device used to calibrate the friction of a material. The second one is an industrial optimization case where the position of a funnel is adjusted to achieve an even split of material flow. In both cases, the algorithm converges to a solution in less than 40 evaluations. Furthermore the early termination and parallel strategies are shown to reduce the time required to solve the calibration case. We finish by discussing and comparing our results, giving some advantages and disadvantages found with the early termination functionality.

Keywords: DEM, Discrete element method, Calibration, Optimization, Surrogate, Black-box, Parallelism

Acknowledgements

I would like to express my deepest gratitude to my supervisors Johannes and Christoffer, without whom this thesis could not have been completed. Both have always had an open ear for discussions and have provided valuable feedback throughout the course of this project. After our meetings I have always felt inspired and energized to continue working. Johannes, your expertise and counsel in discrete element modelling as been invaluable. Thank you especially for helping with the setup of the simulation cases and for general guidance in litterature searching and workflow. Christoffer, your knowledge in optimization and mathematics have been very helpful and appreciated in our discussions. Thank you especially for your valuable advice when implementing early termination and parallelism and for feedback on the final report. I would also like to thank Anita for helping me get set up in the beginning of the year and for always being available for questions about everything from simple git-commands to the theory behind DEM. Furthermore I would like to express gratitude to my examiner Anders for valuable feedback during our meetings; The DEM team at FCC for their support; Simon for helping me with computer troubles; My opponent Fredrik for reading my thesis and his suggestions to improve it; and the whole of FCC for giving me an opportunity to write my thesis here and providing a great work environment.
I also want to thank my friends, family and my partner Ida. In particular I want to give a heartfelt thanks to my father for years of support during my studies and for always keeping an interest in my work.

The computations were enabled by resources provided by Chalmers e-Commons at Chalmers.

Alexander Samuelsson, Gothenburg, May 2023

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

Acronym	**Actual text**
AoR	Angle of repose
APPS	Asynchronous Parallel Pattern Search
BBO	Black box optimization
C3SE	Chalmers Centre for Computational Science and Engineering
CS	Compass Search
DEM	Discrete element method
DOE	Design of experiments
FCC	Fraunhofer Chalmers Centre
GPS	Generalized Pattern Search
GPU	Graphical Processing Unit
LHD	Latin hypercube design
MADS	Mesh Adaptive Direct Search
MPI	Message Passing Interface
MSRS	Metric Stochastic Response Surface Method
POAP	Plumbing for Optimization with Asynchronous Parallelism
pySOT	Python Surrogate Optimization Toolbox
RBF	Radial Basis Function
SLHD	Symmetric Latin Hypercube Design

Nomenclature

NOTE: The following nonclamenture is valid from Chapter 3 and onwards. In Chapter 1 and 2, all notation is specified in the text.

Symbol	Explanation
$f(x)$	Objective function
x	Variable; In this context a set of DEM parameters
x^*	Optimal solution
X	Constraint set
r	Reference data
p	Fixed DEM model parameters and properties
y_i	Model response i
$\varphi(x)_i$	The model response gathering function of model response i
d	Dimension of the optimization problem
η_i	Relative error of model response i
w_i	Weight of model response i
n	Evaluation number
$N_{\max}$	Maximum number of evaluations
A_n	Set of previously evaluated points
B_n	Set of previously evaluated points and their objective values
$s(x)$	Surrogate model of the objective function
J	Set of initial sampling points
t	Number of candidate points in each iteration
Ω_n	Set of candidate points in iteration n
V_n^R	Surrogate criteria score
V_n^D	Distance criteria score
λ	Score function weight
$\mathcal{F}_n(x)$	Score of point x in iteration n
σ_n	Stepsize in iteration n

C_{success}	Number of successful iterations before step size increase
C_{fail}	Number of failed iterations before step size decrease
η_{et}	Relative errors used as early termination criteria
c	Early termination factor
P	Number of workers in the parallel algorithm

Contents

List of Figures

List of Tables

1

Introduction

The rapid increase in computational power has enabled computer simulations to become a vital part in the development process of new products and machinery. Through simulations one can perform virtual measurements and analysis when these would be too expensive to perform physically. It also offers opportunities for discovering unforeseen risks and quality problems early on, thereby optimizing the design process. One area of research that has seen increased interest the last decade is the simulation of large scale particle systems, where systems consisting of millions or even billions of particles are simulated during some time window. Typical examples of such systems are granular materials such as sand, rock or powder and the simulations generally involve material flow or breakage.

The Discrete Element Method (DEM), developed by Peter Cundall in the 1970s [11], is a powerful numerical approach used to simulate the behavior of large scale particle systems. In DEM, each particle is modeled individually and represented by its shape, size, and material properties such as density, Young's modulus, Poisson's ratio, friction, and restitution [16]. These properties determine how each particle interacts with other particles and the surrounding environment through force-displacement relationships, contact forces, and friction. By modeling each particle in this way, it is possible to predict the behavior of the entire system. A major limitation of DEM is its high computational cost, particularly for larger systems. Simulation times can often exceed several days on regular computers and are heavily dependent on the number of particles and timesteps used. Due to these requirements, DEM simulations are typically performed on computer clusters or other high-performance computing platforms.

Fraunhofer Chalmers Centre (FCC) conducts research and develops software solutions for the modelling and simulation of particle systems based on the discrete element method. A DEM solver and modelling tool named Demify has been developed. The software is applied for a wide range of different industrial applications and types of granular materials. A typical application can be powder recoating in additive manufacturing, loading and handling of rock materials in road construction, ploughing of soil in agriculture or rock fracture in minerals processing.

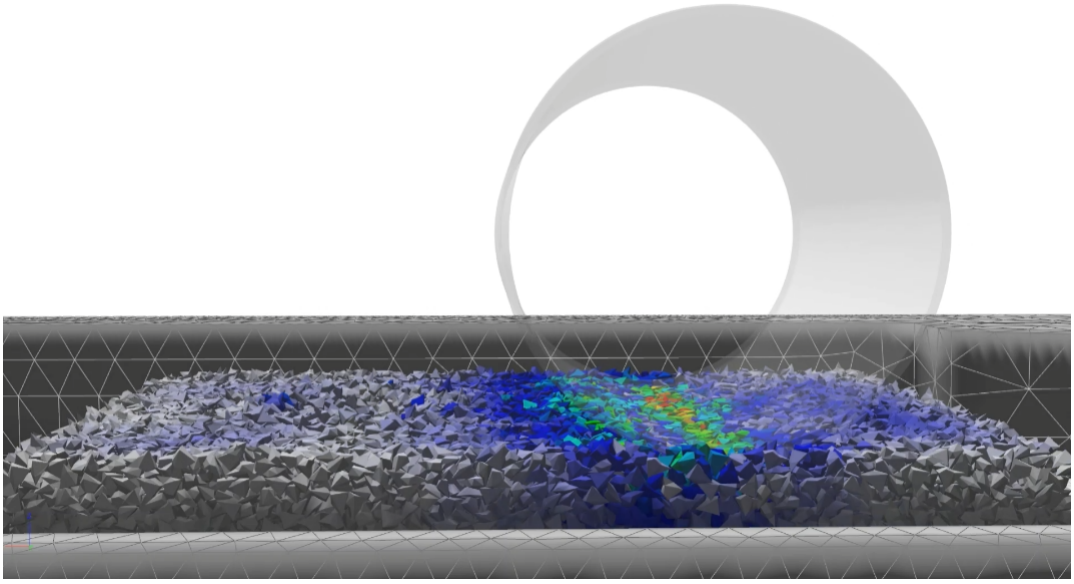


Figure 1.1: An example simulation in Demify(r) showing a vibratory roller for compaction of unbound aggregates in road construction [27]

1.1 Calibration of DEM models

Each DEM simulation requires the specification of a number of model parameters [28]. These parameters have a large influence on the behavior of the simulated material and the results of the simulation. The parameters that need to be specified range from modelling decisions such as particle geometry and collision models used to physical properties of materials such as friction and density. One also needs to set simulation specific parameters such as the timestep and timescale used. Some parameters can be measured in laboratory settings, or are specified as part of the simulation design, but almost all simulations contain some parameters that can only be obtained by calibration [28]. The calibration of a DEM model involves adjusting the model parameters to accurately represent the behavior of the physical system being modeled. The process typically involves comparing the model response to experimental data from the system being modeled, and adjusting the model parameters until a good match is achieved.

The simplest way to perform calibration is to use a trial and error approach wherein parameters are individually varied and the model response observed [15]. This approach has several significant drawbacks, including the need for a large number of DEM simulations, which requires substantial computational resources. In addition, this method provides limited mechanistic insights and the calibrated parameters may be suboptimal [28]. These issues have prompted research into more sophisticated calibration approaches utilizing optimization. FCC have also seen an increased need to apply optimization in model calibration. The computational cost of DEM models means that there are large gains to be made from reducing the number of simulations required.

The optimization approach to calibration usually starts with obtaining some ref-

erence data from real world experiments. These are considered the desired model response. For a given parameter set the model response is calculated and assessed by comparing it to the reference data. If the simulation response fulfills some termination criteria, typically if the model response is "close enough" to the desired response, the optimization is stopped. Otherwise new parameters are selected based on the information gathered and the simulation response is calculated again. This process is continued until the termination criteria is met. It is important to note that many conventional optimization methods are not applicable due to the nature of the problem. The objective function of such an optimization problem can be considered a computationally expensive black box function where gradients are not directly obtainable and the function topology has many local optima [30]. In particular the long simulation times are a major limiting factor in the choice of optimization algorithm. For many computationally expensive problems parallelism is used to speed up the solution process. FCC have noticed a lack of DEM calibration approaches utilizing parallelism and have a clear need for a generic, efficient and robust optimization algorithm. This project aims to provide an optimization framework for efficient calibration of DEM models.

1.2 Problem statement

The first milestone of the project is to implement a black box optimization framework that can be used for DEM calibration. The project aims to answer the following questions:

- How can a black box optimization framework be used to efficiently calibrate DEM models?
- What algorithm is suitable for solving black box optimization problems in the context of DEM calibration?
- How can early termination and parallelism be utilized to improve the speed of calibration?

After the framework is created we evaluate it on a DEM calibration problem and answer the following questions

- Do the iterates produced by the algorithm converge to a solution with which we can consider the models calibrated?
- Does the parallelism and early termination improve efficiency and reduce computation time when compared to the serial optimization algorithm?

1.2.1 Contribution

There exists many articles and methods for DEM calibration in the open literature however most do not utilize optimization in any way, instead reverting to manual calibration in combination with different design of experiments. Some of methods that do use optimization might require many evaluations or only work for some

specific cases. There exists some articles describing more general and efficient optimization frameworks, most notably [30] and [28], these are briefly presented in section 1.3.

The main contribution of this project is a new optimization framework that can be used for efficient calibration of a DEM model. As long as the calibration problem is well defined and the model provides good enough responses for the calibration variables it can be applied to any DEM model. While there have been other frameworks developed, this project is the first to use the black box optimization algorithm developed in [31], thus contributing with a new approach to the calibration problem. In addition, to the best of our knowledge, this is the only DEM calibration framework to utilize asynchronous parallelism in combination with early termination and with built in MPI support for parallel evaluations on computer clusters.

1.3 Field overview

In this section, a short overview of calibration methods used today is presented. C.J. Coetzee categorizes the approaches found in current literature into two main categories in their review [10]. The first category is composed of direct measuring approaches, which involve conducting laboratory experiments to directly measure a specific material property on particle or contact level. The direct measuring approach is generally simpler than other methods, however it may not always yield accurate results. Certain parameters, such as particle size and density, can be measured accurately, whereas other parameters may require alternative methods. As direct measuring methods simply consist of measuring material properties in different ways, they are not suitable for optimization and we do not consider them further in this thesis.

The second category of approaches are called bulk calibration approaches. Here, properties are measured on a material bulk scale instead, and parameters are changed iteratively until the model response matches the measured bulk response. Some parameters are usually measured directly to reduce the complexity of the bulk approach [10]. When using this method it is naturally important to choose to measure bulk properties in which changes to the input parameters correlate strongly with changes in the bulk response. If changing a parameter does not produce a statistically significant change in the model response it is very hard or impossible to calibrate that parameter. There are numerous possible bulk calibration approaches and a thorough review is given in [10]. Two possible bulk measures are the angle of repose and hopper discharge rate or massflow. These are used for calibration in [23] and [35] among others. A hopper or silo is filled with material and then opened, allowing the material to discharge and form a pile on the ground. The angle of repose is defined as the angle between the slope of the pile and the horizontal ground. This measure typically has a significant dependence on both particle-particle friction and particle-ground friction. The discharge rate is also an important bulk response. Typically material flows out of the hopper at a constant rate until nearly all of the material has been discharged.

Utilizing optimization in combination with bulk calibration methods is a relatively new area of research, with many different approaches. In [5], a neural network is trained with results from many DEM simulations and is able to provide a link between bulk properties and parameter input. The network is able to generalize to parameter inputs not included in the training data and can therefore be used for calibration. However the network required hundreds of DEM simulations to train, and if the simulation is changed it might need retraining, meaning that it is not suitable for larger DEM models. Westbrink et al. proposed an approach using multi-objective reinforcement learning, where an agent is trained using an angle of repose test and was able to calibrate three different materials correctly [34]. Once again, the number of DEM simulations required is considered too many for this project. There have also been approaches using Bayesian statistics. Cheng et al. uses an iterative Bayesian filter to estimate the posterior of relevant parameters in the calibration problem [9]. It shows promising results and has some support for local parallelism, but does not have functionality for early termination or cluster parallelism.

Some attempts at a more general framework have also been made. In [28], Rackl and Hanley proposed an efficient calibration procedure for spherical glass beads using angle of repose and bulk density. Their approach consisted of sampling the parameter space using Latin hypercube sampling and using the results as inputs to a Kriging meta-meta model. By optimizing the computationally inexpensive meta-model they found promising starting values for the calibration of the DEM model, thereby requiring fewer steps in the model calibration. The results also demonstrated the existence of a solution space in which different parameter combinations lead to similar results and highlighted the need for a systematic and automatized calibration method. Richter et al. proposed a general surrogate based framework for optimization based calibration [30]. It views the calibration problem as a black box optimization problem and applies a surrogate based optimization algorithms. The optimization produces iterates that converge to local minima. These are deemed to give good enough calibration results by being located in the so called stochastic fluctuation range of the model. Within which the noise of the simulation makes it impossible to identify if a parameter set gave a better objective value due to randomness or due to actually being a better solution. The framework is able to find suitable calibration parameters within a reasonable timeframe but lacks any form of parallelism and early termination functionality.

2

Theory

We first give a brief overview of the theory behind the discrete element model and some of the most common model responses. We then discuss black box optimization with a brief survey of the most popular direct search methods as these lay the foundation for most other optimization algorithms. Finally we take a more in depth look at the surrogate optimization approach to solving black box optimization problems and some parallelism strategies.

2.1 The discrete element model

The discrete element method is a complex model and a comprehensive description is outside the scope of this thesis. As the main focus in this thesis is optimization of the calibration process we limit ourselves to a short overview of the method, with a special focus on the calibration process. The first version of the DEM was developed by Cundall and Strack in 1979 for studying rock mechanics problems [11]. In the discrete element model, each particle is modeled as a geometrical shape with a specified size. The particle-interactions, with each other and the surrounding geometry, are calculated by solving physical equations based on Newtons laws for motion and rotation. These equations determine the movement of the particles. In their most general form, Zhou et al [37] writes the governing equations for the translational and rotational motion of particle i with mass m_i and moment of inertia I_i as

$$m_i \frac{dv_i}{dt} = \sum_j \mathbf{F}_{ij}^c + \sum_k \mathbf{F}_{ik}^{mc} + \mathbf{F}_i^f + \mathbf{F}_i^g \tag{2.1}$$

$$I_i \frac{\omega_i}{dt} = \sum_j \mathbf{M}_{ij} \tag{2.2}$$

where v_i and ω_i are the translational and angular velocities of particle i, respectively, $\mathbf{F}_{ij}^c$ and $\mathbf{M}_{ij}$ are the contact force and torque acting on particle i by particle j or the walls, $\mathbf{F}_{ik}^{mc}$ is the non-contact force acting on particle i by particle k or other sources, F_i^f is the particle fluid interaction force on particle i, and F_i^g is the gravitational force. The simulation cases used in this thesis do not have non-contact forces or

particle fluid interactions, reducing the governing equations to

$$m_i \frac{dv_i}{dt} = \sum_j \mathbf{F}_{ij}^c + \mathbf{F}_i^g \tag{2.3}$$

$$I_i \frac{\omega_i}{dt} = \sum_j \mathbf{M}_{ij}. \tag{2.4}$$

If all the forces and torques are known, equations 2.3 and 2.4 can be computed numerically. For this physical models of the interactions between particles and between particles and material are needed. In Demify the main contact model used is the Hertz-Mindlin-Deresiewicz model [25].

The particles in a DEM simulation can be represented with different types of geometry. The most simple and efficient method is to model particles as spheres. However for many types of particles this is an oversimplification and does not yield accurate results. Some examples of more complex geometries are multi-spheres [22], polyhedrons [6] and dilated polyhedrons [20].

2.1.1 DEM model responses

Choosing the model response is an important step of the calibration process, it should be a measure that is possible to compute both in the simulations and in real world experiments. It should also be a measure that varies when the model parameters are varied and does not introduce to much variance. Here we give a general description of the model responses used in this thesis.

2.1.1.1 Angle of repose

The Angle of Repose (AoR) is a model specific response, that can be defined in different ways. In most cases it is defined as the steepest slope of the unconfined material, measured from the horizontal plane on which the material can be heaped without collapsing [4]. If the material is laying still in a pile the angle of repose is referred to as static. On the other hand if the material is continuously moving, for example in a rotating drum, the angle of repose that forms is referred to as dynamic, see Figure 2.1 for an example of a static angle of repose.

2.1.1.2 Mass flow rate

The mass flow rate is defined as the mass passing through an area per time unit. It is computed as

$$m_f = \frac{\Delta m}{\Delta t}$$

where Δt is the time unit and Δm is the mass passing through the area during that time.

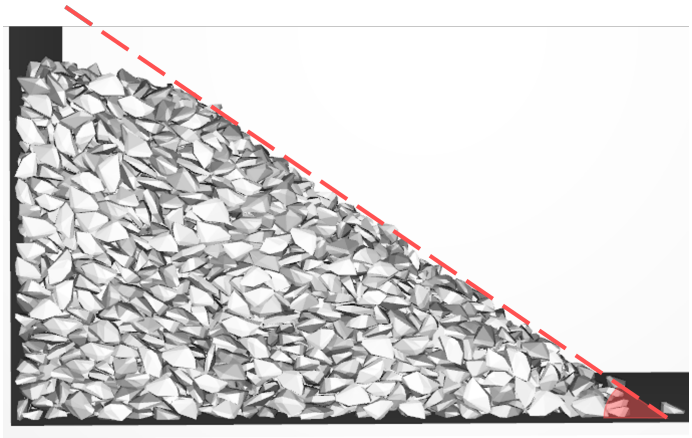


Figure 2.1: An example of the angle of repose model response.

2.1.1.3 Slide rate

In some DEM calibration setups the material is allowed to slide on a plane and the time it takes for some or all of the material to slide down is recorded. This measure generally provides a strong response for variations in particle-material friction.

2.1.2 Some typical calibration variables

As the goal of a DEM simulation is to mimic the physics of reality as well as possible, one wants to keep the number of variables that need calibration low. Many variables are known beforehand or can be measured accurately enough. Typical parameters that need calibration are different coefficients of friction such as particle-particle friction or particle-wall friction. Other examples are the coefficients of restitution and the timestep length used in the simulations.

2.2 Black box optimization

Optimization is the study of minimizing or maximizing mathematical functions, possibly subject to some constraints. An optimization problem is typically stated as

$$\min_x f(x) \ : \ x \in \Omega \tag{2.5}$$

where f is the function to be minimized over the variables x belonging to the constraint set Ω. When solving optimization problems, the nature of f and Ω is of high importance. Concepts such as linearity, convexity and monotonicity may be used to guide the solution process. Many solution methods are also based on utilizing derivatives of f. However there exists many cases where derivatives are not available and we are limited to evaluating $f(x)$ in our solution process. In other cases f can be considered a black box. Within optimization a black box is any process that when provided an input, returns an output, but the inner workings of the process are not analytically available [8]. Typically we are not able to assume any kind of properties of the objective function and the only way to gain information about the

problem is through evaluating the objective function, this makes the solution process challenging. Here we give an overview of some solution methods that have been developed for black-box optimization with a special focus on surrogate optimization methods.

2.2.1 Direct search methods

Direct search methods are a class of derivative free optimization methods that work from an incumbent solution point and look for better solutions from sets of trial points. If an improvement is found, the solution point is updated, if not, a new trial set is generated with decreased step size. These methods, while not used directly in this thesis, are still important to understand as they form the basis of most other black box optimization algorithm. We therefore dedicate a part of this section to an overview of the most popular ones. This will facilitate better understanding of the subsequent algorithms that are built upon them.

Compass Search (CS) or coordinate search was one of the first direct search methods introduced and lays a foundation for more complex algorithms [8]. It explores the neighborhood of an incumbent point by adding or subtracting the step size from each dimension. Formally, we start with an initial point $x_0 \in \mathbb{R}^n$ and a step size Δ. Let e_i be the ith coordinate vector of $\mathbb{R}^n$ and follow Algorithm 1 to generate iterates.

Algorithm 1 Compass search

Require: Initial point $x_0 \in \mathbb{R}^n$, initial step size Δ_0
  $k \leftarrow 0$
  while Termination criteria not met **do**
    Generate the trial set $P_k := \{x_k \pm \Delta_k e_i \ : \ i = 1, \ldots, n\}$
    if $f(y) < f(x_k)$ for $y \in P_k$ **then**
      $x_{k+1} \leftarrow f(y)$ and $\Delta_k \leftarrow \Delta_{k+1}$
    else
      $x_{k+1} \leftarrow x_k$ and $\Delta_{k+1} \leftarrow \frac{\Delta_k}{2}$
    end if
    $k \leftarrow k + 1$
  end while

It can be shown that Compass search converges to a stationary points if the objective function is locally strictly differentiable, a full convergence analysis is available in [8].

Compass search was generalized and combined with elements of other algorithms to form Generalized Pattern Search (GPS) class of algorithms [32]. In short, this method increased the flexibility of CS by not restricting the search directions to the coordinate vectors. It also allows the step size to increase or remain the same when an iteration succeeds in finding improvement and allowing exploration of a finite amount of points other than the trial points in every iteration [2]. This meant that it was possible to show convergence on a larger set of problems.

The pattern search algorithm was further generalized into the Mesh Adaptive Direct Search (MADS) class of algorithms. A key advantage with MADS is that the trial set is not limited to a finite set of directions and that it can handle general constraints [3]. Simplifying one could say that MADS enables a more flexible trial set than CS or GPS by extending the step size parameter into the trial set size parameters Δ_t and the mesh parameter Δ_m. The trial set is constructed with Δ_t limiting the distance between poll point and the current best solution, similarly to CS. The mesh density, i.e, the coarseness or fineness of points, is decided by Δ_m. This enables the trial points to be chosen on a more dense mesh than the one defined by just using Δ_t. For more details we refer to the original paper [3] or the overview [2]. The MADS algorithm is a popular choice to solve black box optimization problems today, and it is implemented in for example the Nomad software [1].

2.2.2 Surrogate optimization

If $f(x)$ is expensive to evaluate in addition to being a black box function the problem 2.5 is called a costly black box optimization problems. The methods described in section 2.2.1 usually requires many function evaluations to reach convergence. With costly black box problems, we are typically very limited in the number of function evaluations we can afford which can pose a problem. A frequently employed strategy for these kinds of problems is to utilize surrogate models. A surrogate model of the problem 2.5 is a function $\hat{f}(x)$, that shares some similarities with f but is faster to evaluate [8]. In its simplest form the surrogate optimization procedure follows the steps in Algorithm 2.

Algorithm 2 Surrogate optimization procedure

Step 1 (initial sampling): Select and evaluate a set of starting points S_0
while Some termination criteria not met **do**
  Step 2 (surrogate model): Create a surrogate model $s_n(x)$ from the data $\{(f(x_i), x_i)|x_i \in S_n\}$
  Step 3 (selection): Select a new evaluation point using $s_n(x)$ and some selection criteria
  Evaluate the new point, update $n = n + 1$ and go to Step 2.
end while

We now discuss the three different steps of the procedure in more detail.

2.2.2.1 Step 1: The initial sampling

The goal of the initial sampling phase is to find a set of points so that, if we evaluate f at these points, we obtain a good picture of the function scope. The set of points is created with an initial sampling strategy, also referred to as Design of experiment (DOE) It aims to cover the variable space as comprehensively as possible with the least number of points. Latin Hypercube Designs (LHD) are frequently used strategies that try to distribute the points evenly throughout the space [24]. Assume we want to sample n points of dimension $\mathbb{R}^d$ with values in the interval $[a, b]^d$. The range of each variable is divided into n intervals of equal probability.

Points are then sampled one at a time so that from each interval only one point is selected. More formally, following [33], a LHD can be interpreted as a set of n points $x_1, \dots x_n \in \{1, 2, \dots, n\}^d$ with the property that for each $j \in \{1, \dots, d\}$, the set $x_{1,j}, \dots, x_{n,j}$ is a permutation of $\{1, 2, \dots, n\}$. This means that for each dimension j no two of the values $x_{1,j}, \dots, x_{n,j}$ will be in the same interval. From this we can define a design matrix of the set $S = \{x_1, \dots, x_n\}$ as

$$M(S) = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,d} \\ x_{2,1} & x_{2,2} & \dots & x_{2,d} \\ \dots & \dots & \dots & \dots \\ x_{n,1} & x_{n,2} & \dots & x_{n,d} \end{pmatrix} \tag{2.6}$$

and say that S is a LHD if each column of $M(S)$ is a permutation of $\{1, 2, \dots, n\}$

Due to the randomness of the sampling process of LHD, some designs may be of higher quality then others. For higher dimensional problems there can be many designs generated and some kind of criteria is typically used to find an optimal design. An example is the maximin criteria from [21]. It evaluates designs by the the minimum distances between all points. The intuition being that a design that has points close to each other likely has a poor space filling. It then simply chooses the design with the largest minimum distance.

In an effort to improve the designs Ye. et al. introduced the Symmetrical Latin Hypercube Design (SLHD) [36]. SLHDs are regular LHDs but with the added constraint that if row i has values $(x_{i,1}, x_{i,2}, \dots, x_{i,n})$ then there must be a row j so that $(x_{i,1}, x_{i,2}, \dots, x_{i,n}) = (n+1-x_{i,1}, n+1-x_{i,2}, \dots, n+1-x_{i,n})$. It is also showed that the SLHD has better performance than regular LHD on the maximin measure discussed above and on an entropy measure [36].

2.2.2.2 Step 2: The surrogate model

The task of the surrogate model is to approximate the objective function with the information gathered from the initial sampling and completed function evaluations. Here we describe three of the most popular surrogate models used today.

Polynomials
Polynomial interpolation is a well studied method. In its simplest form a polynomial is a function

$$p(x) = \sum_{i=0}^{m} a_i x^i \tag{2.7}$$

where $a_i \in \mathbb{R}$. To find a polynomial that interpolates data $(x^1, y^1), \dots, (x^n, y^n)$ we solve the system

$$p(x^i) = y^y \quad \text{for all } i = 1, 2, \dots, n \tag{2.8}$$

which can be rewritten to the equation system $M_x a = b$ where

$$M_x = \begin{pmatrix} 1 & x^1 & (x^1)^2 & \dots & (x^1)^m \\ \vdots & & & & \vdots \\ 1 & x^n & (x^n)^2 & \dots & (x^n)^m \end{pmatrix} \quad a = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} \quad b = \begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^n \end{pmatrix}$$

The system is solvable if M has full rank and the resulting solution a give us coefficients for a polynomial that interpolates the data. Advantages of polynomial interpolation include simplicity and an abundance of software available, however the performance can be poor when compared to other interpolation models such as Radial basis functions [19].

Radial basis functions
A Radial Basis Function (RBF) surrogate model interpolates the target function by representing it as a linear combination of radial basis functions. The target function can be written as

$$\hat{f}(x) = \sum_{i=1}^{n} \lambda_i \phi(||x - x_i||_2) + p(x) \tag{2.9}$$

where $\lambda_1 \dots \lambda_n \in \mathbb{R}$ are the weights to be determined, $||\cdot||_2$ is the euclidean norm, $\phi(\cdot)$ the basis function and $p(x)$ is a low degree polynomial usually called the tail. Björkman and Holmström showed that using a cubic basis function with linear tail gives good results [17]. Thus equation 2.9 becomes

$$\hat{f}(x) = \sum_{i=1}^{n} \lambda_i \phi(||x - x_i||_2) + a^T x + b \tag{2.10}$$

and we can fit the surrogate model to the data by solving for λ, a and b in

$$\begin{pmatrix} \boldsymbol{\phi} & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix} \tag{2.11}$$

where $\boldsymbol{\phi}$ is the $n \times n$ matrix with $\boldsymbol{\phi}_{i,j} = \phi(||x_i - x_j||)_2$ and

$$P = \begin{pmatrix} x_1^T & 1 \\ x_2^T & 1 \\ \vdots & \vdots \\ x_n^T & 1 \end{pmatrix}, \quad \lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{pmatrix}, \quad c = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_d \\ a \end{pmatrix}, \quad F = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f_(x_n) \end{pmatrix}. \tag{2.12}$$

It has been shown that 2.11 has a unique solution and by implication that f has a unique RBF interpolant at the points $x_1 \dots x_n$ if $\text{rank}(P) = d + 1$ [26]. The system can be solved with for example an LU factorization [13].

Gaussian process
We give a short summary of the basics on Gaussian process interpolation and refer to [7] for more in depth descriptions. The basic idea of GPR is to compute the marginal likelihood distribution $p(y|x)$ that serves as an approximation for $f(x)$. This is done using Gaussian processes. A Gaussian process is a stochastic process where any finite number of random variables have a joint Gaussian distribution (also called multivariate normal distribution).

Specifying a mean function $\mu(x)$ and a covariance kernel $k(x, x')$ we can use a Gaussian process to define a distribution over the function we want to approximate f as

$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x')) \tag{2.13}$$

Popular choices for the kernel $k(x, x')$ are the squared exponential $k(x, y) = \frac{||x-y||^2}{2l^2}$ and the Matérn kernel. The marginal likelihood can then be computed using for example algorithm 2.1 in [7].

2.2.2.3 Step 3: Selecting a new evaluation point

The selection of the next evaluation point is a crucial step in the optimization process. It usually involves defining a merit function that assigns a score to candidate points based on the estimated objective function improvement and/or the information gained as well as defining a search region in which to maximize the merit function. An important concept is the trade off between gathering information with which to improve the surrogate, commonly referred to as exploration, and trying to improve the objective value, referred to as exploitation. The optimal merit function also depends on which surrogate model is used. A full review of different merit functions is outside the scope of this thesis. We present some common ones and refer the reader to [33] for a comprehensive guide. In section 3.5 a merit function is implemented based on the Metric response surface weighted score from Regis and Shoemaker [29]. It is designed to work with any surrogate. If a Gaussian process has been used as a surrogate model, it is common that merit functions are based on Bayesian optimization. Commonly used merit functions include the probability of improvement function (PI) where the probability that a new evaluation point x results in an improvement of the objective function is estimated. Another option is the expected improvement (EI) function. In EI we define the improvement of x as $I(x) = \max[y_{\min} - Y(x), 0]$ where $y_{\min}$ is the current best value and $Y(x)$ the predicted value of $f(x)$ by the surrogate. We then compute the expected value of $I(x)$. Finally a merit function connected to RBF surrogates is proposed in [14]. Here an estimated objective value T is assigned and the evaluation point is chosen as the point that make the RBF interpolation the most smooth. This is based on an observation that usually the most accurate RBF models are the most smooth.

This step usually also involves defining a region from which the candidate points to be evaluated by the merit function are selected. Typically this will be some neighbourhood around the best solution found or simply the whole domain. The search region can also serve as a balance between exploration and exploitation, and it is common to decrease the size of the region as the optimization progresses.

2.2.3 Parallelism and early termination

When solving costly black box optimization problems, the advantage of parallelism is clear. Per definition, the majority of computation time will be spent on evaluating the objective function and the time required for other tasks are negligible in comparison. This means that parallel algorithms are relatively simple to implement, one simply performs many evaluations at once, and that the overhead required for synchronisation has a very small effect on total computation time. Furthermore, modern computers and data clusters almost always include functionality for parallelism and taking advantage of this can greatly reduce the wall clock computation time needed to solve optimization problems.

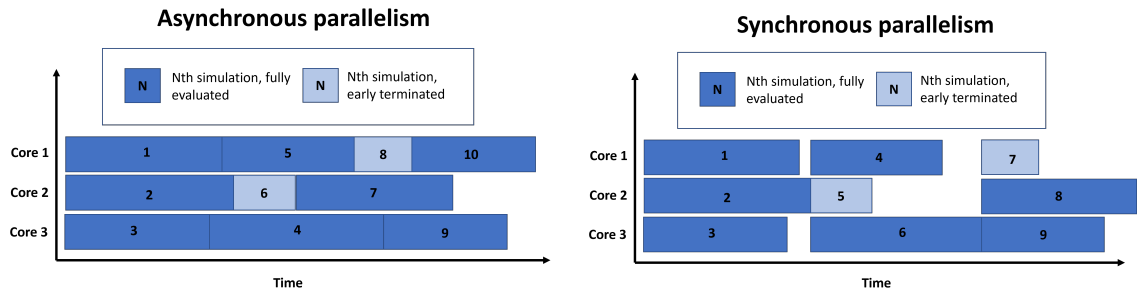


Figure 2.2: A comparison of asynchronous and synchronous parallelism with early termination.

Generally there are two approaches to parallel optimization: synchronous and asynchronous. A synchronous parallel program with P processors starts a batch of P tasks, typically function evaluations, and waits for all tasks to finish before starting a second batch of P tasks. An asynchronous program starts P tasks but does not wait for all tasks to finish. Instead the next task in the queue is immediately started as soon as a processor becomes free. For black box optimization problems, where function evaluations might vary greatly in time, an asynchronous approach will be much faster than a synchronous one.

Another way to speed up the solution process is to utilize early termination. In some black box optimization problems it might be possible to estimate or gain insight into the result of a function evaluation as it is progressing. In a DEM simulation an example would be that one or more model responses are computed before the simulation is completed. In this case one can terminate function evaluations that do not seem promising and save computational effort. Early termination is most useful in an asynchronous setting, see Figure 2.2 for a comparison.

One of the first popular asynchronous black box optimization algorithms is an extension of the pattern search class of algorithm described above called Asynchronous Parallel Pattern Search (APPS) [18]. The algorithm uses P processors to evaluate the objective function at P different points simultaneously using a *peer-to-peer* parallel paradigm. This mean that each processor functions as an independent unit, performing approximately the same algorithm as described in 1, while at the same time communicating with the other processors to update the current optimization parameters consisting of the best solution found x^*, best objective value f^* and step size Δ. In short, each processor has its own opinion on the value on these parameters, and when a processor finds a solution with better objective value it broadcasts this to the other processors. Before each evaluation, the processor checks its incoming broadcasts and updates accordingly, resulting in the best solution propagating over all processors as the algorithm proceeds.

3

An optimization framework for DEM calibration

Here we describe the main contribution of this thesis - an asynchronous parallel surrogate optimization framework adapted for the calibration of DEM models. We select a subset of the methods described in the theory chapter and use them in the corresponding steps to create a black box optimization algorithm. The connections are shown in figure 3.1

3.1 The optimization problem and overall framework

We begin by introducing the problem and some notation. Calibration refers to the process of tuning model parameters to achieve congruence between model response and some prior known reference data. We formulate this by solving the following optimization problem

$$\begin{aligned} \min_{x} \quad & f(x; r, p) \\ \text{s.t.} \quad & x \in X \end{aligned} \tag{3.1}$$

where f is the objective function that should be small when model response and reference data are closely matching and zero if they are equal, x are the model parameters we are tuning, r is the reference data and X is the variable space. Furthermore, p is meant to represent all other fixed parameters and model properties that affect the simulation but we do not have any control over as part of the op-

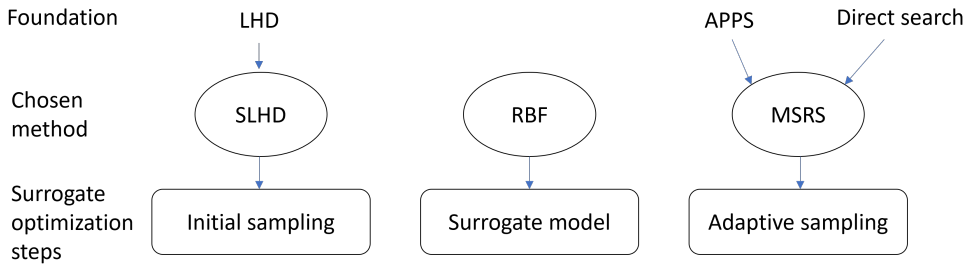


Figure 3.1: Diagram of the methods chosen for the different steps in the surrogate optimization procedure.

timization. All DEM simulations contain many modelling choices, such as particle geometry and contact models, and fixed physical parameters, such as Young's modulus and restitution, that influence the simulations but that are not practical to optimize over. It is important to keep these in mind when calibrating the model, and changing p will likely change the solution to the calibration problem. In the objective function, p manifests as uncertainty and noise. Finally, another important concept is the model response from running a DEM simulation with the variables x. We denote the model response as follow

$$y = (y_1, y_2, \ldots, y_m) = (\varphi_1(x), \varphi_2(x), \ldots, \varphi_m(x)) \tag{3.2}$$

where $y_1, y_2, \ldots, y_m$ and $\varphi_1(x), \varphi_2(x), \ldots, \varphi_m(x)$ are scalar values.

Each evaluation of f corresponds to running a DEM simulation. As previously stated, these typically run over many hours or even days. Furthermore we do not have any explicit formulations of f, no ways to obtain derivatives and cannot make any assumptions about properties of f. Finally we suspect that due to p and randomness in the DEM simulations, f might contain noise and many local minima. Thus the problem 3.1 falls clearly into the realm of black box optimization problems. The variables space X will typically be a hyper-rectangle since x_i is typically set to be within some interval $[a, b]$. It is also worth noting that while calibration is necessary for DEM simulations, one still wants to capture the real physics of the model as close as possible and not rely on calibration for many of the model parameters. This leads to the problem generally being no more than three or four dimensional.

The rest of this chapter is dedicated to the implementation of an optimization framework to solve 3.1, with the goal to find x^* that achieves near equality between model response and reference data. As a starting point, we illustrate the different steps of the optimization framework in figure 3.2. The first step is to create an initial sampling of points that are evenly distributed over the variable space. This is used to build our first surrogate model and to start the evaluation process. We then iterate in the following way

1. Sample many (>100) points from the surrogate model
2. Choose one of these points based on some selection criteria
3. The chosen point is then evaluated by running the DEM simulation with the specified parameters. The objective function is calculated and used to train the model.

The above procedure is called the Adaptive Sampling Strategy and is repeated until some termination criteria is reached, typically a limit on the number of evaluations or a threshold for the objective value. When the algorithm terminates, the point which gave the lowest objective value is picked as the solution. The procedure is described in detail in section 3.5.

For implementation, we use a python package called pySOT: Python Surrogate Optimization Toolbox [13]. It is an asynchronous parallel optimization toolbox especially made for solving costly black-box optimization problems. It contains implementations for initial sampling strategies, surrogate models and optimization strategies as well as support for parallelism.

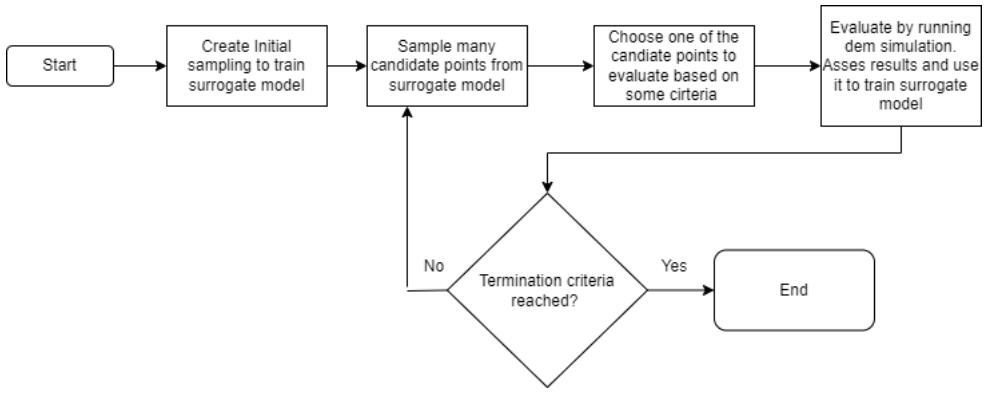


Figure 3.2: An overview of the optimization process for calibrating DEM models.

3.2 Initial sampling strategy

In pySOT, there is built-in support for both regular and symmetric Latin hypercube design. Since our problem is low-dimensional, there is no significant difference between the methods. We choose SLHD based on its theoretically better performance. The SLHD should consists of at least $2d+1$ points, where d is the dimension of the problem, to ensure that the design has full rank. As stated in the Theory section, there is randomness in the generation of designs. To ensure that the best possible design is used, we generate 1000 designs and use the maximin criteria to determine the best one.

3.3 Objective function

The task of the objective function is to give us a measure of how "good" a particular set of variables x with "good" being defined as the model response matching the reference data. We want the objective function f to be small if the model response $y = (y_1, \ldots, y_m)$ is close to the reference data $r = (r_1, \ldots, r_m)$, and large otherwise, and to evaluate this we need a distance metric. Common distance metrics for scalar values are absolute errors calculated as $|y_i - r_i|$ and relative error calculated as $|\frac{y_i - r_i}{r_i}|$. In our case the model responses will often be differently scaled, and we choose to use relative errors since they are automatically normalized. We let η_i denote the relative errors of model response i. The individual distances are then combined into a single value using a weighted sum of squares. We get

$$f(x; r, p) = \sum_{i=1}^{m} w_i \eta_i^2 = \sum_{i=1}^{m} w_i \left(\frac{y_i - r_i}{r_i} \right)^2. \tag{3.3}$$

The weights w_i can all be set to 1 if one wants all responses to be weighed equally, or constrained to $\sum_{i=1}^{m} w_i = 1$. In most applications of this framework we can assume $r_i \neq 0$ without loss of generality. If applied to a case where we have reference data $r_i = 0$ we can use the absolute error and the weights w_i to achieve normalization of the different errors.

It is worth mentioning that the thesis and the optimization framework does not

consider multi-objective optimization problem. This is based on an assumption that there exists a solution which is a global minimum for all the relative errors of the model responses for a DEM calibration. The assumption is reasonable since the reference data typically originates from real world experiments and we want the DEM simulation to capture reality as close as possible, meaning that if the DEM model is setup correctly there should be a combination of parameters that achieve perfect congruence between model response and real world reference data. It is, needless to say, impossible to perfectly mimic reality in practice, however we can still work with that assumption.

In some cases DEM calibration can become a multi-objective optimization problem, specifically if one tries to minimize the computational time while keeping model accuracy high. An example is given by Do et al. where a clear Pareto front between reducing the simulation time-step and minimizing the simulation error can be seen [12]. These problems are not the main target for this framework and not considered further. However, it is worth noting that one could implement two conflicting model responses in the objective function of this framework and achieve similar pareto front by varying the weights.

3.4 The surrogate model

To decide an appropriate surrogate model, a test involving approximating an actual objective function from a DEM calibration test was done. Model responses from the calibration device described in section 4.1 were sampled on a grid of 40 points to be used as data. The reference data r was arbitrarily set to the model responses obtained for some chosen parameters and the objective function calculated according to 3.3. Surrogate models based on polynomials of degree 2-4, radial basis functions with a cubic kernel and linear tails and Gaussian process regression with squared exponential kernel were created, all according to the description in section 2.2.2.2. Each model was trained 10 times on randomly sampled subset of the data points, with the number of points starting at 10 and increasing to 35. The absolute error of each model was calculated and shown in figure 3.3. It is clear that both GP and RBF surrogate models outperform polynomials by a wide margin. We choose to use an RBF surrogate based on it's slightly better performance and since it is the most common surrogate models to use with the optimization strategy implemented in section 3.5. The RBF surrogate model is implemented with pySOT.

3.5 Adaptive sampling strategy

The adaptive sampling strategy is the core of the optimization framework. After the initial sampling phase is complete, the adaptive sampling strategy decides which point to evaluate next. It utilizes the surrogate model and previously evaluated points to balance exploration and exploitation and find a good solution in as few iterations as possible. We use a strategy called *Metric Stochastic Response Surface Method* (MSRS) with RBF as response surface, the strategy was created by Regis and Shoemaker and the rest of this section will be based on their paper [29].

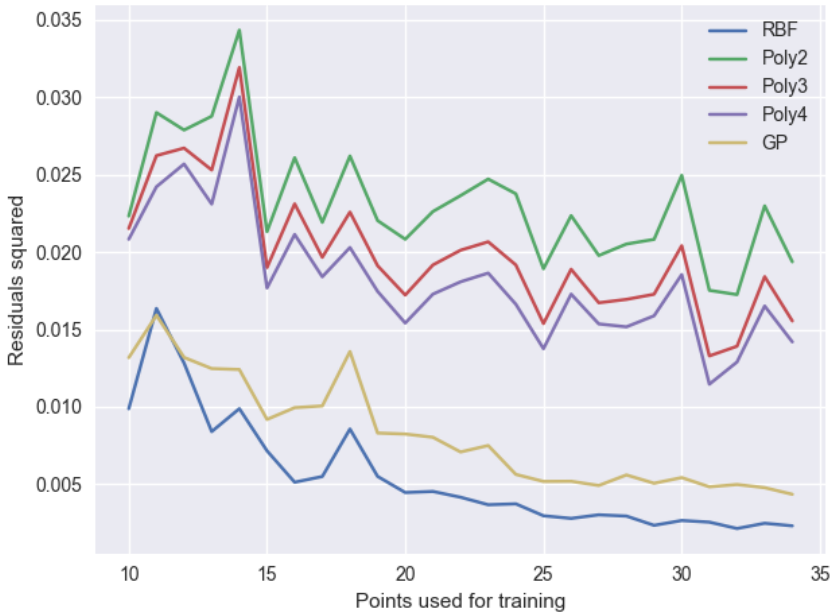


Figure 3.3: Squared error of different surrogate models with increasing number of training points.

3.5.1 Serial MSRS

Let n be the number of previously evaluated points, A_n the set of previously evaluated points and $s_n(x)$ the response surface model after n evaluations. The algorithm, adapted from Regis and Shoemaker, is described in Algorithm 3. By setting some conditions on the candidate points it can be shown that this general framework converges almost surely to an optimal solution, see [29] section 3. We now describe how the candidate points are selected and how an evaluation point is chosen.

3.5.1.1 Selecting evaluation point

To choose the evaluation point, each candidate point is given a score between 0 and 1 based on a sum of two weighted criteria, where a more desirable point is given a score close to 0. The first criteria can be referred to as the *Surrogate criteria*, where candidate points objective value is estimated with the surrogate model, and the second as the *Distance criterion*, where the minimum distance from previously evaluated points is computed. The two criteria are rescaled to give a score within [0,1] and then summed with the weight λ representing whether focus should be on exploration or exploitation. Following [29], we can replace step 2.3 in Algorithm 3 with Algorithm 4.

3.5.1.2 Generating candidate points

In the original paper, Regis and Shoemaker introduce two methods for generating the candidate points. A global optimization algorithm, where the points are generated uniformly throughout the variable space, and a local algorithm, where they are generated as perturbations of the current best solution. The local version shows

Algorithm 3 The Serial optimization algorithm

Require:
f as defined in 3.1
An RBF response surface model
A set of initial sampling points $J = \{x_1, \dots x_{n_0}\}$
The number of candidate points in each iteration t
The maximum number of evaluations $N_{\max}$

Step 1 *(Create initial design)* Evaluate f at each point in J. Set $n = n_0$ and $A_n = J$. Let x_n^* be the point in A_n with the best function value.
while $n < N_{\max}$ **do**
Step 2.1 *(Update the surrogate model).* Update the surrogate model $s_n(x)$ with the data points in $B_n = \{(x_i, f(x_i)) : i = 1, \dots, n\}$
Step 2.2 *(Generate candidate points).* Generate t candidate points $\Omega_n = \{y_{m,1}, \dots, y_{n,t}\}$.
Step 2.3 *(Select the next evaluation point).* Use information from the surrogate $s_n(x)$ and the points B_n to select evaluation point x_{n+1} from Ω.
Step 2.4 *(Do costly function evaluation).* Evaluate the function f at x_{n+1}
Step 2.5 *(Update information.)* Update $A_{n+1} = A_n \cup \{x_{n+1}\}$ and $B_{n+1} = B_n \cup \{(x_{n+1}, f(x_{n+1}))\}$. Let x_{n+1}^* be the point in A_{n+1} with the best function value. Set $n = n + 1$.
end while
Step 3 *(Return the best solution found).* Return $x_{N_{\max}}^*$

Algorithm 4 Selection of evaluation point

Require: : $\lambda \in [0, 1]$, candidate points in iteration n denoted by Ω_n and a distance metric $D(x, y) \in \mathcal{R}^n$
Step 2.3 a) *Compute the surrogate criteria.* For each $x \in \Omega_n$, compute $s_n(x)$. Also, compute $s_n^{\max} = \max\{s_n(x) : x \in \Omega_n\}$ and $s_n^{\min} = \min\{s_n(x) : x \in \Omega_n\}$
Step 2.3 b) *Compute the distance criteria.* For each $x \in \Omega_n$, compute $\Delta_n(x) = \min_{1 \le i \le n} D(x, x_i)$. Also compute $\Delta_n^{\max} = \max\{s_n(x) : x \in \Omega_n\}$ and $\Delta_n^{\min} = \min\{s_n(x) : x \in \Omega_n\}$
Step 2.3 c) *Compute the score for the surrogate criterion.* For each $x \in \Omega_n$, compute $V_n^R(x) = (s_n(x) - s_n^{\min})/(s_n^{\max} - s_n^{\min})$ if $s_n^{\max} \neq s_n^{\min}$ and $V_n^R(x) = 1$ otherwise.
Step 2.3 d) *Compute the score for the distance criterion.* For each $x \in \Omega_n$, compute $V_n^D(x) = (\Delta_n^{\max} - \Delta_n(x))/(\Delta_n^{\max} - \Delta_n^{\min})$ if $\Delta_n^{\max} \neq \Delta_n^{\min}$ and $V_n^D(x) = 1$ otherwise.
Step 2.3 e) *Compute the weighted score.* For each $x \in \Omega_n$, compute the score $\mathcal{F}_n(x) = \lambda V_n^R(x) + (1 - \lambda) V_n^D(x)$
Step 2.3 f) *Select the next evaluation point.* Let x_{n+1} be the point in Ω_n that minimizes $\mathcal{F}_n$

superior performance and is the one we choose to implement. The candidate points are generated by perturbing of the current best solution x_n^* with distances that are normally distributed with mean 0 and covariance matrix $\sigma_n^2 I_d$. We refer to σ_n as the step-size. The step-size is set to an initial value and then adjusted over the optimization. It is doubled if we have C_{success} consecutive evaluations that improve the objective value, and halved if we have C_{fail} consecutive iterations where we have not improved the objective value. There is limit on the minimum value $\sigma_{\min}$ and the maximum value $\sigma_{\max}$ of the step size. As the optimization progresses, the step-size will become smaller as we approach an optima and it becomes harder to find better values, thus facilitating convergence. There is also a restart function available, meaning that if the step size becomes too small, we can restart the algorithm from scratch. The reason is to not get stuck in a local minima. However for the problems in this project, the number of evaluations is so few that we cannot afford to restart the algorithm and we therefore do not make us of this functionality.

3.6 Early termination

The framework also include functionality for terminating function evaluations that do not seem promising. This can save computational resources and increase the speed of the calibration process.

For early termination to be applicable, we assume some information about the progress can be obtained before the function evaluation is complete. For example, if a function evaluation consists of running many DEM-simulations, we can look at the relative errors of the first run to decide whether it is worth evaluating the rest. Another possibility is to look at the relative error of some model response that is obtained before the simulation is complete. It is worth noting that during the initial sampling phase, early termination is not applied, since the goal of the phase is to gather as much information as possible. There are two main considerations when implementing early termination:

- What is the criterion for terminating an evaluation?
- How should a terminated evaluation be handled in the optimization process?

3.6.1 Early termination criteria

When deciding whether to terminate an ongoing evaluation, one must balance the computational cost of continuing the evaluation against the benefit of incorporating the information gained from it into the surrogate model. Early in the optimization process, this information may be more valuable, while later on, the surrogate model may already be sufficiently accurate. Therefore, it is essential to be able to adjust the termination criteria throughout the optimization process. We base our criteria on a comparison between the obtained model response and the current best objective value. Let η_{et} be the relative error of the early model response, f^* be the current best solution and $c \in \mathbb{R}$. We terminate an evaluation if

$$\eta_{\text{et}} > c \cdot f^*. \tag{3.4}$$

The factor c should be large at the start of the optimization run and decreased as the optimization proceeds. The exact implementation is case-specific, mostly depending on the uncertainty level of the model responses, and a user can modify it to suit their needs. The default setup is to let c follow a pattern of decreasing values as the number of iterations increase. During the first half of iterations, after the initial sampling has been completed let $c = a$ for some constant a. During the third quarter of iterations let $c = \frac{a}{2}$ and let $c = \frac{a}{4}$ for the remainder of iterations. As an example when applied to the calibration device in section 4.1 we let c follow the pattern $[10, 5, 2.5]$. In general, a lower bound for c should be such that we should only terminate evaluations in cases where we are sure that the resulting objective value will be worse. For example, if we set $w_i = 1$ for all i in the objective function 3.3, a value of $c = 1$ means that we only terminate runs which are guaranteed to give a smaller objective value (since $\mathrm{f} = \sum_i \eta_i \geq \eta_{\mathrm{et}}$).

3.6.2 Handling terminated runs

There are three options of how to handle a terminated evaluation and a corresponding solution x_i. The first is to simply regard it as a point that has never been evaluated. In algorithm 3 Step 2.5 this corresponds to letting $A_{n+1} = A_n$ and $B_{n+1} = B_n$. This has the obvious disadvantage of having spent computational resources to evaluate x_i but not obtaining any information. Additionally there is a risk that x_i is chosen again as an evaluation point later. Another option would be to try to estimate $f(x_i)$ and then letting $A_{n+1} = A_n \cup \{x_i\}$ and $B_{n+1} = B_n \cup \{(x_i, \widehat{f(x_i)})\}$. This raises the question of how to estimate $f(x_i)$ and the risk of adding poorly estimated values to the surrogate model.

Therefore, we choose the third and final option where x_i is used to update the distance information $A_{n+1} = A_n \cup \{x_i\}$, but not the surrogate information $B_{n+1} = B_n$. This means that the algorithm remembers that x_i was evaluated and will likely not choose new points close to x_i, but does not pose a risk of adding poor values to the surrogate. The lost information from x_i is typically not of high importance as early terminations occur mostly at later evaluations where the surrogate model should already be accurate and the focus is on finding good solutions.

3.7 Asynchronous parallelism

Section 2.2.3 highlights how parallelism can significantly reduce the computation time for black box optimization problems, as well as the importance of asynchrony in the case of varying simulation times. In this framework, we employ asynchronous parallelism, which allows for multiple objective function evaluations (i.e. DEM simulations) to be conducted simultaneously. Since function evaluations require a vast majority of the computational effort, the time spent updating the surrogate and determining the next evaluation point is negligible in comparison. This in combination with asynchrony, which ensures that no process remains idle while waiting for others to complete, results in a close to linear speedup with the number of processes.

Regis and Shoemaker describe a synchronous parallel version of their SMRS algo-

rithm in [31]. Here we make the algorithm asynchronous instead according to the framework provided by Algorithm 3 in [13]. A master-worker framework is utilized where the master task involves building the surrogate model, choosing points for the workers to evaluate and checking for termination. The P number of worker tasks consists solely of evaluating the objective functions and checking for early termination. The parallel algorithm is very similar to Algorithm 3. We start with an initial sampling strategy based on SLHD with the addition that points are now evaluated in parallel instead. A potential problem occurs when transitioning between the initial and adaptive sampling strategy: Assume we need to evaluate q points before building the surrogate and proceeding to the adaptive phase. If $q-1$ workers finish their evaluation quickly they need to wait for the last point to be completed before proceeding which results in idle time. This is solved by including $q+P$ points in the initial design, meaning that when all points in the initial design are either completed or under evaluation, we are guaranteed to be able to proceed to the next phase. The adaptive sampling phase is almost the same as the serial version, with the exception that the weight w is cycled between some common values to achieve a balance of exploration and exploitation.

Algorithm 5 presents the asynchronous version of SMRS, including the early termination method described in Section 3.6. In this implementation, we maintain a queue of evaluations and assign points from the queue to any available worker. After an evaluation is completed, we update the surrogate model, distance information, and, if necessary, the best point found. If the queue is empty, we generate new points to evaluate using the adaptive sampling method. A worker evaluation consists of running the DEM simulation with the specified parameters, checking for early termination and returning the results. This means that the communication required between the master and the worker is minimal which simplifies the implementation. A master only has to send the variable values to be evaluated to the worker, and the worker only has to send the objective value back. The workers are also responsible for keeping track of an eventual early termination functionality, if the worker terminates its run it tells this to the master. The algorithm is designed to be ran on a cluster with workers corresponding to different compute nodes that communicate with each other via MPI. It is implemented through the framework provided by pySOT and the closely related package POAP. POAP provides python classes for the master and workers as well as a framework for MPI communication. These were implemented and fitted to the optimization algorithm as a part of the framework creation.

Algorithm 5 The asynchronous parallel optimization algorithm with early termination

Require:

f as defined in 3.1
An RBF response surface model
A set of initial sampling points $J = \{x_1, \dots x_{n_0}\}$
The number of candidate points in each iteration t
The maximum number of evaluations $N_{\max}$
The number of workers P

▷ Create initial design
Add each point in J to the queue. Set $n = n_0$ and $A_n = J$.
▷ Launch initial evaluations
for each worker **do**
    Pop point from queue and dispatch to worker
end for
▷ Main algorithm
while $n < N_{\max}$ **do**
    if evaluation completed **then**
        if evaluation point early terminated **then**
            Update $A_{n+1} = A_n \cup \{x_{n+1}\}$
        else
            Update $A_{n+1} = A_n \cup \{x_{n+1}\}$ and $B_{n+1} = B_n \cup \{(x_i, f(x_i))\}$
            If $x_{n+1} > x_n^*$ set $x_{n+1}^* = x_{n+1}$ other wise set $x_{n+1}^* = x_n^*$
        end if
    end if
    if worker ready **then**
        if evaluation queue empty **then**
            Build/Update the surrogate model $s_n(x)$ with the data points in
            $B_n = \{(x_i, f(x_i)) : i = 1, \dots, n\}$.
            Generate t candidate points $\Omega_n = \{y_{m,1}, \dots, y_{n,t}\}$.
            Select point based on Algorithm 4 and add to queue.
        end if
        Pop point from queue and dispatch to worker
    end if
end while
Return best solution $x_{N_{\max}}^*$

4

Application

The framework described in Chapter 3 is applied to two DEM models, the first involving a calibration problem and the second an industrial optimization problem.

4.1 Case 1: Calibration Device

The following model is a calibration device created by Quist and Evertsson [**?**], existing both as a digital DEM model and a physical device at FCC. It is used to calibrate the particle-particle and particle-steel friction of a granular material by first letting the material flow through the real world device and recording reference data. Then the DEM simulation is run with varying friction coefficients until the DEM model response matches the reference data. This case is relatively computationally cheap, with simulation times around 40 minutes on a Nvidia GTX 1650, and is therefore used as a testing setup and as a proof of concept for the algorithm. The device can be seen in Figure 4.1 and a simulation consists of the following steps:

1. The hopper is filled with material while the trapdoor is closed
2. The trapdoors open and the material flows out. The massflow rate out of the hopper is recorded.
3. The material slides on the sliding plane. The time from trapdoor opening until all material has slid past the plane is recorded
4. The material forms a pile in the bottom left corner. The angle of repose is recorded

The recorded massflow rate, sliding time and angle of repose make up the model responses y_1, y_2, y_3 used in this calibration case. The massflow is calculated by recording the mass above the hopper 10 times per second and fitting a slope to the data. The slidetime is calculated by recording the time when there is less than 0.01 kg of mass on the slide. In some cases, e.g, if the friction between the slide and material is very high, more than 0.01 kg of mass will remain on the slide when the simulation ends. In this case we set the slide time to

$$\text{simulation end time} + 10 \cdot \frac{\text{mass remaining on slide}}{\text{total mass}}$$

where the addition of the second term ensures that there is still a model response for differing but very high values of friction. Finally the angle of repose is calculated

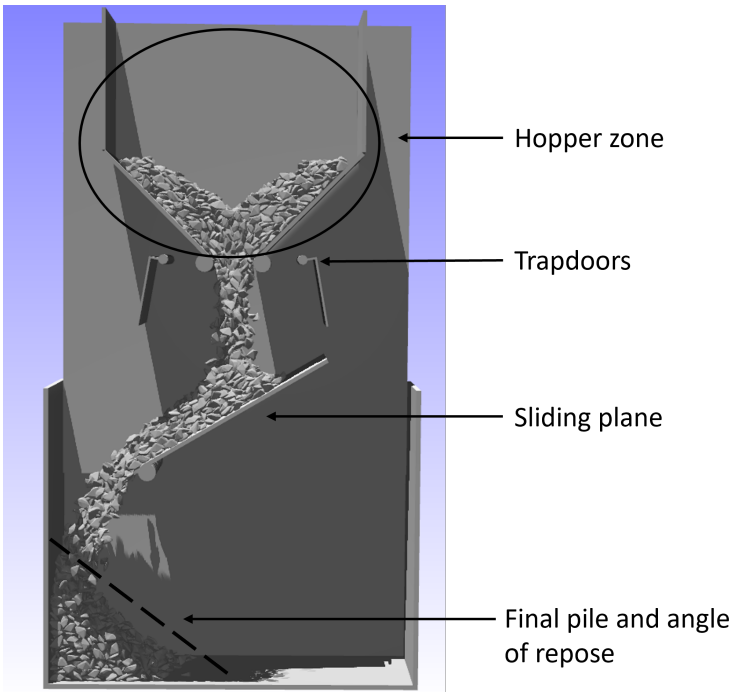


Figure 4.1: Components of the calibration device.

by fitting a sigmoid curve to the pile of material and taking the angle at the curves inflexion point, see Figure 4.2 for reference.

To reduce uncertainty the model can be run several times and the final model responses taken as an average. The objective functions can be written as

$$f(x;r,p) = \left(\frac{\bar{y_1} - r_1}{r_1}\right)^2 + \left(\frac{\bar{y_2} - r_2}{r_2}\right)^2 + \left(\frac{\bar{y_3} - r_3}{r_3}\right)^2. \tag{4.1}$$

where $\bar{y_1}, \bar{y_2}, \bar{y_3}$ denotes the mean responses of n runs, i.e $\bar{y_i} = \sum_{j=1}^{n} y_{i_j} \cdot \frac{1}{n}$ with corresponding reference data r_1, r_2, r_3.

There are two options for how to implement early termination depending on the value of n. If $n = 1$ we can use the massflow model response y_1 as an early termination criteria as it is obtained before the other responses, about halfway into the simulation. We let $\eta_{et} = (\frac{y_1 - r_1}{r_1})^2$ and use the criterion described in equation 3.4. If $n > 1$ we can instead use the results of the first simulation as a criteria by calculating the summed relative error for that run and setting it as η_{et}.

The parameters used for the particles in the simulation are presented in Table 4.1. The material for all non-particle objects is steel and has parameters set accordingly. The calibration parameters are μ_{pp} and μ_{ps}. These are varied in the interval [0.2,0.9]. Particles are randomly generated during each run according to a pre-specified size distribution until the total mass of the system is 3.5 kg. This means that the number of particles varies around a mean of 4300.

The framework is applied to the simulation case with parameters as described in Table 4.2. Most are set to the default values provided in the article by Regis and Shoemaker [31]. The weights for the objective function are all set to 1, meaning that

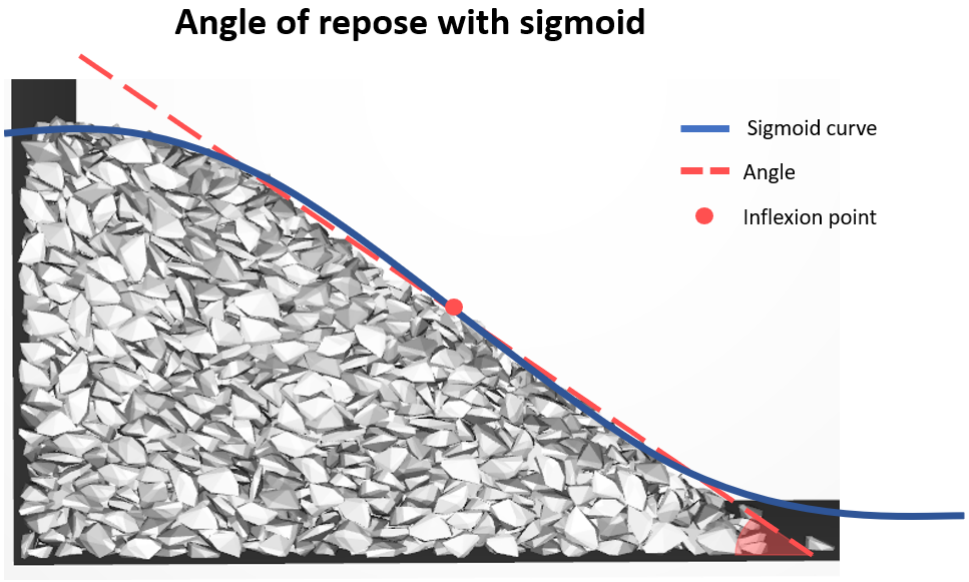


Figure 4.2: The AoR is set as the angle of the sigmoid curve at the inflexion point represented by the red dot.

Property	Symbol	Unit	Value
Contact model	-	-	HMD (Hertz-Mindlin-Deresiewicz)
Particle geometry	-	-	Dilated polyhedron
Particle density	ρ	kg/m^3	2750
Young's Modulus	E	GPa	1
Poisson's ratio	ν	-	0.25
Restitution (particle-particle)	e_{pp}	-	0.2
Restitution (particle-steel)	e_{ps}	-	0.2
Friction (particle-particle)	μ_{pp}	-	[0.2,0.9]
Friction (particle-steel)	μ_{ps}	-	[0.2,0.9]
Time-step	Δt	s	8.0E-06

Table 4.1: DEM parameters used in simulation case 1.

Parameter	Symbol	Value
Dimension	d	2
Variables	$\mu_{p,p}, \mu_{p,s}$	[0.2,0.9]
Max evaluations	$N_{\max}$	40
Number of workers	P	Varying 1-4
Initial sampling points	n_0	$5 + P$
Candidate points in each iteration	t	200
Step-size max	$\sigma_{\max}$	0.2
Step-size min	$\sigma_{\min}$	$\frac{\sigma_{\max}}{2^6} = \frac{1}{320}$
Step-size increase criteria	C_{success}	3
Step-size decrease criteria	C_{fail}	4
Merit function weight pattern	w	[0.3, 0.5, 0.8, 0.95]
Early termination factor patter	c	[10, 5, 2.5]

Table 4.2: Optimization parameters used for simulation case 1.

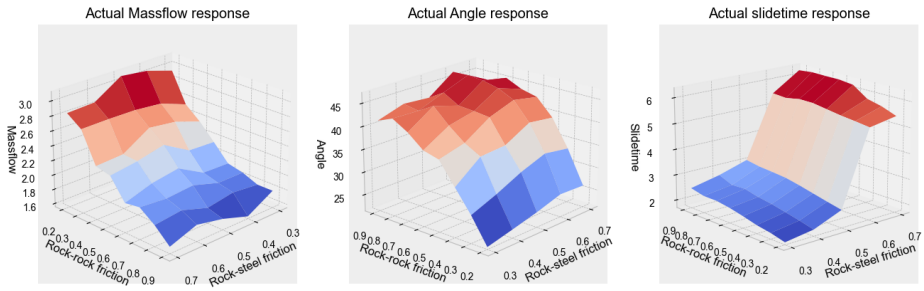
Reference friction $(\mu_{p,p}, \mu_{p,s})$	Reference response (y_1, y_2, y_3)
(0.4,0.4)	(2.372,36.948,2.223)
(0.55,0.55)	(2.157, 45.485, 3.765)
(0.67,0.67)	(1.924507,45.256,6.092)

Table 4.3: Reference friction and responses.

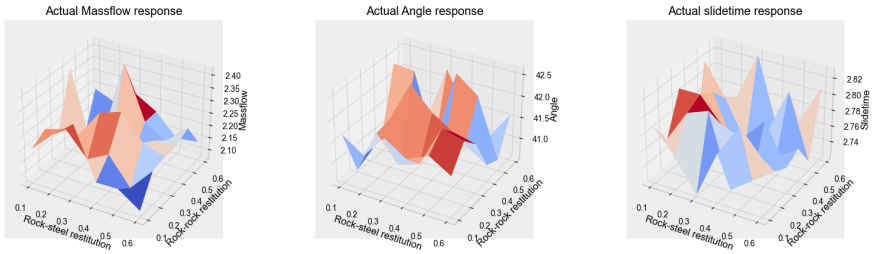
all models responses are weighted equally. To find the reference model responses r there are two options. The first options is that we perform the physical experiment that the simulation case is based on and manually measure the model responses by hand. This has the advantage of being similar to how most DEM simulations are calibrated today and to how FCC might use the framework, but the disadvantage of introducing human error into the reference data and in being time consuming. It was decided that the above method was outside the scope of this project and option two was chosen. Consisting of gathering reference data by running the model with a set of fixed friction parameters many times and taking an average. This also has the advantage of being able to evaluate the framework better, as we know from the start what the optimal solution should be. We gather reference data from three different parameter sets corresponding to low, medium and high friction in the system, to be able to assess the framework in a large area of the parameter space. The reference friction values and model responses used are shown in table 4.3, the responses are taken as an average of 10 simulations. The optimization was run on the C3SE Vera cluster with a varying number of workers depending on availability. The number of workers has no significant effect on the results.

4.2 Evaluation and sensitivity analysis

Before the optimization is applied, we verify that the simulation case is suitable for calibration of the friction coefficients. We study this by varying the friction values over a linearly spaced meshgrid, running the simulation for each combination of



(a) Actual responses for varying frictions $\mu_{p,p}, \mu_{p,s}$



(b) Actual responses for varying restitution $e_{p,p}, e_{p,s}$

Figure 4.3: Response surfaces for the calibration device when varying parameters over a linearly spaced grid with 40 points.

friction values and save the response. Since the solution space is in $\mathbb{R}^2$ we can make surface plots of the model responses and, for a given value of the reference data r, the objective function. These surface plots and the corresponding surrogate models are shown in Figure 4.3. It is clear that varying the friction parameters gives different responses in the model. We note that the massflow has a strong response to varying $\mu_{p,p}$ while the angle of repose and slidetime have strong responses to varying $\mu_{p,s}$. To provide context we also show surface plots with varying coefficients of restitution, which does not give a strong model response. In Figure 4.3b, all that is visible is the noise in the model.

The calibration problem of any DEM case contains noise and uncertainties. To be able to assess whether an optimization algorithm converges correctly it is therefore vital to have a sense of what the uncertainties are and the limitations of the calibration case. Some of the causes of uncertainty in this case are presented in Figure 4.4. Modelling uncertainties refer to modeling choices such as the contact model and particle shapes used and the fact that computations are run on the GPU. These are are only relevant if the reference data comes from real world experiments. Simulation uncertainties refer to how the particles are randomly generated each run. This means that the initial particle positions, number of particles, and sizes of particles will vary each run. Finally the estimation of model responses contain noise from numerical approximations such as the sigmoid and from simulation randomness such

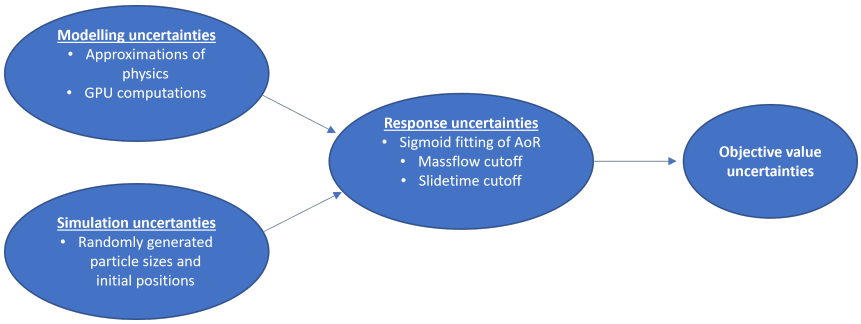


Figure 4.4: Cause of uncertainties in the simulation case.

	Massflow	Angle of repose	Slidetime
Mean	2.1928	42.744	2.896
Std	0.0928	0.9619	0.0738
Std/Mean	0.0423	0.0225	0.0254

Table 4.4: Mean and standard deviation of model responses over 50 simulations with $\mu_{\text{pp}} = 0.5$ and $\mu_{\text{ps}} = 0.5$.

as varying massflow over time.

The uncertainty is assessed by running the model 50 times with identical parameters with the results presented in table 4.4. Most relevant is the standard deviation divided by mean, showing a standard deviation of around 4% for massflow and 2% for the other responses. This provides a limit of how close to an optimal solution we can get, which we will denote as the uncertainty range. Within the uncertainty range the responses from different parameter combinations is mostly random, and there is more than one parameter combination that can give the wanted response. This means that we cannot be sure if a better objective value was due to the randomness, or due to actually finding better calibrated parameters. Visually we can think of the uncertainty as the objective function surface being more rough with many local optima. As much of the uncertainty stems from the random generation of particles in each simulation, the simplest way to reduce it is to run the simulation multiple times. This will smooth out the objective function and decrease the uncertainty range. In the results section we compare the results of just running one DEM simulation per evaluation, meaning $n = 1$, with letting each function evaluation correspond to five simulations, meaning $n = 5$.

4.3 Case 2: Transfer chute material divider

In DEM simulations, optimization might be used to solve more problems than just calibration of the models. A calibrated model will typically be used to optimize or calibrate some industrial machine. This can also be formulated as a black box optimization problem. As an example we apply the algorithm on a DEM case involving an industrial machine called a Transfer chute, designed to split a flow of material into two equally sized pools. The case is four dimensional and serves as

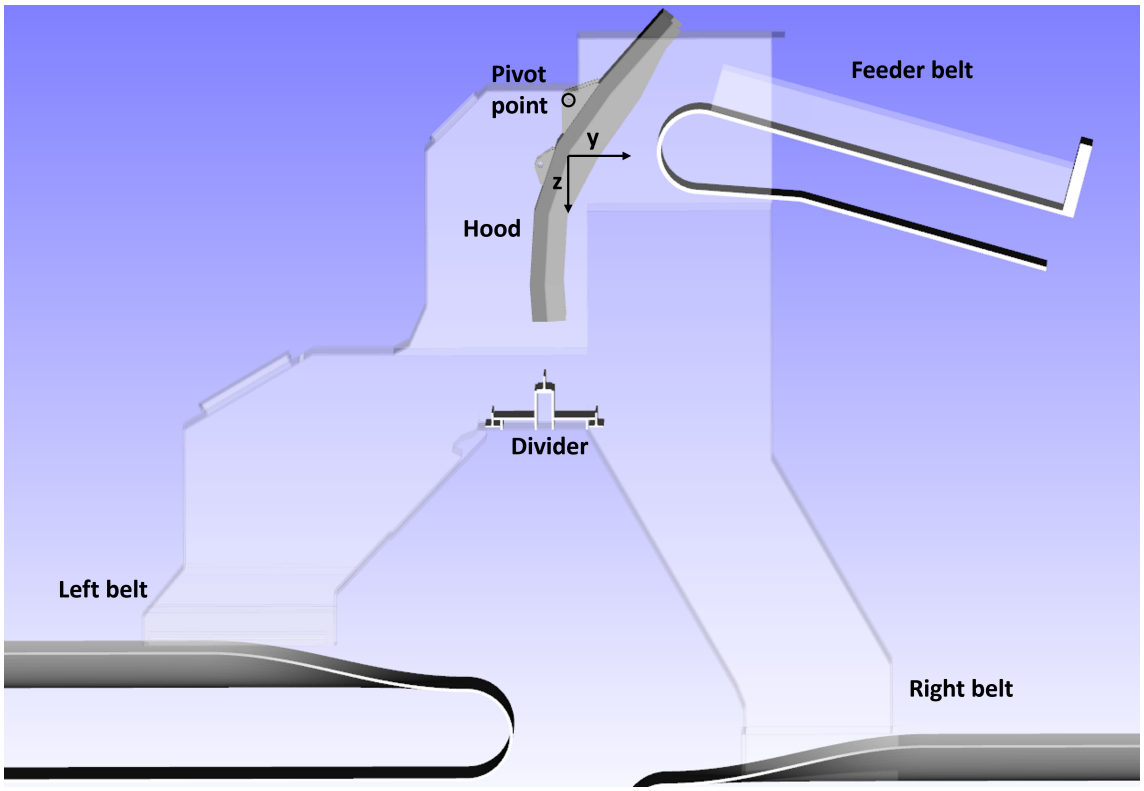
Angle ϕ	Y-deviation y	Z-deviation z	Belt speed v	Response mean	Response std
0	0	0	4	153.58	13.26

Table 4.5: Mean and standard deviation of model response over 10 simulations.

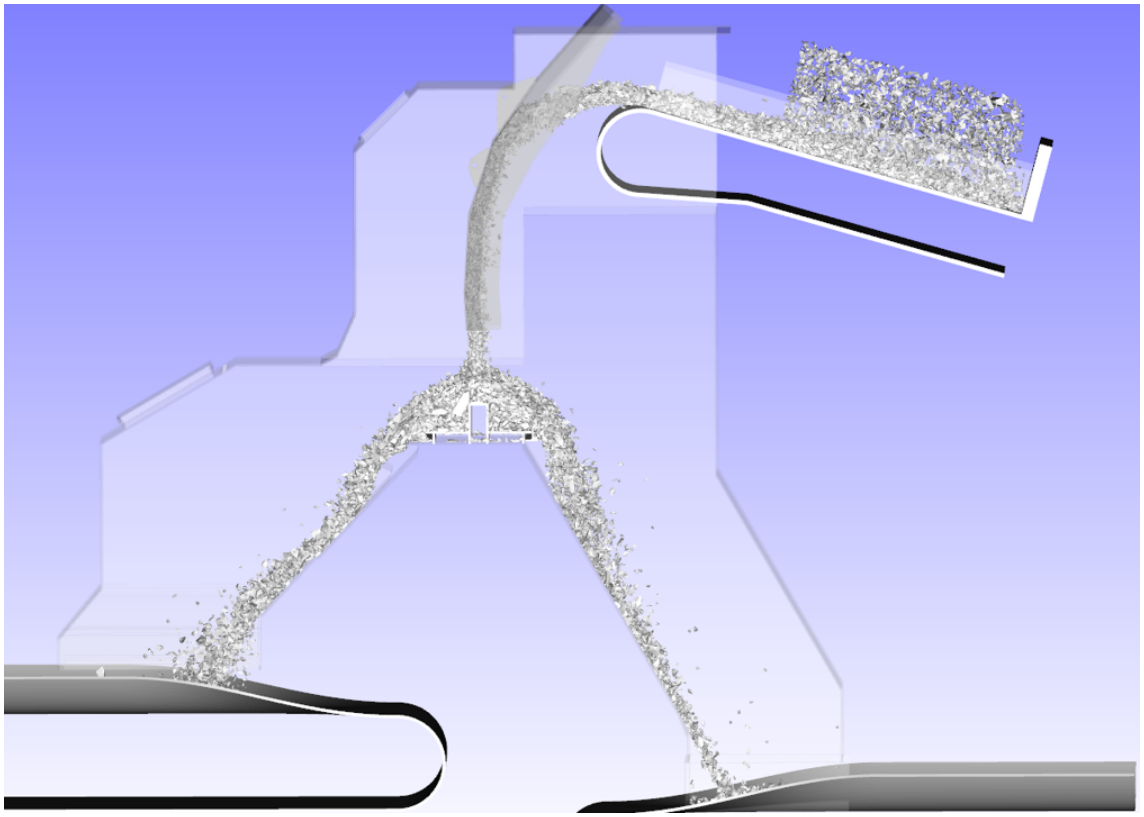
a proof of concept for showing that the algorithm works on larger problems and for real world optimization problems as well. The case is described in Figure 4.5. The material is generated on the feeding belt in the top right. The material is then funneled onto the divider and is split into the left and right chutes. The objective is to achieve a 50/50 split of material flow and particle distribution. The funnel can be moved in the y and z direction, as well as rotated ϕ degrees around the pivot point marked in Figure 4.5a. Furthermore the speed of the feeding belt v can be modified. These make up the variables of the optimization problem.

As the only model response and objective function, we use the absolute difference between the massflow on the left and right belts. This is large if the material is highly concentrated to one side, and zero if the material is perfectly split. The belts are moving at a fixed speed of 1 m/s, meaning that the massflow can be calculated by simply taking the mass in a cross section of the belt. Another possibility is to take the difference in particle distribution between the chutes as a model response. This is done by finding the distribution of characteristic lengths of the particles in each chute and using some distribution distance measure. However, our analysis show that the particle distribution largely follows the massflow distribution, meaning that such as response does not add any information to the optimization algorithm. On the contrary, it could be a source of noise and we therefore choose to not use it. Similar to the calibration device, there is noise in the model response. We evaluate this by running the model with fixed parameters 10 times. The model uncertainty is shown in Table 4.5.

The algorithm is applied to the case using the DEM parameters found in Table 4.6. The materials used for the belts is rubber, and for all other objects including the funnel and divider, it is steel. Since there is only one model response, early termination is not applicable in this case. The optimization was run on the C3SE Vera cluster with one worker.



(a) The case setup.



(b) An example of optimal material flow.

Figure 4.5: The splitting flow case.

Property	Symbol	Unit	Value
Contact model	-	-	HMD (Hertz-Mindlin-Deresiewicz)
Particle geometry	-	-	Dilated polyhedron
Particle density	ρ	kg/m^3	2750
Young's Modulus	E	GPa	4
Poisson's ratio	ν	-	0.28
Restitution (particle-particle)	e_{pp}	-	0.2
Restitution (particle-steel)	e_{ps}	-	0.15
Restitution (particle-rubber)	e_{ps}	-	0.15
Friction (particle-particle)	μ_{pp}	-	0.65
Friction (particle-steel)	μ_{ps}	-	0.45
Friction (particle-rubber)	μ_{ps}	-	0.7
Particle distribution	-	-	Normal with mean 50mm; std 40 mm
Particle distribution cutoff	-	-	Upper: 100mm; Lower: 20 mm
Time-step	Δt	s	8.0E-06

Table 4.6: DEM parameters used in simulation case 2.

Parameter	Symbol	Value
Variables		
Angle	ϕ	[-6,6] [°]
Y deviation	y	[-4,4] [dm]
Z deviation	z	[-4,4] [dm]
Belt speed	v	[3,4] [m/s]
Dimension	d	4
Max evaluations	$N_{\max}$	40
Initial sampling points	n_0	9
Candidate points in each iteration	t	200
Step-size max	$\sigma_{\max}$	0.2
Step-size min	$\sigma_{\min}$	$\frac{\sigma_{\max}}{2^6} = \frac{1}{320}$
Step-size increase criteria	C_{success}	3
Step-size decrease criteria	C_{fail}	4
Merit function weight pattern	w	[0.3, 0.5, 0.8, 0.95]

Table 4.7: Optimization parameters used for simulation case 2.

5

Results

5.1 Case 1: Calibration device

5.1.1 Convergence

We first run the optimization with $n = 1$, meaning the simulation only being ran once per function evaluation. Due to the uncertainty the spread of solutions is wide in this case. Therefore, to reduce the uncertainty and evaluate what happens if we add more computational effort to the calibration, we also run the optimization with a modified objective function with $n = 5$. Here each evaluation corresponds to running the DEM simulation five times and taking the mean of the responses.

Convergence plots are presented in Figure 5.1. We see that the optimization converges for both $n = 1$ and $n = 5$ in all three test cases. The curves are almost flat between evaluation 30 and 40, which tells us that a maximum number of evaluations at 40 is a good choice. It is possible that increasing the number of evaluations could improve the solution slightly, but we declare the computational effort not worth it. We also see that running with $n = 5$ gives a better objective value in all cases.

The solutions found are presented in Figure 5.2. The model responses and the differences between calibrated responses and reference responses are seen in Figure 5.3. For $n = 5$, we see that the solutions found are close to the optima and that the response values are within the standard deviations described in in Table 4.4. This means that the optimum found is within the uncertainty range of the model and that we cannot expect to get any closer due to noise in the problem. We therefore can state that the framework converges to the optimal solution for $n = 5$. For $n = 1$, we see that the variance of solutions is very high and that they in general are further away from the actual optimum and not always within the standard deviations of the model. However the calibrated responses are closer to the reference data on average for $n = 1$.

5.1.2 Early termination and performance

Figure 5.4a shows a graph of the objective value as a function of wall time elapsed. We see that optimization with early termination converges quicker and to a similar optimum as without. The number of function evaluations is fixed at 40 for both runs meaning that the performance increase is solely due to simulations being early terminated. The number of DEM simulations per evaluation is set at 5, i.e $n = 5$,

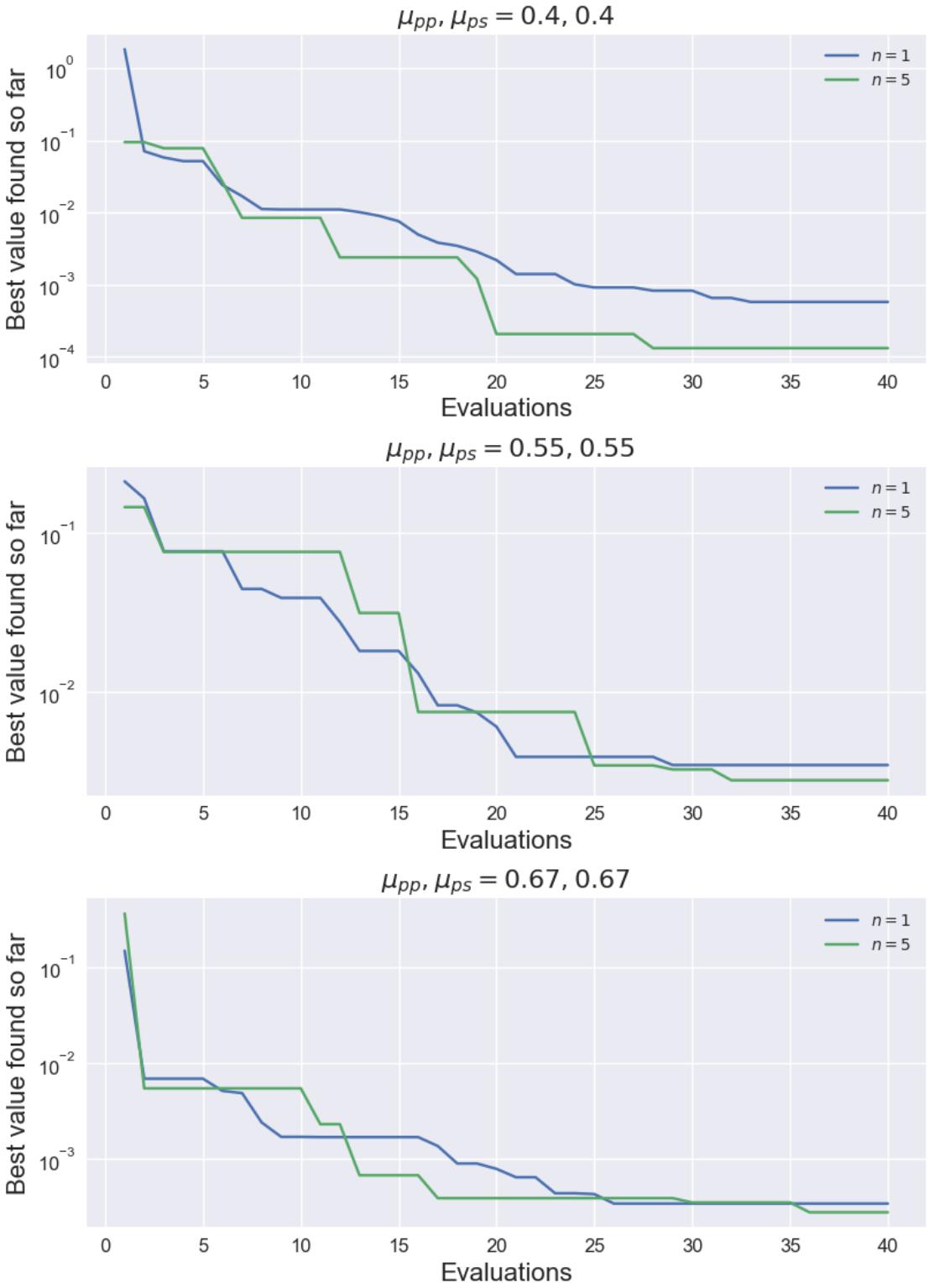


Figure 5.1: Convergence plots for case 1. For $n = 1$ we have taken the mean of five runs.

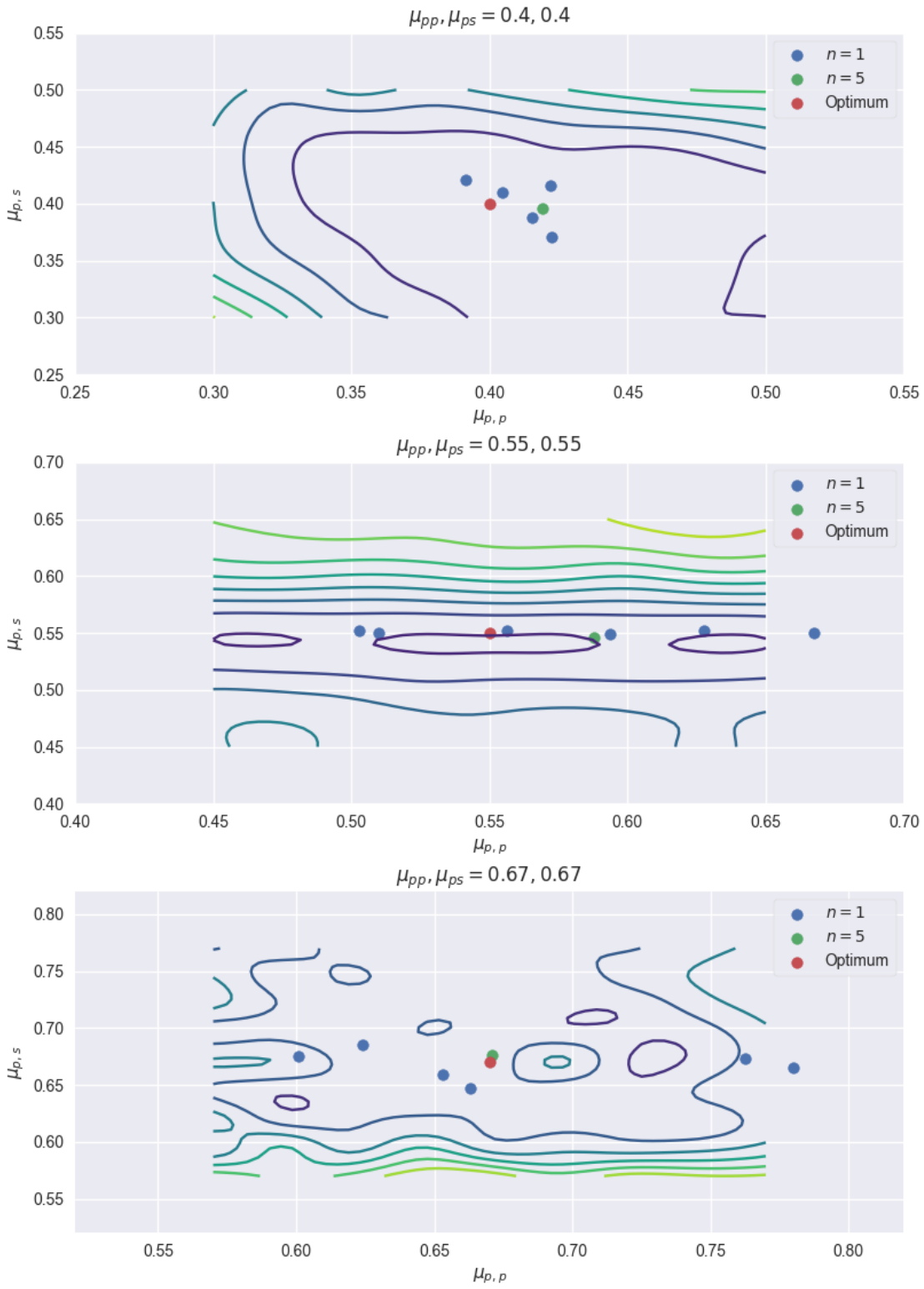


Figure 5.2: Solutions and contours of the objective function around optima.

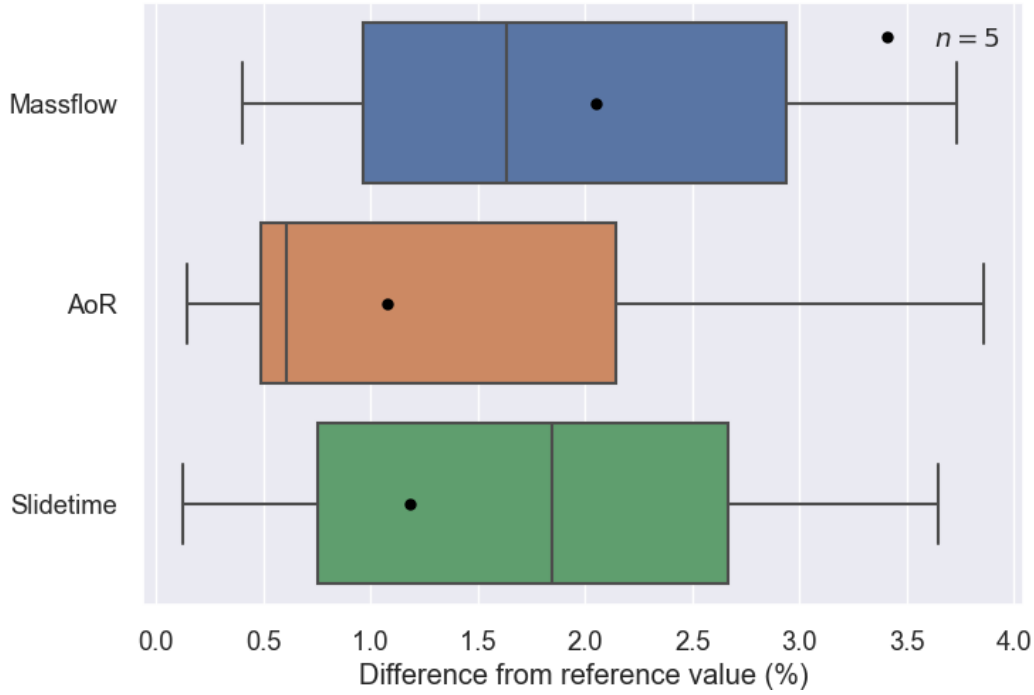


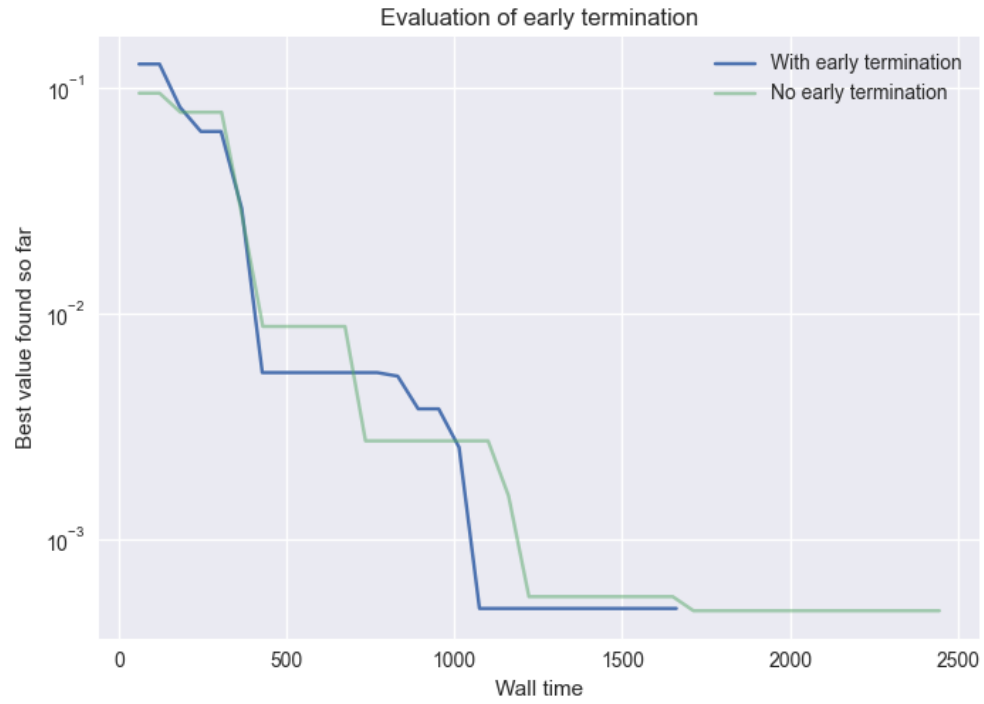
Figure 5.3: Boxplot of the difference between desired and calibrated responses for $n = 1$. Black dots represent $n = 5$.

and the termination criteria checks the results of the first simulation. This means that an early terminated evaluation requires only $\frac{1}{5}$ of the computational effort. The total time elapsed for the optimization is 41 and 28 hours, meaning that early termination yields a speedup of around 46% in this case. We refer to Section 6.1.1 for a more indepth discussion on the advantages and disadvantages of early termination.

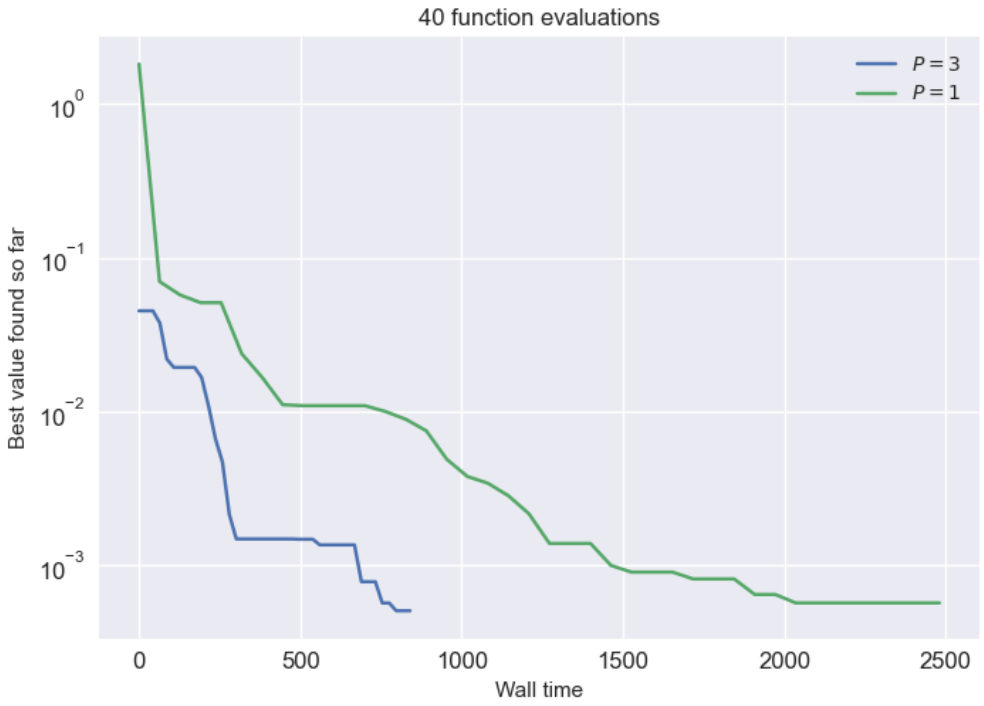
We also evaluate the impact of parallelism on the optimization. Figure 5.4b shows a graph of the objective value as a function of wall time elapsed for the serial and asynchronous parallel version of the algorithm, taken as mean of three runs with $\mu_{p,p} = 0.4$ and $\mu_{p,s} = 0.4$. The number of evaluations is fixed at 40 in both cases. We see that the best objective values are similar and that the parallel version is approximately three times faster. As stated, we expect a close to linear speedup with the number of workers P as the vast majority of computation time is spent on function evaluation and the additional overhead from more workers is negligible.

5.2 Case 2: Transfer chute material divider

The optimization framework is applied to the case once and the convergence plot is shown in Figure 5.5a. The algorithm converges to a solution that gave a 1.2 kg/s difference in the left and right massflow after about 30 evaluations. This is well within the uncertainty range of the model. The solution values are found in Table 5.1. To validate the solution we run the simulation with the parameters



(a) Convergence plot over time with and without early termination for optimization run with reference $\mu_{p,p} = 0.4, \mu_{p,s} = 0.4$.



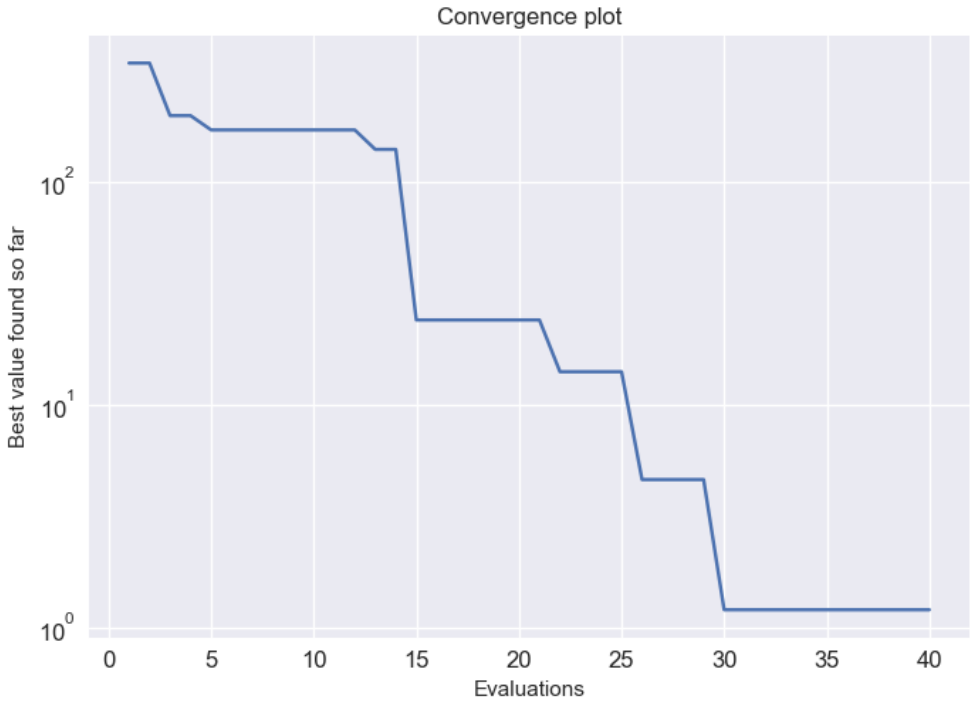
(b) Convergence plot over wall time comparing the serial and asynchronous parallel versions of the algorithm. Reference $\mu_{p,p} = 0.4, \mu_{p,s} = 0.4$.

Figure 5.4: Performance evaluation of the optimization algorithm when applied to the calibration device.

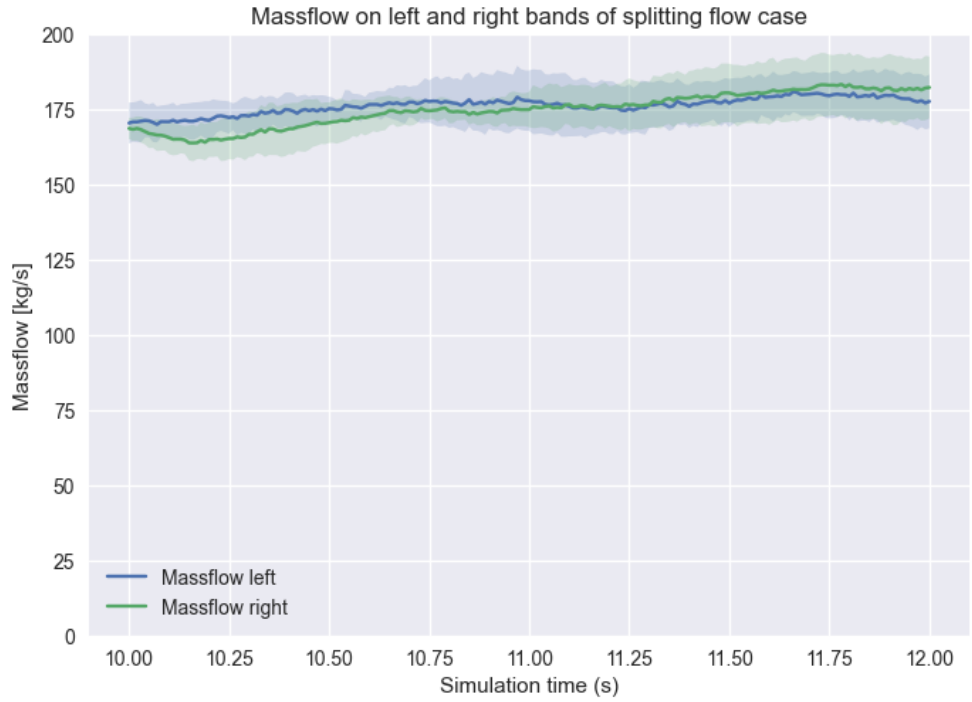
Angle ϕ	Y-deviation y	Z-deviation z	Belt speed v
-1.93	-0.18	0.84	3.69

Table 5.1: Best solution found by the algorithm when applied to case 2.

found five times. The average and standard deviations are shown in Figure 5.5b. We see that the mean of the massflows are varying, but mostly equal over time, and that the envelopes created by the standard deviations are close to each other. We conclude that the solution found by the optimization is valid. Due to the long simulation times and unavailablity of GPU:s at the cluster we choose to not evaluate the performance difference when running the algorithm asynchronous parallel. As is the previous case, a vast majority of computational time is spent evaluating the objective function, meaning that we expect a linear speedup with the number of processors in this case as well.



(a) Convergence plot of the optimization algorithm applied to case 2.



(b) Massflow of the found solution.

Figure 5.5: Optimization results on case 2

6

Discussion

6.1 Discussion of the results

We consider the results in Section 5.1.1 as good. The optimization converges for both $n = 1$ and $n = 5$, and finds solutions close to the optimum. We also see that the optimization finds solutions closer to the actual optima for $n = 5$ which is expected due to the reduced uncertainty resulting in a more smooth objective function. It seems as if the limiting factor for the results is the computational effort required, and not the optimization algorithm, which is positive.

Figure 5.3 show that the response errors are small for all runs, even though the solutions found are very different. We also see that $n = 1$ gives slightly better responses on average. This is possible in part due to noise in the model and in part due to the fact that different parameter sets can give the same model response. It is likely that in the runs with $n = 1$, a parameter set not necessarily close to the optimum gave very good model responses by chance, causing a local optimum that is hard for the algorithm to get out of. With $n = 5$, this is much less likely to happen since we average the responses from five simulations. We can also note that the mean and standard deviations are similar to the ones found in Table 4.4 which is expected. Finally, the results are achieved with a limit of 40 function evaluations, which is a manageable number in the context of costly black box optimization. The number of actual DEM simulations is dependent on n which is in turn dependent on how accurate we want the results to be and how much noise the model contains.

The optimization also converges on the second case and achieves a close to equal split of the material flow. This shows that the algorithm is viable for problems of higher dimension and that it has a capability to solve these within a reasonable time. It also tells us that the framework is useful for more applications than model calibration. Many projects at FCC involve creating a DEM model with the purpose of optimizing some industrial application at a partner company. The framework could be used to first design an optimization problem for calibrating the DEM model, and then for optimizing the industrial application.

We have successfully shown that our black box optimization framework is applicable to and can be used to calibrate or optimize a DEM simulation case. It also indicates the validity of our assumptions that DEM model calibration can be seen as a black box optimization problem. Furthermore it seems like the black box optimization algorithm developed by Regis and Shoemaker is a valid approach to solve DEM cal-

ibration problems. The main advantages and contributions of our framework is the utilization of asynchronous parallelism and early termination strategies. The largest limiting factor for DEM calibration today is the time and computational effort required. Parallelism reduces the wall time and early termination the computational effort required to perform calibration. This allows the framework to calibrate larger DEM-cases and to free up time for DEM users.

A direct comparison with other calibration frameworks is not possible since the calibration tests, parameters and parameter ranges differ from other proposed methods. To have a fair comparison, all algorithms must be tested under the same conditions. However we can still make some general remarks. Richter et al. propose a calibration framework in [30]. Here the coefficient of friction between particles μ and the coefficient of rolling friction μ_r are calibrated. An optimum is identified at $\mu = 0.8$ and $\mu_r = 0.2$. Their algorithm converges in a similar fashion to ours and finds solutions close to the optimum and within the estimated standard deviations of the model responses. The solutions we find are closer to the actual optimum, but this could be due to our model having less uncertainty. The method proposed by Rackl and Hanley is used to calibrate 6 different parameters with some success [28]. Their optimization needs between 30 and 75 DEM simulations to converge which is a good result considering that the problem is 6 dimensional. However, the allowed ranges of parameter values are narrow and the uncertainty in the solutions is high meaning that either the calibration problem contains too many local optima or that the optimization is not able to converge.

During the optimization process, we did not observe any significant difference, besides the wall-time, in the asynchronous and serial versions of the algorithm. Both versions showed very similar performance, despite the fact that one would expect the serial version to perform better given the same number of evaluations, as it should always have more or equal information (due to more evaluations being completed when the next point is selected) than the asynchronous version. The results are consistent with those reported in [13], where a similar asynchronous algorithm to the one we implemented, also developed by Regis and Shoemaker, performed better than the serial version. It seems as if for multi-modal optimization problems with many global or local optima, exploration is more important than maximizing information. As such, the asynchronous algorithm's ability to explore multiple regions of the search space simultaneously gives it an advantage over the serial version. Having said this, one can reasonable assume that there exists a number of processors P where adding more parallel evaluations yields diminishing returns.

An important takeaway for further work is that optimization of the calibration of a DEM model is only possible if the DEM model contains appropriate model responses for calibrating the chosen parameters. An optimization algorithm needs some quantitative way of measuring how good a suggested set of parameters is, and if the measures do not respond to changes in the parameters or if they contain too much noise and uncertainty, the algorithm will not be able to converge. Another important takeaway is that if a model contains much noise, it will likely require more computational effort to calibrate.

6.1.1 Early termination

Figure 5.4a shows that early termination has the potential to greatly reduce the wall time computation required. It also shows that if the function evaluations are suitable for early termination, there is clear benefit in utilizing it. However during the course of this project, it has also been found that it can be highly situational and if implemented poorly, lead to worse convergence. We will now discuss the prerequisites for early termination to work, and some advantages and disadvantages. The number one requirement for it is that there exists some form of information early in the evaluation that can be used to assess whether to continue or not. In a DEM calibration setting this can be either a model response that is available earlier than other, or the results from the first in a sequence of simulations. The information should preferably not contain much noise or uncertainty as this will make the decision whether to terminate or not much more difficult. Designing a simulation to contain this information is typically not very difficult. As an example, with Case 2 we could easily implement some kind of a massflow sensor earlier (further up the chutes) in the system. This could be used catch cases where all the mass flows into one of the chutes and terminate these cases quickly.

From the early information, it should be possible to design a suitable criteria that decides whether to terminate or not. The criteria should not add too many parameters to the optimization algorithm. It should also be robust in the sense that noise or uncertainty in the information does not influence the decision and make sure to never terminate runs that could lead to improvements in the objective value. Finally the optimization algorithm should be able to handle terminated evaluations.

Early termination can reduce the computation time required, but it comes at the cost of requiring more parameters in the algorithm and the risk of terminating promising runs. For example, the early termination criteria implemented in equation 3.4 has the disadvantage of having to choose values for c. These have a large effect on the performance of the algorithm. If c is too small, the algorithm will not be able to gather enough information and convergence will be slow. On the other hand, if c is too large early termination will happen very rarely. Proper parameter value might require an experienced user, or running the algorithm without early termination to study its behaviour on each specific case. Finally, the early termination functionality is case specific and requires some manual implementation for each new optimization case. We conclude that early termination has great potential to reduce the computational effort required in the optimization runs, but also requires some time spent implementing and validating the functionality of it. It is up to the user to decide whether the computational time saved is worth the extra effort.

6.2 Conclusion

In this thesis we have developed and implemented an optimization framework for use in DEM calibration problems. We have studied the DEM calibration problem and formulated it as a mathematical costly black box optimization problem. We solve the optimization problem with a surrogate based method, based on the Metric

Stochastic Response Surface method and using a Radial Basis function surrogate model. Furthermore with the help of Python software pySOT, we implemented early termination and asynchronous parallelism to speed up the solution procedure.

The framework is applied to two optimization problems involving DEM simulations. The first was a calibration device that can be used find the friction of a material. A sample of the material is used in the device to gather model responses to be used as reference data. The task of the optimization is to find friction values so that the DEM simulation responses are close to the reference. We show that the framework can be successfully used on the device with reference data from many different friction values. We also show that we get different levels of uncertainty in the solution depending on the computational power committed. The capabilities of asynchronous parallelism and early termination to decrease the wall time required for solving the optimization problem is demonstrated.

The second case was an industrial optimization problem involving a transfer chute. The objective is to find the position of a hood so that a material flow is equally split into two chutes. We designed the optimization problem to minimize the absolute difference in massflow in the two chutes and showed that the framework can find a solution successfully. Through this we demonstrated the capability of the framework to be applied on a wider range of problems.

This framework can now be used at FCC to make the calibration process more effective and simple. The optimization approach is more effective and can yield better results than a trial and error approach to calibration. Moreover, the framework is quick to implement on new DEM calibration cases and provides a standardized calibration procedure. Hopefully, this will save both time and effort and enable the researchers at FCC to spend more time applying DEM simulations to solve real world problems.

6.2.1 Research questions

In Section 1.2 a set of questions were formulated at the start of the project. We now try to answer them based on the results of the thesis.

- *How can a black box optimization framework be used to efficiently calibrate DEM models?*

  We formulated the calibration of DEM models as an optimization problem and reasoned about why we can view it as a black box optimization problem. A suitable objective function for the problem was proposed and a solution method developed. We highlighted the importance of recognizing that the solution of the optimization problem is dependent on model parameters, simply referred to as p in the objective function, that the optimization algorithm has no control over.

- *What algorithm is suitable for solving black box optimization problems in the context of DEM calibration?*

  We first performed a review of some state of the art approaches to calibrating

DEM models, with a special focus on approaches using some form of optimization. We then contributed with a new approach using surrogate optimization based on the MSRS algorithm by Regis and Shoemaker, with parallelism and early termination to decrease the wall time required for calibration. Our proposed approach performs well on the two applications and we discuss some of its advantages. As usual, there is no one-size-fits-all approach and each algorithm comes with trade-offs regarding accuracy, speed and ease of implementation.

- *How can early termination and parallelism be utilized to improve the speed of calibration?*

  We implemented early termination and showed that it decreases the wall time required when applicable. A disadvantage is that it requires knowledge of the model to set appropriate termination criteria, meaning that it is not always applicable. We also implemented asynchronous parallel function evaluations and showed a linear speedup with the number of processors. Since we did not observe any effects in the speed of convergence we concluded that early termination and asynchronous parallelism is a worthwhile way to speed of the calibration process.

- *Do the iterates produced by the algorithm converge to a solution with which we can consider the models calibrated?*

  In the two optimization problems we applied the framework to, the optimization converges to a solution in all cases. For the calibration device, the solution found is not always the best calibration parameters, but the optimization still converges in regards to the objective function becoming small. We say that the model has an uncertainty range. Within the range we can't tell if an objective value improvement is due to the parameters being better calibrated, or just due to variance in the model. In all cases the solutions found are within the uncertainty range of the models, and we can therefore say that they calibrate the model.

- *Does the parallelism and early termination improve efficiency and reduce computation time when compared to the serial optimization algorithm?*

  The framework requires relatively few evaluations to converge, meaning that the surrogate model usually provides a good approximation of the objective function. If early termination is used it saves computation time and resources by terminating unnecessary evaluations. Parallelism does not reduce the computational resources used, but decreases the wall time required to find a solution. Thus we can say that both parallelism and early termination improve the efficiency when compared to a purely serial algorithm.

6.2.2 Future work

Possible future work could involve assessing how to decrease the impact of model noise and uncertainty on the optimization model in other ways than simply averaging model responses. Some possible approaches could be weighing the responses

based on the uncertainty or testing other surrogates based on probabilistic models such as Gaussian processes. It should also involve applying the framework to more optimization cases. This will expose more advantages and disadvantages of the framework and see how it performs on a larger set of problems.

Another interesting topic would be to investigate how a large number of parallel evaluations affects the convergence speed of the algorithm. In the thesis we did not observe any significant difference between $P = 1$ or $P = 3$ parallel evaluations, but one can image that there exists some value of P when adding more processors gives diminishing returns. The simplest way to do this would be to just run the optimization on a case with varying P. The challenge would lie in constructing a case where the convergence variance is small enough to be able to find statistically significant results.

Finally, while this framework has been developed with the DEM in mind, it could also be used in calibration of other computationally expensive simulation methods. At FCC, there is research and simulation conducted within both the Finite Element Method and Computational Fluid Dynamics. Future work could include applying the framework to these methods as well.

Bibliography

[1] Audet, C. ; Le Digabel, S. ; Rochon Montplaisir, V. ; Tribes, C.: Algorithm 1027: NOMAD version 4: Nonlinear optimization with the MADS algorithm. In: *ACM Transactions on Mathematical Software* 48 (2022), Nr. 3, 35:1–35:22. http://dx.doi.org/10.1145/3544489. – DOI 10.1145/3544489

[2] In: Audet, Charles: *A Survey on Direct Search Methods for Blackbox Optimization and Their Applications*. New York, NY : Springer New York, 2014, S. 31–56

[3] Audet, Charles ; Dennis, J. E.: Mesh Adaptive Direct Search Algorithms for Constrained Optimization. In: *SIAM Journal on Optimization* 17 (2006), Nr. 1, 188-217. http://dx.doi.org/10.1137/040603371. – DOI 10.1137/040603371

[4] Beakawi Al-Hashemi, Hamzah M. ; Baghabra Al-Amoudi, Omar S.: A review on the angle of repose of granular materials. In: *Powder Technology* 330 (2018), 397-417. http://dx.doi.org/https://doi.org/10.1016/j.powtec.2018.02.003. – DOI https://doi.org/10.1016/j.powtec.2018.02.003. – ISSN 0032–5910

[5] Benvenuti, L. ; Kloss, C. ; Pirker, S.: Identification of DEM simulation parameters by Artificial Neural Networks and bulk experiments. In: *Powder Technology* 291 (2016), 456-465. http://dx.doi.org/https://doi.org/10.1016/j.powtec.2016.01.003. – DOI https://doi.org/10.1016/j.powtec.2016.01.003. – ISSN 0032–5910

[6] Bilock, Adam: *A GPU Polyhedral Discrete Element Method*. 2020

[7] Carl Edward Rasmussen, Christopher K. I. W.: *Gaussian Processes for Machine Learning*. The MIT Press, 2005. http://dx.doi.org/https://doi.org/10.7551/mitpress/3206.001.0001. http://dx.doi.org/https://doi.org/10.7551/mitpress/3206.001.0001. – ISBN 0–262–18253–X

[8] Charles Audet, Warren H.: *Derivative-Free and Blackbox Optimization*. Springer Cham, 2017. http://dx.doi.org/https://doi.org/10.1007/978-3-319-68913-5. http://dx.doi.org/https://doi.org/10.1007/978-3-319-68913-5. – ISBN 978–3–319–68913–5 Published: 02 December 2017

[9] Cheng, Hongyang ; Shuku, Takayuki ; Thoeni, Klaus ; Tempone, Pamela ; Luding, Stefan ; Magnanimo, Vanessa: An iterative Bayesian filtering

framework for fast and automated calibration of DEM models. In: *Computer Methods in Applied Mechanics and Engineering* 350 (2019), 268-294. http://dx.doi.org/https://doi.org/10.1016/j.cma.2019.01.027. – DOI https://doi.org/10.1016/j.cma.2019.01.027. – ISSN 0045–7825

[10] Coetzee, C.J.: Review: Calibration of the discrete element method. In: *Powder Technology* 310 (2017), 104-142. http://dx.doi.org/https://doi.org/10.1016/j.powtec.2017.01.015. – DOI https://doi.org/10.1016/j.powtec.2017.01.015. – ISSN 0032–5910

[11] Cundall, Peter A. ; Strack, Otto D. L.: A discrete numerical model for granular assemblies. In: *Geotechnique* 29 (1979), S. 47–65

[12] Do, Huy Q. ; Aragón, Alejandro M. ; Schott, Dingena L.: A calibration framework for discrete element model parameters using genetic algorithms. In: *Advanced Powder Technology* 29 (2018), Nr. 6, 1393-1403. http://dx.doi.org/https://doi.org/10.1016/j.apt.2018.03.001. – DOI https://doi.org/10.1016/j.apt.2018.03.001. – ISSN 0921–8831

[13] Eriksson, David ; Bindel, David ; Shoemaker, Christine A.: pySOT and POAP: An event-driven asynchronous framework for surrogate optimization. In: *arXiv preprint arXiv:1908.00420* (2019)

[14] Gutmann, HM: A Radial Basis Function Method for Global Optimization. In: *Journal of Global Optimization* 19 (2001), S. 201–227. http://dx.doi.org/https://doi.org/10.1023/A:1011255519438. – DOI https://doi.org/10.1023/A:1011255519438

[15] Hanley, Kevin J. ; O'Sullivan, Catherine ; Oliveira, Jorge C. ; Cronin, Kevin ; Byrne, Edmond P.: Application of Taguchi methods to DEM calibration of bonded agglomerates. In: *Powder Technology* 210 (2011), Nr. 3, 230-240. http://dx.doi.org/https://doi.org/10.1016/j.powtec.2011.03.023. – DOI https://doi.org/10.1016/j.powtec.2011.03.023. – ISSN 0032–5910

[16] Hans-Georg Matuttis, Jian C.: *Understanding the discrete element method*. Wiley, 2014

[17] Holmström, Quttineh NH. E. K.: An adaptive radial basis algorithm (ARBF) for expensive black-box mixed-integer constrained global optimization. In: *Optim Eng 9, 311–339* (2018). http://dx.doi.org/https://doi.org/10.1007/s11081-008-9037-3. – DOI https://doi.org/10.1007/s11081–008–9037–3

[18] Hough, Patricia D. ; Kolda, Tamara G. ; Torczon, Virginia J.: Asynchronous parallel pattern search for nonlinear optimization. In: *SIAM Journal on Scientific Computing* 23 (2001), Nr. 1, S. 134–156

[19] Hussain, Mohammed F. ; Barton, Russel R. ; Joshi, Sanjay B.: Metamodeling: Radial basis functions, versus polynomials. In: *European Journal of Operational Research* 138 (2002), Nr. 1, 142-154. http:

//dx.doi.org/https://doi.org/10.1016/S0377-2217(01)00076-5. – DOI https://doi.org/10.1016/S0377–2217(01)00076–5. – ISSN 0377–2217

[20] Ji, Shunying ; Sun, Shanshan ; Yan, Ying: Discrete Element Modeling of Rock Materials with Dilated Polyhedral Elements. In: *Procedia Engineering* 102 (2015), 1793-1802. http://dx.doi.org/https://doi.org/10.1016/j.proeng.2015.01.316. – DOI https://doi.org/10.1016/j.proeng.2015.01.316. – ISSN 1877–7058. – New Paradigm of Particle Science and Technology Proceedings of The 7th World Congress on Particle Technology

[21] Johnson, M.E. ; Moore, L.M. ; Ylvisaker, D.: Minimax and maximin distance designs. In: *Journal of Statistical Planning and Inference* 26 (1990), Nr. 2, 131-148. https://www.sciencedirect.com/science/article/pii/037837589090122B. – ISSN 0378–3758

[22] Li, Cheng-Qing ; Xu, Wen-Jie ; Meng, Qing-Shan: Multi-sphere approximation of real particles for DEM simulation based on a modified greedy heuristic algorithm. In: *Powder Technology* 286 (2015), 478-487. http://dx.doi.org/https://doi.org/10.1016/j.powtec.2015.08.026. – DOI https://doi.org/10.1016/j.powtec.2015.08.026. – ISSN 0032–5910

[23] Li, Qiang ; Feng, Mingxia ; Zou, Zongshu: Validation and Calibration Approach for Discrete Element Simulation of Burden Charging in Pre-reduction Shaft Furnace of COREX Process. In: *ISIJ International* 53 (2013), Nr. 8, S. 1365–1371. http://dx.doi.org/10.2355/isijinternational.53.1365. – DOI 10.2355/isijinternational.53.1365

[24] McKay, M. D. ; Beckman, R. J. ; Conover, W. J.: A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. In: *Technometrics* 21 (1979), Nr. 2, 239–245. http://www.jstor.org/stable/1268522. – ISSN 00401706

[25] Mindlin, R. D. ; Deresiewicz, H.: Elastic Spheres in Contact Under Varying Oblique Forces. In: *ASME. J. Appl. Mech. September 1953; 20(3): 327–344.* (1953). http://dx.doi.org/https://doi.org/10.1115/1.4010702. – DOI https://doi.org/10.1115/1.4010702

[26] Powell, MJD: *Radial Basis Functions in 1990 in Advances in Numerical Analysis Volume II-Wavelets, Subdivision Algorithms and Radial Basis Functions.* 1991

[27] Quist, Bilock A. Jareteg K. Persson A. J.: Undersökning av separationseffekter vid kompaktering av obundna material / Fraunhofer-Chalmers Centre for Industrial Mathemathics. Version: 2021. https://www.sbuf.se/Projektsida?project=005ad6c4-5b0a-4a78-a991-aadabad36a79. Göteborg, 2021 (SBUF project: 13820). – Forschungsbericht

[28] Rackl, Michael ; Hanley, Kevin J.: A methodical calibration procedure for discrete element models. In: *Powder Technology* 307 (2017), 73-83. http://dx.doi.org/https://doi.org/10.1016/j.powtec.2016.11.048. – DOI https://doi.org/10.1016/j.powtec.2016.11.048. – ISSN 0032–5910

[29] Regis, Rommel G. ; Shoemaker, Christine A.: A Stochastic Radial Basis Function Method for the Global Optimization of Expensive Functions. In: *INFORMS Journal on Computing* 19 (2007), nov, Nr. 4, S. 497–509. http://dx.doi.org/https://doi.org/10.1287/ijoc.1060.0182. – DOI https://doi.org/10.1287/ijoc.1060.0182

[30] Richter, Christian ; Rössler, Thomas ; Kunze, Günter ; Katterfeld, Andre ; Will, Frank: Development of a standard calibration procedure for the DEM parameters of cohesionless bulk materials – Part II: Efficient optimization-based calibration. In: *Powder Technology* 360 (2020), 967-976. http://dx.doi.org/https://doi.org/10.1016/j.powtec.2019.10.052. – DOI https://doi.org/10.1016/j.powtec.2019.10.052. – ISSN 0032–5910

[31] Rommel G. Regis, Christine A. S.: Parallel Stochastic Global Optimization Using Radial Basis Functions. In: *INFORMS Journal on Computing* 21 (2009), aug, Nr. 3, S. 411–426. http://dx.doi.org/https://doi.org/10.1287/ijoc.1090.0325. – DOI https://doi.org/10.1287/ijoc.1090.0325

[32] Torczon, Virginia: On the Convergence of Pattern Search Algorithms. In: *SIAM J. Optim.* 7 (1997), S. 1–25

[33] Vu, Ky ; D'Ambrosio, Claudia ; Hamadi, Youssef ; Liberti, Leo: Surrogate-based methods for black-box optimization. In: *International Transactions in Operational Research* 24 (2016), 04. http://dx.doi.org/10.1111/itor.12292. – DOI 10.1111/itor.12292

[34] Westbrink, Fabian ; Elbel, Alexander ; Schwung, Andreas ; Ding, Steven X.: Optimization of DEM parameters using multi-objective reinforcement learning. In: *Powder Technology* 379 (2021), 602-616. http://dx.doi.org/https://doi.org/10.1016/j.powtec.2020.10.067. – DOI https://doi.org/10.1016/j.powtec.2020.10.067. – ISSN 0032–5910

[35] Yan, Wilkinson S.K. Stitt E.H. et a. Z.: Discrete element modelling (DEM) input parameters: understanding their impact on model predictions using statistical analysis. In: *Comp. Part. Mech. 2, 283–299 (2015)* (2015). http://dx.doi.org/https://doi.org/10.1007/s40571-015-0056-5. – DOI https://doi.org/10.1007/s40571–015–0056–5

[36] Ye, Kenny Q. ; Li, William ; Sudjianto, Agus: Algorithmic construction of optimal symmetric Latin hypercube designs. In: *Journal of Statistical Planning and Inference* 90 (2000), Nr. 1, 145-159. http://dx.doi.org/https://doi.org/10.1016/S0378-3758(00)00105-1. – DOI https://doi.org/10.1016/S0378–3758(00)00105–1. – ISSN 0378–3758

[37] Zhu, H.P. ; Zhou, Z.Y. ; Yang, R.Y. ; Yu, A.B.: Discrete particle simulation of particulate systems: Theoretical developments. In: *Chemical Engineering Science* 62 (2007), Nr. 13, 3378-3396. http://dx.doi.org/https://doi.org/10.1016/j.ces.2006.12.089. – DOI https://doi.org/10.1016/j.ces.2006.12.089. – ISSN 0009–2509. – Frontier of Chemical Engineering - Multi-scale Bridge between Reductionism and Holism

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY