Implementation of the dual permeability model and application of population-based metaheuristics in inverse modeling

Johanna R. Blöcher

April 18, 2017

Under supervision of doc. Ing. M. Kuráž

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Environmental Sciences

DIPLOMA THESIS ASSIGNMENT

Johanna Ruth Blöcher

Environmental Modelling

Thesis title

Implementation of the dual permeability model and application of population-based metaheuristics in inverse modeling

---

Objectives of thesis

Implementation of dual permeability model into in-house code DRUtES.

Design, implementation and evaluation of novel and state of the art population-based metaheuristic algorithms for theoretical and real world problems.

Methodology

literature retrieval

research on implemented algorithms in R

code structure of DRUtES

DEBUGGING

benchmark function evaluations

Richards equation model based inverse problems

The proposed extent of the thesis

standard

Keywords

standard

Recommended information sources

Kogelbauer, I, Batkova, K, Dolezal, F, Matula, S, Loiskandl, W, Preferential percolation quantified by large water content sensors with artifactual macroporous envelopes. HYDROLOGICAL PROCESSES, SEP 15, 2015, DI 10.1002/hyp.10491.

Expected date of thesis defence

2016/17 SS – FES

The Diploma Thesis Supervisor

doc. Ing. Michal Kuráž, Ph.D.

Supervising department

Department of Water Resources and Environmental Modeling

Advisor of thesis

doc. Ing. Michal Kuráž, Ph.D.

Electronic approval: 10. 4. 2017

doc. Ing. Martin Hanel, Ph.D.

Head of department

Electronic approval: 10. 4. 2017

prof. RNDr. Vladimír Bejček, CSc.

Dean

Prague on 18. 04. 2017

Author's declaration

I hereby declare that this submitted thesis *Implementation of the dual permeability model and application of population-based metaheuristics in inverse modeling* is my own work and only sources listed in the Bibliography were used.

Prague, 18^{th} April 2017

Johanna R. Blöcher

Acknowledgments

I would like to sincerely thank Michal Kuráž, who has been an incredible supervisor. Thank you for helping me create a somewhat overambitious thesis project and for accommodating most of my crazy ideas.

I would also like to thank Petr Máca for introducing me to metaheuristics and in particular Particle Swarm Optimization. His passion for optimization algorithms is truly inspiring. I would also like to thank the Department of Water Resources and Environmental Modeling, a great bunch of people!

I would also like to thank František Doležal for providing case study data for this thesis.

I would also like to thank my family for all the support. I would like to especially thank my husband and personal hero, Mark Thomson, for getting me through the tough times. I would also like to thank my sister, Solveig Blöcher, for her incredible confidence and her enthusiasm.

Abstract

Soil often exhibits a variety of heterogeneities such as fractures, fissures, cracks, and macropores, and can show dynamic instabilities of the wetting front during infiltration, which can lead to preferential flow. Preferential flow can accelerate the movement of contaminants in the unsaturated zone and is therefore of concern to environmental science. The dual permeability model after Gerke and Genuchten (1993a, 1993b) can be used to model preferential flow in soil. It was constructed around the assumption that the soil medium can be separated into two distinct pore systems. In this approach the matrix and fracture systems are each treated as homogeneous media with separate hydraulic properties that can both be described with a Richard's equation. In this work, the dual permeability model was implemented in the free software Dual Richards' Unsaturated Equation Solver ($DRUtES$). Different scenarios were designed to test under which conditions preferential flow occurs.

In order to perform parameter estimation as part of inverse analysis, global metaheuristic optimization algorithms were reviewed and implemented. Adaptations of Particle Swarm Optimization (PSO) and Teaching-Learning-Based Optimization (TLBO) were used, which are population-based metaheuristics with different learning strategies. These are high-level stochastic-based search algorithms that don not require gradient information or a convex search space. Introduced variants include a simple bad neighborhood approach and shuffling complexes inspired by Duan et al. (1993). Despite increasing computing power and parallel processing, an overly fine mesh is too computationally intensive to be used for parameter identification. This creates the need to find a mesh that optimizes both accuracy and simulation time. A bi-objective PSO algorithm was designed and implemented. This algorithm is based on a dynamic neighborhood approach and can generate a Pareto front of optimal meshes to account for both objectives. The global optimizers were successfully tested on commonly used benchmark functions. The created inverse set-up involved the linking of the software to the optimization algorithms and was tested on both virtual inverse problems and real data.

Keywords: Dual permeability model, Preferential Flow, Inverse modeling, Population-Based Metaheuristics, Particle Swarm Optimization, Teaching-Learning-Based Optimization, Shuffled Complexes, Bad Neighborhood Approach.

Contents

List of Figures

List of Tables

1 Introduction

Soil is not only the growth medium to plants, the very basis for our terrestrial food chain, but also the connecting layer of the atmosphere, biosphere, lithosphere, hydrosphere and pedosphere (Nieder 2011). Soil is of immense importance for transport and storage processes between these spheres.

Heterogeneities such as fractures, fissures, cracks, and macropores in the soil can lead to preferential flow (Gerke and Genuchten 1993a). Preferential flow can accelerate the movement of agricultural contaminants and other pollutants through the unsaturated zone and is therefore of concern to hydrologists, geophysicists and environmental scientists (Šimůnek et al. 2003). It is therefore important to be able to model preferential flow.

Furthermore, an exact evaluation of experiments under transient flow conditions is only possible through modeling the flow process in combination with parameter estimation (Iden and Durner 2007). Estimating soil hydraulic properties from observations or known output is an inverse analysis. Computational tools have become very important in inverse modeling. To perform parameter estimation, optimization algorithms can be used to perform the optimization task. Metaheuristic optimization offers interesting possibilities. Metaheuristics are high-level stochastic-based search algorithms that don't require gradient information or a convex search space. Although they have been utilized in estimating soil hydraulic properties and calibrating hydrological models (Iden and Durner 2007; Jakubcova et al. 2015; Piotrowski et al. 2017), they are not widely used in soil physics.

1.1 Motivation and goals

The main motivation of this thesis is the implementation of a soil physics model, which is able to simulate preferential flow, in the open-source objective library Dual Richards' Unsaturated Equation Solver and to create a set-up for inverse modeling with global optimization algorithms. For this task, the dual permeability model after Gerke and Genuchten (1993a) was chosen.

Further aims of this thesis included a review on population-based metaheuristics and the selection of interesting algorithms and the selection and design of promising modifications. These modifications include a novel bad neighborhood approach and shuffling complexes inspired by Duan et al. (1993). Additionally, this work aims to test these modifications on benchmark functions in order to select the most promising for an inverse modeling case study.

Lastly, the dual permeability model and the selected algorithm were used to perform parameter estimation on a case study. In this case study, the Time Domain Reflectometer (TDR) sensor readings showed a very steep increase during rapid rainfall events and

subsequent steep decrease. This was theorized to be an effect of artificial macroporous envelopes surrounding TDR sensors creating an anomalous region with distinct local soil hydraulic properties and preferential flow. One of the objectives of this thesis is to test how well the dual permeability model can describe the observed infiltration behavior. Parameter estimation is also conducted with the standard uni-modal van Genuchten model to analyze if the dual permeability shows a significantly better description than the standard uni-modal van Genuchten model. Despite increasing computing power and parallel processing, an overly fine mesh is not feasible for parameter identification. This creates the need to find a mesh that optimizes both accuracy and simulation time. A further objective of this thesis was thus to develop a bi-objective algorithm to generate a Pareto front of optimal meshes to account for both objectives.

1.2 Thesis structure

Chapter 2 begins with a derivation of the water flow equations in soil and the dual permeability model approach according to Gerke and Genuchten (1993a). Chapter 3 includes a description of the model implementation into *DRUtES* with implemented boundary types and coupling variants as well as test simulations. Chapter 4 includes a literature review on metaheuristics with a focus on population-based metaheuristics. Chapter 4 also describes the chosen classes of algorithms, namely Particle Swarm Optimization (*PSO*) and Teaching-Learning Based Optimization (*TLBO*), their underlying working principles and introduced modifications. Chapter 4 also includes a description of a bi-objective optimization algorithm, which is able to find a set of optimal solutions when two objective functions require optimization. Chapter 5 reviews benchmark functions and describes chosen benchmark functions from the CEC 2013 real parameter optimization session and created *DRUtES* benchmark functions. Chapter 5 also includes a benchmark function results section. This was decided to be more adequate than placing all results into a single chapter and aimed to improve the structure and readability. Chapter 6 contains the case study problem and proposed mesh optimization in addition to the inverse modeling problem. Chapter 7 contains the final conclusions with further research aims. The most important codes and created software have been made accessible on Github. Links to the respective repositories can be found in Appendix C.

2 Mathematical flow model

2.1 Introduction

This chapter derives basic constitutive functions for flow through porous media under variably saturated conditions. Variably saturated flow occurs in porous systems above aquifers, in soils and more specifically in the vadose zone. The accurate characterization of hydraulic properties in unsaturated soils is critical to understand and solve problems in agriculture, hydrology, environmental sciences and many other disciplines, and knowledge of them is required in many management tasks (Durner and Flühler 2005). This chapter concludes with a description of the dual permeability model approach as postulated in Gerke and Genuchten (1993a) and Gerke and Genuchten (1993b).

2.2 Basic governing equations

A fundamental law describing flow in porous media was discovered and published in 1856 by Henry Darcy. His observations lead to the conclusion that the volume flux density $\vec{q}$ is proportional to the energy gradient, which is expressed as the gradient of the hydraulic head (Darcy 1856). His equation underlies the principle of homogenization. The relationship between the volume flux density and the energy gradient is

$$\vec{q} = -K\nabla H, \tag{2.1}$$

where $\vec{q}$ is the flux [L T^{-1}], K is saturated hydraulic conductivity [L T^{-1}] and H is the total hydraulic head [L], which is defined as

$$H = h + z, \tag{2.2}$$

where h is the pressure head [L] and z is the geodetic head $z = \begin{pmatrix} 0 \\ 0 \\ z \end{pmatrix}$.

If the coordinate system is positive upward $\nabla z = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$. Darcy's law can then be described as

$$\vec{q} = -K\nabla(h + z) = -K(\nabla h + \nabla z) = -K(\nabla h + 1). \tag{2.3}$$

Originally, this law was derived for a saturated state, where all pores are filled with water and the pressure head is positive. Edgar Buckingham (1867-1940) derived a similar law for the unsaturated state, leading to the Darcy-Buckingham law (Buckingham 1907), where the hydraulic conductivity is dependent on the water content θ and thus $K(\theta)$ becomes the unsaturated hydraulic conductivity and equation (2.1) turns into

$$\vec{q} = -K(\theta)\nabla H, \tag{2.4}$$

where K is dependent on the gravity field, properties of the liquid and porous media and defined as

$$K = -\frac{kg}{\upsilon}, \tag{2.5}$$

where k is permeability [L^2], g is gravitational acceleration [L T^{-2}] (9.81 m s^{-2}) and υ is the kinematic viscosity [L^2 T^{-1}]. υ can be expressed as $\upsilon = \frac{\mu}{\rho}$, where μ is the dynamic viscosity [L^2 T^2 M^{-1}] and ρ is the liquid's density [M L^{-3}].

Darcy-Buckingham law is constrained by the flow regime. The flow regime has to be laminar, which can be determined using the Reynold's number (Re). A commonly used threshold is $Re = 10$. For $Re < 10$ we can assume laminar flow in porous media (Czachor 2011). Generally, one can assume laminar flow for finer soils. However, Re may exceed 10 in macropores, where when the equivalent grain size is $d_e >$ 2-3 mm. For the simulations in this thesis we assume laminar flow.

$$Re = \frac{qd_e}{\upsilon}, \tag{2.6}$$

where d_e is the equivalent grain size [L]. K can be considered a second order tensor, so that K is

$$K(x,y,z) = \begin{pmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{yx} & K_{yy} & K_{yz} \\ K_{zx} & K_{zy} & K_{zz} \end{pmatrix}. \tag{2.7}$$

When the principle directions are aligned with the selected global coordinate system it simplifies to

$$K(x,y,z) = \begin{pmatrix} K_{xx} & 0 & 0 \\ 0 & K_{yy} & 0 \\ 0 & 0 & K_{zz} \end{pmatrix}. \tag{2.8}$$

The volume flux can be expressed as

$$\vec{q} = -K(x,y,z,\theta)\nabla h + K_{zz}(\theta). \tag{2.9}$$

In a homogeneous medium K is independent of spatial coordinates and in an isotropic medium

we can simplify the hydraulic conductivity term to $K = K_{xx} = K_{yy} = K_{zz}$. The flow in porous media is described by mass conservation. We assume the flow of an incompressible liquid, i.e. no change of density, in a non-deformable medium, i.e. no swelling and shrinking of the soil.

The mass conservation law for incompressible liquids and non-deformable media in soil is often expressed by the following ordinary differential equation (ODE):

$$\nabla.\vec{q} = -\frac{\partial\theta}{\partial t}, \tag{2.10}$$

where $\nabla.$ denotes the vector operator divergence. Combining the mass conservation law with the Darcy-Buckingham law, we obtain the mixed-form Richard's equation for a homogeneous porous medium. The Richard's equation is a convection diffusion (reaction) equation. Often, the Richard's equation is denoted with an additional sink term S that can represent processes such as root water uptake. The derived second order partial differential equation (PDE) is

$$\nabla.(K(\theta)\nabla h) + \frac{\partial K(\theta)}{\partial z} - S = \frac{\partial\theta}{\partial t}. \tag{2.11}$$

Using the chain rule, for a homogeneous medium, $\frac{\partial K(\theta)}{\partial z}$ can be expressed as $\frac{\partial K(h)}{\partial z} = \frac{\mathrm{d}K(h)}{\mathrm{d}h}\frac{\partial h}{\partial z}$. Here, the primary solved variables are both the pressure head and the water content θ. The Richard's equation can also be formulated as an h-based form and θ-based form. The h-based form is implemented in Dual Richards' Unsaturated Equation Solver ($DRUtES$).

For the h-based form, a water retention capacity term C $[\mathrm{L}^{-1}]$ is introduced

$$C(h) = \frac{\mathrm{d}\theta}{\mathrm{d}h}, \tag{2.12}$$

which is valid for $\theta \in (\theta_r, \theta_s)$ and for $h \in (-\infty, \infty)$. $C(h) = 0, h \geq 0$. Since $\frac{\partial\theta}{\partial t} = \frac{\partial\theta(h)}{\partial t}$ we can use the chain rule to obtain $\frac{\partial\theta(h)}{\partial t} = \frac{\mathrm{d}\theta}{\mathrm{d}h}\frac{\partial h}{\partial t}$. The capacity term can subsequently be substituted for $\frac{\partial\theta}{\partial t}$

$$\frac{\partial\theta}{\partial t} = C(h)\frac{\partial h}{\partial t}. \tag{2.13}$$

Since θ is dependent on h, the dependency on θ in K can be substituted for h. The h-based Richards equation can thus be expressed as

$$\nabla.(K(h)\nabla h) + \frac{\partial K(h)}{\partial z} - S = C(h)\frac{\partial h}{\partial t}. \tag{2.14}$$

In terms of the convection-diffusion reaction equation, the $\nabla.(K(h)\nabla h)$ represents the

diffusion term, $\frac{\partial K(h)}{\partial z}$ represents the convective term, which can be non-zero, $C(h)\frac{\partial h}{\partial t}$ represents the elasticity, or, as the name already states, capacity term. The sink term S is a reaction term of the 0th order. However, the Richards equation may contain further reaction terms of higher order.

2.3 Dual permeability model

Porous media often exhibit a variety of heterogeneities such as fractures, fissures, cracks, and macropores, and can show dynamic instabilities of the wetting front during infiltration (Gerke and Genuchten 1993a). This can result in nonuniform flow with widely distinct velocities. Beven and Germann (2013) state that the primary drivers for interest in preferential flow and macropores in soils was the problem of explaining how pesticides and other sorbing pollutants were being transported to field drains, groundwaters and rivers. Preferential flow and non-equilibrium flow have strong implications in accelerating the movement of agricultural contaminants, radionuclides and non-aqueous liquids and other pollutants through the unsaturated zone to underlying groundwater and are therefore of concern to hydrologists, geophysicists and environmental scientists (Šimůnek et al. 2003). Common models describing preferential non-equilibrium flow are the dual permeability model and the dual porosity model. Both approaches divide the soil into a fracture and matrix domain. The dual permeability model was constructed around the assumption that the medium can be separated into two distinct pore systems. The dual porosity model assumes an immobile matrix domain, whereas the dual permeability model assumes a mobile matrix domain. There are different implementations of the dual permeability model, which mainly differ in the description of the macroporous or fracture domain (Gerke 2011). Germann (1985) and Germann and Beven (1985) suggested a kinematic wave approach in combination with a sorbance factor r, where

$$\frac{\partial q}{\partial t} + v\frac{\partial q}{\partial z} + vr\theta_f = 0 \tag{2.15}$$

and

$$q = b\theta_f^a, v = \frac{\partial q}{\partial \theta_f}, \tag{2.16}$$

where a is a kinematic exponent and b is a macropore conductance parameter [L T^{-1}].

In the Gerke and Genuchten (1993a) approach the matrix and fracture systems are each treated as homogeneous media with separate hydraulic properties that can both be described with a Richard's equation. The dual-permeability model is considered to be a superposition of these two pore systems. The two media are coupled with a first order transfer coefficient Γ_w.

Gerke and Genuchten (1993a) describe the dual permeability model as consisting of soil

aggregates or rock matrix blocks surrounded by inter-aggregate macropores or fractures which form a more or less continuous network. The matrix medium may contain macropores or fractures and the fracture medium may also include mesopores and micropores in the immediate vicinity of the macropores as well as some mineral or organic particles along the macropore walls.

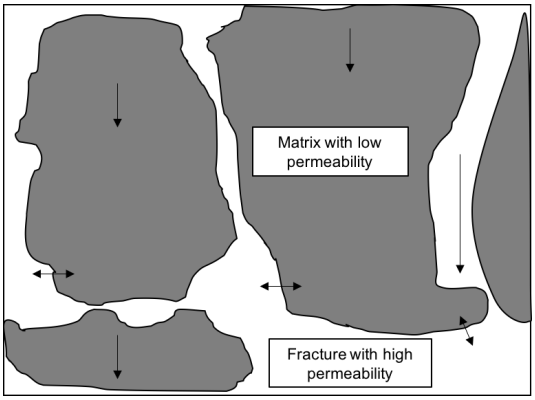


Figure 1: Schematic of a medium representing the dual permeability approach. The white area represents the highly permeable fracture medium, whereas the shaded area represents the matrix medium with low permeability.

The resulting coupled partial differential equation (PDE) is

$$\begin{aligned} C(h_m)\frac{\partial h_m}{\partial t} &= \nabla.(K(h_m)\nabla h_m) + \frac{\partial K(h_m)}{\partial z} + \frac{\Gamma_w}{1-w_f} - S_m \\ C(h_f)\frac{\partial h_f}{\partial t} &= \nabla.(K(h_f)\nabla h_f) + \frac{\partial K(h_f)}{\partial z} - \frac{\Gamma_w}{w_f} - S_f, \end{aligned} \tag{2.17}$$

where w_f is the fracture weight, $1 - w_f = w_m$ is the matrix weight and Γ_w is a first order transfer term and proportional to the difference in pressure head between the fracture and matrix pore system as follows:

$$\Gamma_w = a_w(h_f - h_m), \tag{2.18}$$

where a_w is a first-order transfer coefficient for water [L T$^-$1] defined as

$$\alpha_w = \alpha_w^* K_a \tag{2.19}$$

with

$$\alpha_w^* = \frac{\beta\gamma_w}{a^2}, \tag{2.20}$$

where β depends on the geometry of the aggregates (3 for slabs and 15 for spheres), a represents

the distance form the center of a fictitious matrix block to the fracture boundary [L], γ_w is an empirical coefficient found to be 0.4 and K_a is the hydraulic conductivity function of the boundary or, in other words, the exchange boundary conductivity [L^{-T}].

The soil properties in the dual permeability model relate to each other differently than in the dual porosity model. The fracture weight w_f is defined in Gerke and Genuchten (1993a) as

$$w_f = \frac{V_{t,f}}{V_t}, \tag{2.21}$$

where $V_{t,f}$ is the total fracture volume and V_t is the total volume. The matrix weight w_m can be computed with $w_m = 1 - w_f$. The soil properties of the fracture and matrix media and the superposed total medium are related in the following way

$$V_t = V_{t,f} + V_{t,m} \tag{2.22}$$

and

$$V_p = V_{p,f} + V_{p,m}, \tag{2.23}$$

where V is the volume [-], subscript t denotes the total volume and subscript p denotes the pore volume. The subscripts f and m denote the respective volume of the fracture and matrix domain. The porosities $\epsilon = \frac{V_p}{V_t}$ [-] relate to each other as follows

$$\epsilon = w_f \cdot \epsilon_f + w_m \cdot \epsilon_m. \tag{2.24}$$

Similarly, the total volumetric water content θ [-] is

$$\theta = w_f \cdot \theta_f + w_m \cdot \theta_m. \tag{2.25}$$

According to Gerke and Genuchten (1993a) the total volumetric volume flux q [$L\ T^{-1}$] is

$$q = w_f \cdot q_f + w_m \cdot q_m, \tag{2.26}$$

an area-weighted sum where the water flux density in the matrix domain q_m is

$$q_m = \frac{Q_m}{A_m} = w_m \frac{Q_m}{A}, \tag{2.27}$$

where Q_m is unit volume flowing through the unit area of the matrix A_m and the water flux density in the fracture domain q_f is

$$q_f = \frac{Q_f}{A_f} = w_f \frac{Q_f}{A}, \tag{2.28}$$

where Q_f is unit volume flowing through the unit area of the fracture A_f and

$$q = \frac{Q_m}{A} + \frac{Q_f}{A} = \frac{Q_m + Q_f}{A_m + A_f}. \tag{2.29}$$

In the implementation in *DRUtES* to solve the system, we multiply by the weights. The total flux is then more accurately described as a sum of q_f and q_m

$$q = q_f + q_m, \tag{2.30}$$

where q_f and q_m are either area weighted fluxes of the total assigned flux with

$$q = w_f \cdot q + w_m \cdot q, \tag{2.31}$$

or more general when area-independent infiltration weights w_{inf} are used

$$q = w_{inf} \cdot q + (1 - w_{inf}) \cdot q. \tag{2.32}$$

2.4 Parameterization of hydraulic functions

In this thesis the parameterization of the van Genuchten model (Genuchten 1980) was used to parameterize the retention function and the van Genuchten-Mualem model (Mualem 1976) was used to parameterize the hydraulic conductivity function. The van Genuchten retention description is a reflection of a unimodal pore size distribution. The van Genuchten-Mualem hydraulic conductivity model was derived based on a pore bundle model. Mualem (1976) predicted unsaturated hydraulic conductivity curves by using the moisture content-capillary head curve and measured the value of the hydraulic conductivity at saturation.

The retention function is parameterized as

$$\theta(h) = \begin{cases} \frac{\theta_s - \theta_r}{(1+(-\alpha h)^n)^m} + \theta_r, & \forall\, h \in (-\infty, 0) \\ \theta_s, & \forall\, h \in \langle 0, +\infty) \end{cases}, \tag{2.33}$$

the retention water capacity functions is parameterized as

$$C(h) = \begin{cases} \frac{\alpha m n (-\alpha h)^{n-1} (\theta_s - \theta_r)}{(1+(-\alpha h)^n)^{1+m}}, & \forall\, h \in (-\infty, 0) \\ 0, & \forall\, h \in \langle 0, +\infty) \end{cases} \tag{2.34}$$

and the hydraulic conductivity function is parameterized as

$$K(h) = \begin{cases} K_s \frac{(1-(-\alpha h)^{nm}(1+(-\alpha h)^n)^{-m})^2}{(1+(-\alpha h)^n)^{\frac{m}{2}}}, & \forall\, h \in (-\infty, 0) \\ K_s, & \forall\, h \in \langle 0, +\infty) \end{cases}, \tag{2.35}$$

where α is the inverse of the air entry value or bubbling pressure [L^{-1}], m and n define pore-size distribution and the slope of the retention function and hydraulic conductivity function. θ_s is the saturated water content [-], θ_r is the residual water content [-] and K_s is the saturated hydraulic conductivity [L T^{-1}].

2.5 Initial and Boundary conditions

The initial condition describes the state of the system at the beginning of a computation. It is often assumed as a known distribution of the pressure function or water content function leading to

$$h(x, t_0) = h_0(x),\ \forall x \in \Omega \tag{2.36}$$

or

$$\theta(x, t_0) = \theta_0(x),\ \forall x \in \Omega, \tag{2.37}$$

where Ω is the computational domain bounded by $\Gamma = \delta\Omega$.

Neumann and Dirichlet type boundary conditions can be applied solving Richard's equation. A Dirichlet boundary condition assigns a known pressure head. In some software this type of boundary condition is termed constant head, but theoretically, we could also assume a time dependent Dirichlet condition. Dirichlet boundary conditions can be used to simulate ponding. In our case the Dirichlet condition can be expressed as

$$h(x, t) = h_\Gamma \equiv \theta(x, t) = \theta_\Gamma,\ \forall (x, t) \in \Gamma \times [0, T). \tag{2.38}$$

A Neumann boundary condition assigns a known flux to the boundary. Neumann boundary conditions can be used to simulate rainfall intensity and evaporation. It can also be used to assign no-flow boundaries, where the boundary flux q_Γ is assigned 0. The Neumann condition can be stated as

$$-K\left(\frac{\partial h(x, t)}{\partial \vec{n}} + n_3\right) = q_\Gamma,\ \forall (x, t) \in \Gamma \times [0, T), \tag{2.39}$$

where n_3 is the boundary normal ($\vec{n}$) vertical component.

Free drainage is a special boundary, where the pressure head gradient is zero and only the geodetic head gradient applies. It is a homogeneous Neumann boundary. In a vertically

aligned coordinate system the pressure head can be expressed as $\frac{\partial h}{\partial z} = 0$, otherwise $\frac{\partial h}{\partial n} = 0$. Assuming a vertically aligned coordinate system, this leads to:

$$\frac{\partial H}{\partial z} = \frac{\partial h + z}{\partial z} = \frac{\partial h}{\partial z} + \frac{\partial z}{\partial z} = 0 + 1 = 1. \tag{2.40}$$

3 Dual permeability model implementation

3.1 Introduction

The dual permeability model was implemented in the Dual Richards' Unsaturated Equation Solver (*DRUtES*) (Kuraz and Mayer 2008), an object oriented library written in Fortran 2003/2008, which can be installed on unix based systems. Dual permeability solute transport was not implemented as part of this thesis. This chapter briefly describes the implementation in *DRUtES* and presents test simulations. It should be noted that *DRUtES*, like most software, is under constant development and that this section refers to the implementation of the dual permeability model in early 2017.

3.2 DRUtES

DRUtES was first created as part of the doctoral research of Doc. Ing. M. Kuráž and is distributed under the GPL v3 licence. *DRUtES* is free software and can be redistributed and modified under the terms of the GNU General Public License. *DRUtES* can solve the nonlinear problem both with a standard Picard method and with so called dd-adaptivity. *DRUtES* is a command-line software and takes input from configuration files.

3.3 Implementation

The dual permeability model was implemented in total head form and is available in 1D and 2D. Fig. 2 depicts a *DRUtES* file tree relevant to the dual permeability model. *other* in Fig. 2 summarizes all other subroutines and functions, which are naturally vital to the functioning of the dual permeability model, but would complicate the description too much. The dual permeability implementation was divided into six separate source code files and added to the branch *models* in the DRUtES source code *src*. The folder containing the dual permeability source code was named Re_dual, and contains following file structure. *Re_dual_globals.f90* defines global variables that are used specifically in the dual permeability subroutines. Other global variables defined in *DRUtES* are also used. *Re_dual_reader.f90* reads the input values from dual permeability configuration files, which are located in *drutes.conf* branch. *Re_dual_pointers.f90* defines the targets to globally defined pointers. Here, the respective constitutive functions related to the convection-dispersion-reaction are pointed to, as are boundary and initial conditions. Case statements facilitate an easy switch between different boundary conditions and coupling variants. *Re_dual_totH.f90* contains the governing equations for the dual permeability model. *Re_dual_coupling.f90* contains different coupling variants. *Re_dual_tab.f90* contains subroutines to tabulate the governing equations. This can speed up the simulation

considerably. The values between two tabulated entries are linearly approximated.

The range and density of the tabulation can be defined in the configuration input files *dual.conf*. Here, the user also sets van Genuchten-Mualem parameters as well as the initial condition. The initial condition can be set per layer and is a value in pressure heads or total heads. For 2D computations, a newly added subroutine allows the user to define 1D output files with coordinates and pressure head information to be vertically mapped onto a 2D mesh to be used as the initial condition. The boundary conditions can be defined in a separate configuration file *dual_bc.conf*. Before running a simulation, the domain needs to be described and discretized. *DRUtES* contains a simple 1D and 2D mesh generator, but can also read Gmsh mesh files for more advanced meshes. Gmsh is a free three-dimensional finite element mesh generator with built-in pre- and post-processing facilities published under the GPL v3 licence and developed by Geuzaine and Remacle (2009). A tutorial describing how to build a Gmsh mesh to use in *DRUtES* is planned to be made available soon.

To select the dual permeability model, the user has to type the model name *Re_dual_totH* into the global configuration file *global.conf*. The *global.conf* file lets the user define computational set-up that is not model specific. Most important to casual users are possibly time discretization, observation times, observation points and input and output format.

The user can choose a range of boundary types. For time-varying boundary types additional data needs to be supplied. For a bottom boundary with id 101 a file with the filename 101.bc will be read. Following boundary types are currently implemented:

1. Dirichlet condition: constant pressure head;
2. Constant or time-varying flux. This is a Neumann type condition. In the dual permeability model, this condition assigns a weighted flux according to the domain weights $q_{f,\Gamma} = w_f q_\Gamma$ and $q_{m,\Gamma} = w_m q_\Gamma$;
3. Free drainage (also of type Neumann);
4. Atmospheric boundary condition (also of type Neumann). This requires precipitation and potential evapotranspiration (PET) data. The evapotranspiration (ET) is assumed to be $ET = PET\theta^{\frac{2}{3}}$. $q_\Gamma = rain - ET$. This special Neumann condition is also assigned weighted;
5. Weighted infiltration. The user can define separate weights for the infiltration. A weight of 1 assumes all of the water infiltrates into the fractures and a weight of 0 assumes all water infiltrates into the matrix. This is a modification of boundary type 2.

The coupling term contains an K_a term. K_a can be evaluated in different ways. In Gerke and Genuchten (1993a) it was evaluated as a function of the average pressure head $\bar{h}$. Gerke and Genuchten (1993b) undertook a more rigorous comparison of implementations of K_a finding

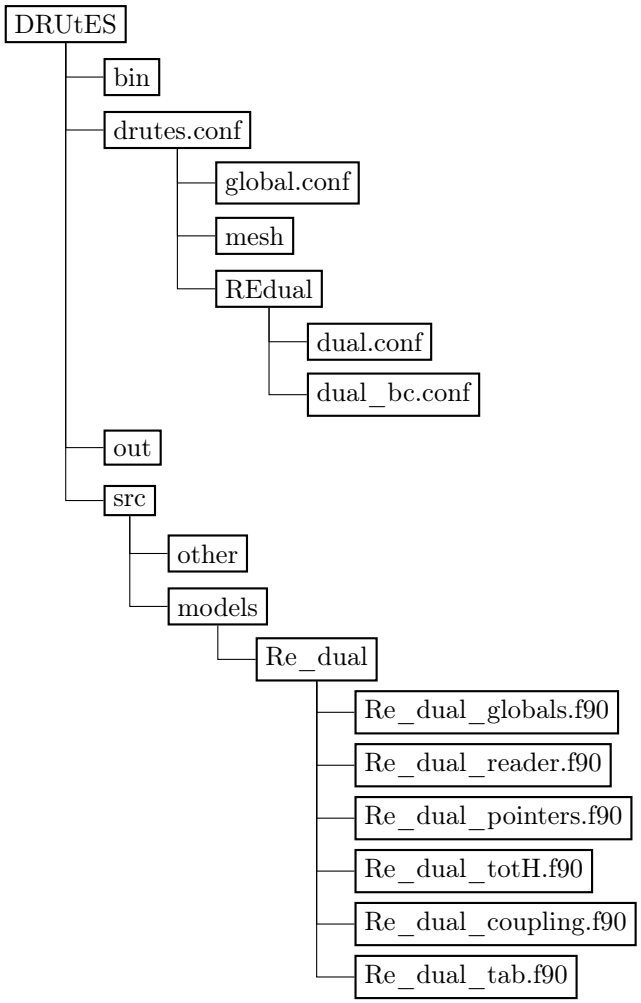


Figure 2: Simplified *DRUtES* file tree relevant to the implemented dual permeability model, where other represents the rest of the source code

significant difference between implementations. They compared five alternative methods with K_a being a function of

- the matrix pressure head $K_a = K_a(h_m)$,
- the fracture pressure head $K_a = K_a(h_f)$,
- the arithmetic mean of $\mathrm{K}_a(\mathrm{h}_f)$ and $\mathrm{K}_a(\mathrm{h}_m)$,
- the geometric of $\mathrm{K}_a(\mathrm{h}_f)$ and $\mathrm{K}_a(\mathrm{h}_m)$ and
- the integral of $\frac{1}{h_f - h_m} \int_{h_m}^{h_f} K_a$.

They concluded that the arithmetic mean in combination with a scaling coefficient was the most practical way of evaluating K_a. Gerke and Genuchten (1993b) also introduced an exponent p for the pressure head difference between the two pore systems. In other applications the exchange term is simply constant and independent of the pressure head in the media (Šimůnek et al. 2012). Šimůnek et al. (2012) state that the arithmetic mean is conceptually attractive, however it may not be identifiable.

There are five different variants available in the DRUtES implementation, each describing Ka differently. These require different parameterization and allow different flexibility. There are five different variants available in the $DRUtES$ implementation.

1. The arithmetic mean of h_m and h_f:

$$K_a = \frac{K_a(h_f) + K_a(h_m)}{2} \tag{3.1}$$

2. The geometric mean of h_m and h_f:

$$K_a = \sqrt{K_a(h_f) \cdot K_a(h_m)} \tag{3.2}$$

3. Constant conductivity:

$$K_a = const \tag{3.3}$$

4. Lowest possible hydraulic conductivity of matrix and fracture domain:

$$K_a = min(K_f(h_f), K_f(h_m), K_m(h_f), K_m(h_m)) \tag{3.4}$$

5. Lowest hydraulic conductivity of the weighted pressure heads:

$$K_a = min(K_f(w_f \cdot h_f + w_m \cdot h_m), K_m(w_f \cdot h_f + w_m \cdot h_m)) \tag{3.5}$$

Variant 4 and 5 do not require additional parameters for the exchange boundary, but are solely based on the matrix and fracture description. Variants 3.1, 3.2 and 3.3 allow assigning extremely small or large exchange boundary conductivity independent on the present system. These variants make non-equilibrium and interesting bypass flow simulations possible. The exchange boundary conductivity is likely to be very large when using variant 4 and 5, leading to relatively fast equilibrium between the two pore systems, however this is dependent on the simulation and can also be compensated for by other coupling parameters, but one should be aware of the physical meaning of these parameters. Variants 3.1 and 3.2 use van Genuchten parameterization and therefore lead to a potentially time-varying and pressure-head dependent exchange term beyond the dependence on the pressure head difference. Often, the matrix parameterization is used for the exchange boundary's van Genuchten parameterization as well. This is to avoid overparameterization of parameters that are very difficult to identify.

The mathematical subroutines solving the system of partial differential equation (PDE) were already implemented and are not my work. The following description aims to give a short, superficial overview. In this implementation, to solve the PDE dual permeability problem,

we multiply by the weights to reach following form:

$$\begin{aligned}(1-w_f)C(h_m)\frac{\partial h_m}{\partial t} &= (1-w_f)\nabla.(K(h_m)\nabla h_m) + (1-w_f)\frac{\partial K(h_m)}{\partial z} + \Gamma_w \\ w_f C(h_f)\frac{\partial h_f}{\partial t} &= w_f\nabla.(K(h_f)\nabla h_f) + w_f\frac{\partial K(h_f)}{\partial z} - \Gamma_w\end{aligned} \tag{3.6}$$

The PDE problems are solved by numerical integration with linear piece-wise Galerkin's finite element method. This means that the basis functions are linear, which are evaluated as a solution to a system of linear equations. The linear coefficients are stored in a sparse matrix, which is possible to store in computer memory even for huge dimensions. In finite element approximations, stability issues can occur due to small time steps compared to mesh size causing a violation of the discrete maximum principle. Stabilization subroutines are implemented to avoid the violation of the discrete maximum principle first described by Rank et al. (1983). As the hydraulic conductivity and retention capacity are non-linear functions, the solution in the elements is approximated using a three-point Gauss-quadrature formula, which is proven to give an exact integral value for fifth-degree polynomials (Cheney and Kincaid 2008).

The capacity matrix is assembled using the implicit Euler method. To avoid stability issues, the capacity matrix can be diagonalized by lumping. The conjugate gradient method is used to iteratively solve the derived system of equations for 2D problems. LU decomposition is implemented for 1D problems. The Picard method approximates the non-linear operator at each time level. The user of $DRUtES$ is able to determine the Picard iteration criterion and maximum iteration number in the global configuration files *global.conf*. The dual permeability problem is solved as a fully coupled problem, creating one stiffness matrix including the matrix and fracture domain such as

$$\left(\begin{array}{cccccccccccccccccccc}
\multicolumn{10}{c}{\overbrace{\textit{matrix domain}}} & \multicolumn{10}{c}{\overbrace{\textit{fracture domain}}} \\
a_{1,1} & a_{1,2} & . & . & \dots & . & . & . & & & a_{1,11} & . & . & . & \dots & . & . & . & & \\
a_{2,1} & a_{2,2} & a_{2,3} & . & \dots & . & . & . & & & . & a_{2,12} & . & . & \dots & . & . & . & & \\
. & a_{3,2} & a_{3,3} & a_{3,4} & \dots & . & . & . & & & . & . & a_{3,13} & . & \dots & . & . & . & & \\
\vdots & & & & & & & & & & \vdots & & & & & & & & & \\
. & . & . & . & \dots & a_{9,8} & a_{9,9} & a_{9,10} & & & . & . & . & . & \dots & . & a_{9,19} & . & & \\
. & . & . & . & \dots & . & a_{10,9} & a_{10,10} & & & . & . & . & . & \dots & . & . & a_{10,20} & & \\
a_{11,1} & . & . & . & \dots & . & . & . & & & a_{11,11} & a_{11,12} & . & . & \dots & . & . & . & & \\
. & a_{12,2} & . & . & \dots & . & . & . & & & a_{12,11} & a_{12,12} & a_{12,13} & . & \dots & . & . & . & & \\
. & . & a_{13,3} & . & \dots & . & . & . & & & . & a_{13,12} & a_{13,13} & a_{13,14} & \dots & . & . & . & & \\
\vdots & & & & & & & & & & \vdots & & & & & & & & & \\
. & . & . & . & \dots & . & a_{19,9} & . & & & . & . & . & . & \dots & a_{19,18} & a_{19,19} & a_{19,20} & & \\
. & . & . & . & \dots & . & . & a_{20,10} & & & . & . & . & . & \dots & . & a_{20,19} & a_{20,20} & &
\end{array}\right)$$

The system to solve is a combination of two PDE coupled by the exchange term. Therefore the matrix solution depends on fracture and the fracture's solution depends on the matrix.

The above matrix system can therefore be generalized into four blocks

$$\begin{pmatrix} \overbrace{MATRIX\ \ DOMAIN}^{matrix\ domain} & \overbrace{FRACTUE\ \ DOMAIN}^{fracture\ domain} \\ \begin{matrix} terms\ \ in \\ matrix\ \ operator \end{matrix} & \begin{matrix} terms\ \ in \\ fracture\ \ operator \end{matrix} \\ MATRIX\ \ DOMAIN & FRACTURE\ \ DOMAIN \\ \begin{matrix} Coupling\ \ term\ \ in \\ fracture\ \ operator \end{matrix} & \begin{matrix} Coupling\ \ term\ \ in \\ matrix\ \ operator \end{matrix} \end{pmatrix}$$

3.4 Test simulations

The author is not aware of any analytical verification functions that can be used for the dual permeability model. Nonetheless, attempts were made to prove the implementation is in working order. The first step in testing the implementation was to undertake a comparison between the dual permeability model variants and the already implemented and well tested standard Richard's total head model with uni-modal van Genuchten parameterization. The comparison is possible when the fracture and matrix descriptions are identical. With identical domain descriptions and the same boundary and initial conditions, the results should not depend on domain weights. 1D simulations with different boundary conditions (Tab. 20) were created and the outputs compared (Appendix A). The outputs were identical when sufficiently small time steps and discretization are chosen. Differences can occur due to numerics, when the set-up is insufficient or the problem is ill-conditioned. When the coupling term is set to zero, i.e. by setting the exchange boundary conductivity to zero the difference between the standard model and the dual permeability model may be decreased. Note, including a coupling term, which is not active, may worsen the conditioning, however it also assures that no exchange between two identical domains occur. Furthermore, different 1D simulations were constructed to show a) under which conditions preferential flow through the macropores occurs and b) the effect of different K_a on the water transfer between the domains and non-equilibrium.

Preferential flow

The macroporous or fracture domain is generally parameterized with a higher saturated hydraulics conductivity K_s than the matrix domain. Under saturated or near-saturated conditions this leads to faster flow through the macropore or fracture domain. This can cause infiltrating water to bypass the matrix. Macropores have rather steep retention properties. This reduces the hydraulic conductivity under unsaturated conditions. Under sufficiently dry conditions the unsaturated hydraulic conductivity K in the matrix domain becomes greater than in the fracture domain. When the fracture domain is not connected to free water, this can lead to an effect often termed "capillary barrier effect" as the capillaries of the fracture cannot hold water at greater tensions and therefore create a barrier and keep

the soil dry. This also raises the question if preferential flow through the macropores occurs under dry antecedent moisture conditions and if so, how it should be described. If one assumes ponding, a positive Dirichlet boundary condition can be applied, which assigns a very wet condition to the domain boundary, which can result in an extremely steep hydraulic gradient. However, when the boundary is of the Neumann type, the situation is quite different. First, the dual permeability model originally assigns flux according to the volume of each sub-domain. The volume of the fracture is generally a lot smaller so the fracture will get a smaller infiltration flux assigned. Second, the pressure head is initially very small despite assigned fluxes. One way of creating preferential flow under dry initial conditions is limiting the infiltration boundary flux into the matrix domain. This can be achieved using a sufficiently high weight with boundary type 5 explained previously. The fracture and matrix description (Tab. 1) were chosen so that the fracture becomes more conducting than the matrix domain at pressure heads greater than approximately $h_{intersect}$=-60 cm.

Coupling variants

The coupling term increases with increasing magnitude of the pressure head difference between the matrix and fracture domain. Smaller coupling terms lead to slower equilibration between the two pore systems, whereas greater coupling terms lead to faster equilibration between the two pore systems as differences in pressure head can be compensated for faster. With sufficiently large K_a the difference in pressure head can be compensated for quite rapidly. For soils near equilibrium it may be very difficult to identify K_a as the pressure head difference is zero or close to zero, which causes the coupling term to vanish. On the contrary, if K_a is sufficiently small the non-equilibrium can be maintained longer despite a large pressure head difference between the two pore systems.

Test simulation results

Scenario 1 and 2 were each run with three different infiltration weights and three different K_a. The simulation results are summarized in Fig. 3 for scenario 1 with dry initial conditions and Fig. 4 for scenario 2 with wet initial conditions. It becomes evident that larger infiltration weights with smaller exchange boundary conductivity K_a cause more water to bypass the matrix. This leads to minimal change of water content in the matrix. This effect can be compensated for by assigning greater exchange conductivity, which can annihilate the effect of bypass flow. The fracture parameterization exhibits medium steep retention properties. The fracture domain does not need to be fully saturated to conduct the simulated infiltrating water. This shows, if we allow infiltration flux, which is below the fracture's capacity, to bypass the matrix and mainly infiltrate into the fracture, the overall water content of the system does not necessarily need to increase. On the contrary, rapid changes in water content can indicate

Table 1: Soil hydraulic properties and domain description used for test simulations of the dual permeability model.

Domain	**Parameter**	**Symbol**	**Value**		
Matrix	inverse of air entry value [cm^{-1}]	α_m	0.05		
	shape parameter [-]	n_m	1.8		
	sat. water content [-]	$\theta_{s,m}$	0.45		
	res. water content [-]	$\theta_{r,m}$	0.05		
	sat. hydraulic conductivity [cm d^{-1}]	$K_{s,m}$	5		
	matrix weight [-]	w_m	0.7		
Fracture	inverse of air entry value [cm^{-1}]	α_f	0.1		
	shape parameter [-]	n_f	2.5		
	sat. water content [-]	$\theta_{s,f}$	0.45		
	res. water content [-]	$\theta_{r,f}$	0.0		
	sat. hydraulic conductivity [cm d^{-1}]	$K_{s,f}$	1000		
	fracture weight [-]	w_f	0.3		
Exchange boundary	distance to center of aggregate [cm]	a	1		
	material parameter [-]	β	0.4		
	material parameter [-]	γ	15		
	exchange boundary conductivity [cm d^{-1}]	K_a	1e-2	1e-4	1e-6 [a]
All	domain length [cm]	L	10		
	spatial discretization [cm]	dx	0.05		
	simulation time [days]	t	2		
	minimum time step [days]	dt_{min}	5e-8		
	maximum time step[days]	dt_{max}	0.1		

[a]constant exchange conductivity

Table 2: Boundary and initial conditions for dual permeability test simulations.

		Top bc			**Bottom bc**		**Initial h_{pres}**
		type[a]	value	w_{inf}	type[a]	value	
scenario 1	a	2		-			
	b	5	0.5 cm d^{-1}	0.7	1	-500 cm	-500 cm
	c	5		0.95			
scenario 2	a	2		-			
	b	5	2 cm d^{-1}	0.7	1	-60 cm	-60 cm
	c	5		0.95			

[a]1=Dirichlet, 2= Volume weighted flux, 3=Free drainage, 4=Atmospheric, 5=Weighted infiltration

that either the majority of the water is infiltrating into the matrix or that the exchange boundary conductivity is rather large resulting in fast equilibration of the two pore systems, but no bypass flow through the fracture domain. Furthermore, it becomes evident that the soil system becomes independent of the weighted infiltration boundary conditions when a sufficiently large exchange boundary conductivity is assigned resulting in fast equilibration and

the same water content and pressure head distribution in the different test simulations.

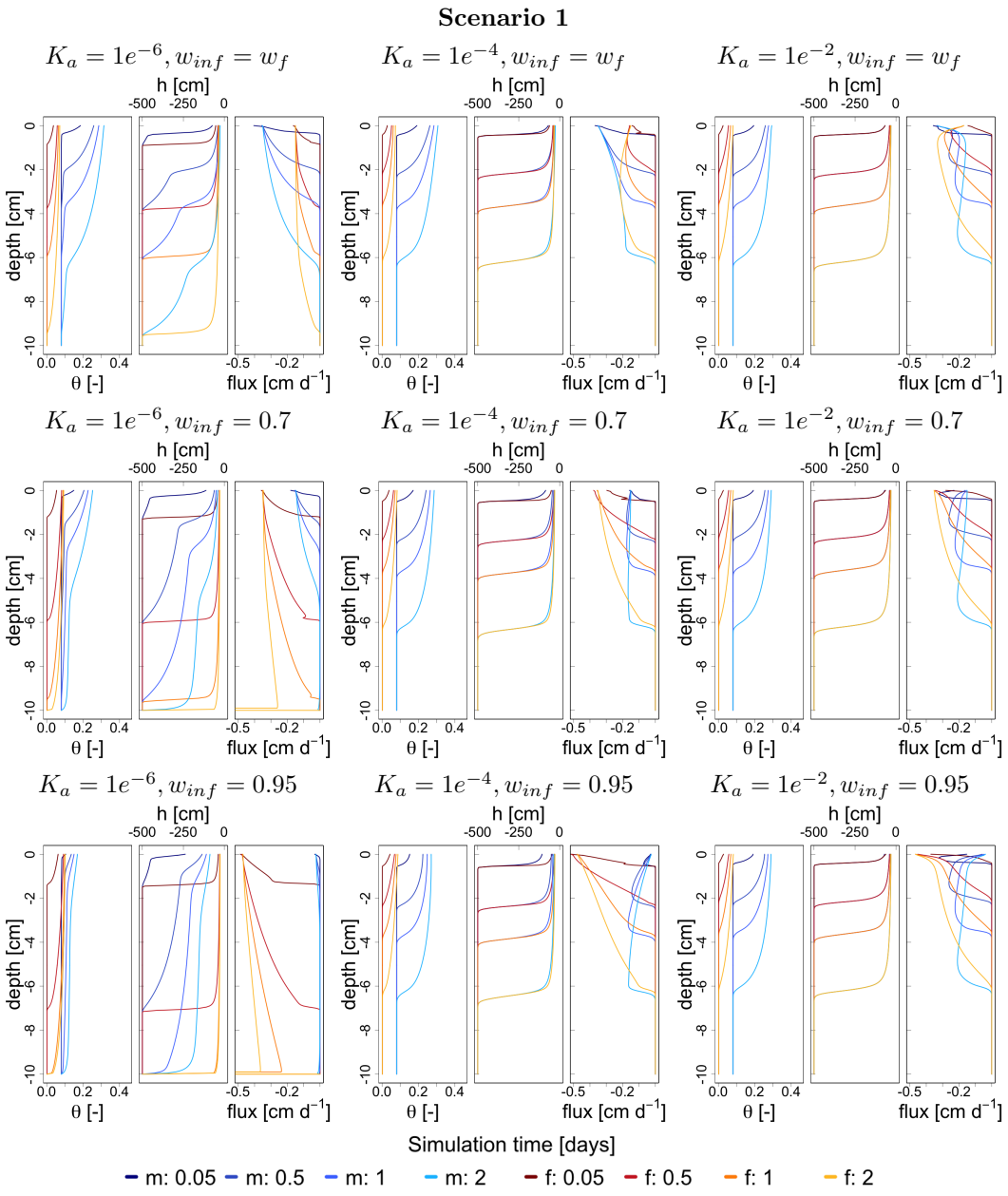


Figure 3: Simulation output after 0.1, 0.5, 1 and 2 days of scenario 1 (dry initial condition) with different exchange boundary conductivity terms K_a and infiltration weights w_{inf}, where red colors represent the fracture domain (f) and blue colors represent the matrix domain (m).

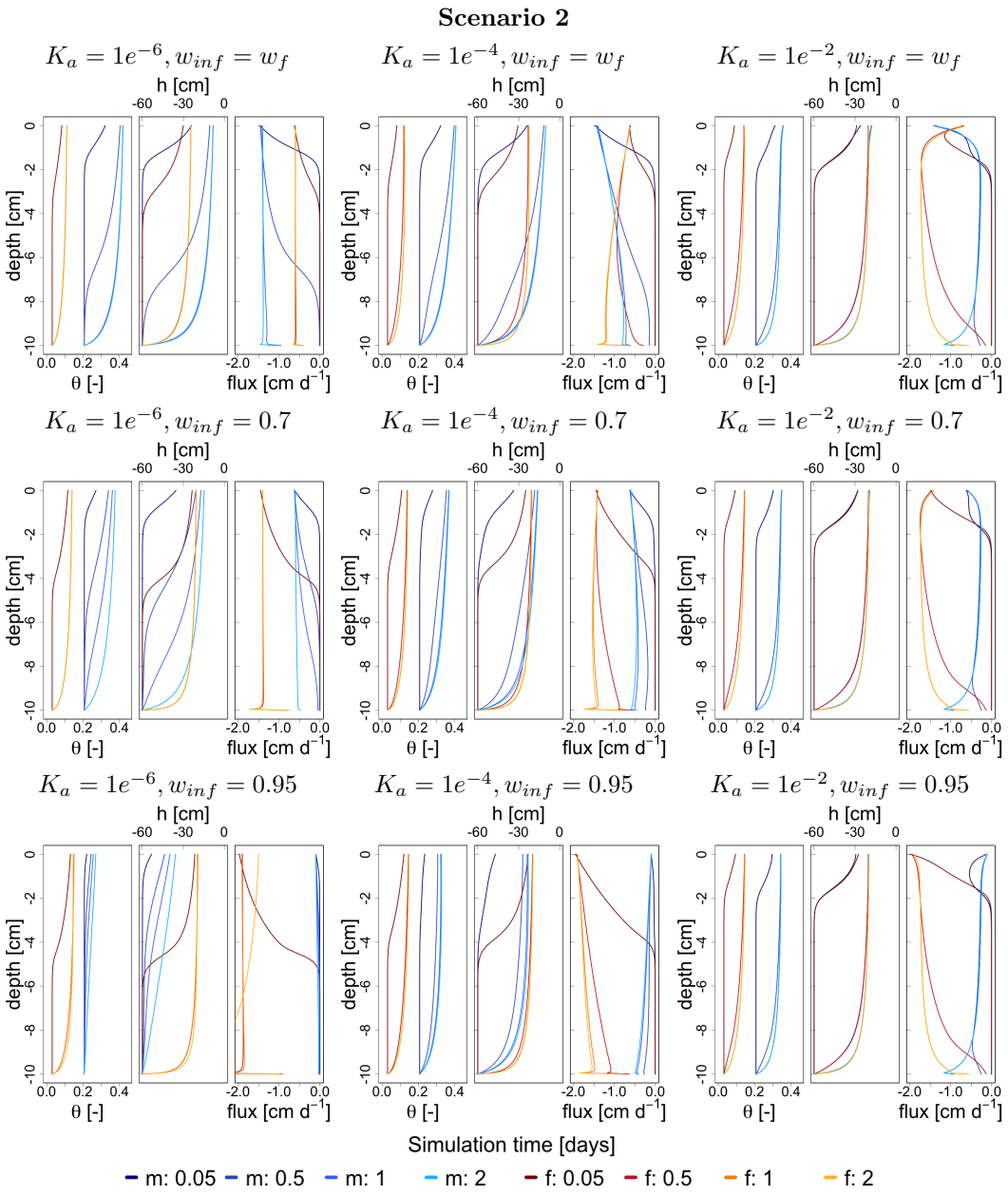


Figure 4: Simulation output after 0.1, 0.5, 1 and 2 days of scenario 2 (wet initial condition) with different exchange boundary conductivity terms (K_a) and infiltration weights (w_{inf}), where red colors represent the fracture domain (f) and blue colors represent the matrix domain (m).

4 Population-based metaheuristics

This section describes principles of metaheuristics with a focus on population-based metaheuristics. In this thesis two classes of population-based metaheuristics, namely Particle Swarm Optimization (PSO) (Eberhart and Kennedy 1995) and Teaching-Learning Based Optimization ($TLBO$) Rao et al. (2012) were used. Descriptions of basic PSO and $TLBO$ algorithms and modifications used in this thesis are given. This section concludes with a description of adaptations aiming at an increase of diversity and handle premature convergence.

4.1 Principles of Metaheuristics

Metaheuristics summarize high level concepts for exploring search spaces by using different learning strategies to find near-optimal solutions. The different strategies should be chosen to balance exploitation and exploration to quickly identify regions with high quality solution and not waste time searching regions that don't provide high quality solutions (Blum and Roli 2003 in Bianchi et al. 2009). An important part is the incooperation of structured randomness. In principle, algorithms of metaheuristic nature sample a set of potential solutions instead of the entire search space to find the optimal solution. In a minimization problem, where $G(x)$ is the cost function, the goal is to find $min\ \ G(x)$, where $x \in S$ and S is the search space. The global optimal solution at $x*$ can be defined as

$$G(x*) \leq G(x)\ \ \forall\ \ x \in S. \tag{4.1}$$

Whereas exact optimization guarantees to find the optimal solution in a finite amount of time, in heuristics, the best solution might not be the global best solution (Sörensen 2015). There are many hard optimization problems where it would be unfeasible to use exact methods and heuristics are needed. Bianchi et al. (2009) describe several optimization problems where metaheuristics were superior to exact methods. It is not easy to theoretically prove the efficiency of metaheuristic algorithms and studies usually rely on empirical results. One great advantage of metaheuristic algorithms is the variety of objective functions that can be used. The objective functions can be non-smooth and discontinuous. Furthermore, no gradient information or Hessian matrix are used. Information on the objective function's derivative is therefore not necessary.

There are several ways to update solutions. Bianchi et al. (2009) distinguish between two groups of heuristic algorithms, constructive and local search algorithms. Constructive algorithms build a solution by joining together pieces that are added one after the other until a solution is complete. Local search algorithms try to improve a current solution by modifying some of its components.

Metaheuristics often require parameter tuning that can highly influence the quality of the search (Das and Suganthan 2011, Boussaïd et al. 2013).

Several concepts or classifications of metaheuristic optimization algorithms exist. The groups often overlap or are a subset of one another but also stress different aspects of their optimization approach; or are named to distinguish between entirely different optimization techniques. Terminologies of these groups include soft computing, evolutionary optimization algorithms (Simon 2013) or evolutionary computing (Bianchi et al. 2009), random search methods (Amaran et al. 2015), bio-inspired optimization (Kar 2016), population-based optimization and computational swarm intelligence (Engelbrecht 2005). One classification example was created by Dréo (2011) (Fig. 5).

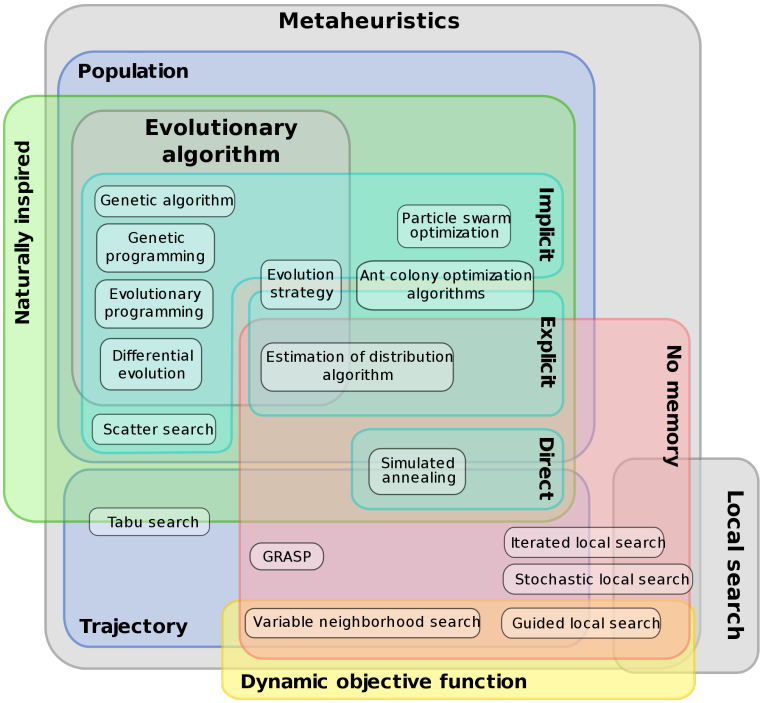


Figure 5: Different classification of metaheuristics (Dréo (2011), copyright: Creative Commons Attribution-Share Alike 3.0)

Gogna and Tayal (2013) compare characteristics of metaheuristics and classify them as follows:

- Nature-inspired vs. non-nature-inspired

This distinguishes the algorithm based on where the idea for the algorithm comes from. It can also give insight on the working principle. For nature-inspired algorithms these are often metaphor-based, whereas for non-nature-inspired algorithms they are more descriptive.

- Population-based vs. single-solution-based

Single-solution-based methods manipulate a single solution, whereas population-based methods iterate and manipulate a whole set of solutions.

- Iterative vs. greedy

Iterative algorithms start from a solution that is then manipulated in the search process, whereas greedy algorithms start with an empty solution, and at each stage, a decision variable is added to the solution set.

- Memory usage vs. only Current State

Current State only uses information about the current state of the search, while others use information gathered through the iterative search.

- One neighborhood vs. multiple neighborhoods

Algorithms can make use of one or multiple neighborhood structures.

In population-based metaheuristics, swarm intelligence is a fundamental concept. Engelbrecht (2005) describes the very basic principle of swarm intelligence with a simple treasure hunt analogy. In this illustration, the approximate area, but not the exact location, of the treasure is known (search space). Each person is equipped with a metal detector and and can communicate the strength of the signal (evaluation of the objective function) and the current location. The basic underlying assumption is that communicating with your peers increases the chance of finding the treasure (optimum).

The reasons why the two classes of algorithms, PSO and $TLBO$, were chosen has partly to do with current trends in metaheuristics and the different design approaches of the two, which will be explained in the following. The author decided to focus on population-based metaheuristics and chose, on the one hand, a well established class of algorithms, namely PSO, which was first introduced in 1995 by Eberhart and Kennedy (1995) and on the other hand, a relatively new class of algorithms, $TLBO$ introduced in 2012 by Rao et al. (2012), which has received increasing attention. One might argue that PSO and $TLBO$ only refer to algorithms and not a class per se. However, due to the manifold of improvements and modifications of these two algorithms, it appears appropriate to to use a wider term such as class.

Metaheuristics is still a growing field. Improvements of algorithms are generally accomplished by implementing modifications or by hybridizing different algorithms. Overall, to determine which metaheuristic optimization algorithms are state of the art proved to be a challenge. In recent years there has been a notable increase in metaphor-based metaheuristic algorithms. Seemingly anything, from insects to groups of animals and even jazz music are chosen as metaphors for new algorithms (Sörensen 2015). On top of the vast quantity of algorithms, there appears to be an issue with the reinvention of already existing algorithms (Weyland 2010). Sörensen (2015) details how metaphor-based algorithms have contributed rather little to the scientific field and how the research is difficult to follow when metaphor-based terminologies, such as *pheromones*, *waggle dance* or *note* are used instead of

understandable terminologies, such as solution or objective function. For an overview on algorithms, the author would like to refer to following review papers: Bäck and Schwefel (1993), Bianchi et al. (2009), Das and Suganthan (2011), Boussaïd et al. (2013), Gogna and Tayal (2013) and Amaran et al. (2015) and following book: Simon (2013). This is by far not a complete selection but provides great starting material.

Despite the above criticism, some algorithm specific terminologies are used. For PSO the terminologies swarm or population are used for a set of potential solutions. The term particle denotes one specific solution at a given time or iteration. For $TLBO$ the term class is used to denote a set of potential solutions, the term teacher is used to refer to the global best solution and the term learner is used for one potential solution at a given time or iteration.

4.2 Particle Swarm Optimization

The *PSO* algorithm consists of a swarm of particles, where each particle represents a potential solution (Engelbrecht 2005). The particles move through the search space and adjust their position based on both their own experience and the experience of the neighborhood. In parameter optimization a particle at a given time represents a parameter set for the given problem.

4.2.1 Basic *PSO*

When $x_i(t) = (x_1^i, x_2^i, ...x_{dim}^i)$ is the position of particle i in the multi-dimensional search space at time t, the position is changed by adding a velocity term $v_i(t) = (v_1^i, v_2^i, ..., v_{dim}^i)$, so that $x_i(t+1)$ is

$$x_i(t+1) = x_i(t) + v_i(t+1). \tag{4.2}$$

The velocity term for dimension d is updated using inertia, cognitive and social components, which act as attractors (Fig. 6). The updated velocity is given by

$$v_d^i(t+1) = wv_d^i(t) + c_1 r_{1d}(t)[P_d^i(t) - x_d^i] + c_2 r_{2d}(t)[G(t) - x_d^i], \tag{4.3}$$

where c_1 and c_2 are acceleration constants and r_1,$r_2 \sim$ U(0,1). The previous velocity term wv_d^i of particle i in dimensions $d = 1, ..., dim$ serves as a memory of previous flight direction and is often referred to as inertia component. $c_1 r_{1d}[P_d^i(t) - x_d^i]$ represents a cognitive component retaining information that was best for the particle. $c_2 r_{2d}(t)[G(t) - x_d^i]$ is a social component, which draws particle i towards the best particle in the neighborhood. The personal best $P_i(t)$ of a particle i is the best position a particle has visited since the initialization. It is only updated when a better personal best position was found. For a minimization problem, where $f(x)$ represents a function evaluation, the personal best position at time t can be formulated so that

$$P_i(t)|f(P_i(t)) \leq f(x_i(T)) \; \forall \; T \in [0, t]. \tag{4.4}$$

For global best or gbest model the global best position G is stored. The global best can be defined as the best personal best position in the swarm, for a minimization problem the global position at time t can be defined as

$$G(t) \in \{P_0(t), ..., P_n(t)\}|f(G(t)) = min\{f(P_0(t)), ..., f(P_n(t))\} \tag{4.5}$$

where n denotes the number of particles in the swarm. This definition is the best position discovered by any particles so far. Engelbrecht (2005) suggests that the global best position

can also be calculated from the current swarm

$$G(t) \in \{x_0(t), ..., x_n(t)\} | f(G(t)) = min\{f(x_0(t)), ..., f(x_n(t))\}. \tag{4.6}$$

For local best or lbest model the local best position L is stored. The swarm is divided into neighborhoods and each particle is connected to k particles. The local best can be defined as the best position in the neighborhood. For a minimization problem the local position at time step t can be defined as

$$L(t) \in \{x_0(t), ..., x_{k+1}(t)\} | f(L(t)) = min\{f(x_0(t)), ..., f(x_{k+1}(t))\} \tag{4.7}$$

when a particle is connected to k particles, the neighborhood consists of $k + 1$. Different numbers of k can lead to varying topologies, e.g. $k = 2$ creates a ring structure. The velocity term of a local best PSO would be updated with the social component containing $L(t)$ instead of $G(t)$:

$$v_d^i(t+1) = v_d^i(t) + c_1 r_{1d}(t)[P_d^i(t) - x_d^i] + c_2 r_{2d}(t)[L(t) - x_d^i]. \tag{4.8}$$

The initial velocities were set to zero, i.e.

$$v(t) = v_{init} = 0, \quad \text{if } t = 0 \tag{4.9}$$

to simulate a stationary initial condition. It is also possible to initialize velocities at random, which violates the assumed stationary initial condition. Furthermore randomly initialized velocities can cause large velocities, which in return can cause particles to leave search space boundaries and may result in large position updates. This can cause the swarm to require more iterations before settling on a single solution (Engelbrecht 2005).

Algorithm 1 Generalized basic *PSO*

```
1:  procedure PSO
2:      Uniformly initialize each particle in the swarm S
3:      Evaluate swarm S
4:      P ← S
5:      Compute global best or local best
6:      repeat
7:          for each particle in i=1,...,n do
8:              Update velocity according to Eq. 4.3
9:              Update position according to Eq. 4.2
10:             Evaluate fitness value
11:             Update personal best if new particle is better than current personal best
12:         Update global best or local best if new particle is better than current global best
    or local best
13:     until stopping criterion is true
```

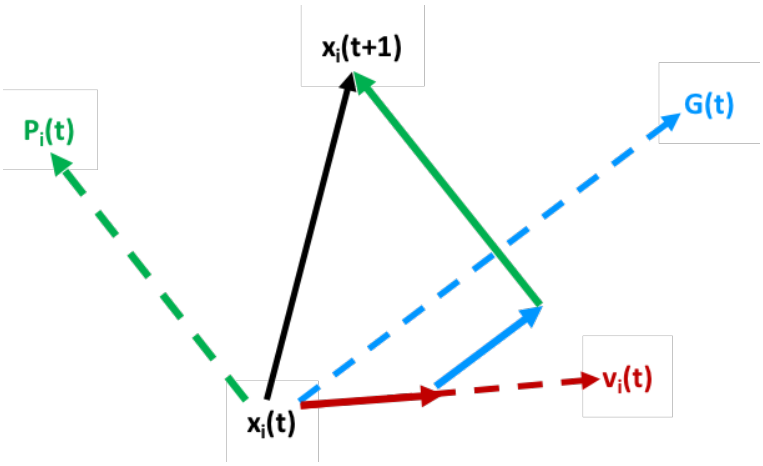


Figure 6: Working principle of updating position $x_i(t)$ of particle i to $x_i(t+1)$ depending on the inertia, cognitive and social attractor. The dashed lines indicate the direction of the attractors and the solid lines indicate the length of each attracting component leading to the final updated position $x_i(t+1)$.

4.2.2 Implemented modified *PSO* algorithm

Common modifications include velocity clamping, modifications of the inertia term w and the acceleration constants c_1 and c_2. It is with these modifications where tuning parameters becomes important. All modifications aim at improving the exploration-exploitation trade-off. It is often desired to have high exploration in the beginning with increased exploitation towards the end.

For all *PSO* algorithms a maximum velocity v_{max} was assigned. The maximum velocity helps to slow down particles that are too fast and are likely to overshoot an optimum. v_{max} has distinct value for all dimensions. If a particle is too fast the updated velocity $v_i(t+1)$ is set equal the maximum velocity. Note, depending on the direction of the particle, v_i can also be negative and velocity clamping has to be applied in two directions.

Assuring exploration in the beginning and exploitation as the search proceeds, v_{max} decreases linearly with time. In this thesis the following condition applied, where v_{max} was set relative to the dimension space:

$$v_{max}(t) = (v_{max\_ini} - v_{max\_end})\frac{t_{max} - t}{t_{max}} + v_{max\_end}. \tag{4.10}$$

where $v_{max\_ini} = \frac{x_{max} - x_{min}}{2}$ and $v_{max\_end} = \frac{x_{max} - x_{min}}{20}$. In this thesis the results where ranked. When ties occurred the tied particles were ranked randomly. The rank of each particle was then used to further refine v_{max}. Particles with worse results would be allowed to be faster and explore, whereas particles with better results would be slowed down and motivated to exploit. The particle's v^i_{max} is

$$v^i_{max}(t) = \frac{v_{max(t)}}{n - rank_i + 1} \tag{4.11}$$

where n denotes the total number of particles in the swarm and $rank_i$ is the rank of the particle.

Often, high level problems have local minima. It is desirable to get the particles out of the minima. The gbest PSO algorithms in this thesis were modified in order to increase their v_{max} when the global best position has not changed for 10 consecutive generations. The terminology generation originates from evolutionary algorithms and refers to iteration of an updating cycle. v_{max} continues to increase as long as no improvement occurs. However, v_{max} was not allowed to become greater than the initial v_{max}.

Self-adapting and time-varying inertia weights and acceleration coefficients have been suggested by various authors (Shi and Eberhart 1998, Ratnaweera et al. 2004, Naka et al. 2003). Similarly to the implemented velocity clamping, the inertia term w and the acceleration constants c_1 decrease linearly with time. Acceleration constant c_2 increases linearly with time to create increasing attracting to the global best. The linear decrease for the inertia term is

$$w(t) = (w_{max} - w_{min})\frac{t_{max} - t}{t_{max}} + w_{min} \tag{4.12}$$

which was first suggested by Shi and Eberhart (1998) with recommended $w_{min} = 0.4$ and $w_{max} = 0.9$. This was modified to $w_{min} = 0.2$ as it improved the performance.

The linearly decreasing acceleration constant simulating the cognitive component c_1 was implemented as follows

$$c_1(t) = (c_{1,max} - c_{1,min})\frac{t_{max} - t}{t_{max}} + c_{1,min}, \tag{4.13}$$

where $c_{1,max} = 1.5$ and $c_{1,min} = 0.1$, which were found to be a working boundaries by trial and error.

The linearly increasing acceleration constant simulating the social component was set as follows

$$c_2(t) = (c_{2.max} - c_{2,min})\frac{t}{t_{max}} + c_{2,min}. \tag{4.14}$$

where $c_{2,max} = 2.5$ and $c_{2,min} = 0.5$ as suggested by Ratnaweera et al. (2004).

The idea behind the linearly changing acceleration coefficients is, again, to allow exploration and wandering of particles in the beginning and an increasingly stronger attraction to the global best as time passes allowing exploitation of the most feasible region found. These modifications are hoped to be somewhat problem independent without the need for much parameter tuning.

4.2.3 Bi-objective *PSO*

There are several ways to adapt PSO algorithms to solve bi-objective and multi-objective problems, where bi-objective is the simplest multi-objective. These algorithms are termed Multi-objective Particle Swarm Optimization ($MOPSO$). The difference to single-objective problems is the need to optimize two (bi) or more (multi) objectives at the same time. A global optimizer can still be used when a weighting scheme is applied, where the objective function becomes a weighted sum of all objective functions in question. Multi-objective optimization often does not result in a single solution, but a set of equally optimal non-dominant solutions called a Pareto front. Only in rare cases are multi-objective solutions single-dominant leading to one solution. A Pareto front with global optimization can be obtained by changing weights and rerunning the optimization.

The bi-objective algorithm implemented is a modified version of the dynamic neighborhood $MOPSO$ developed by Hu and Eberhart (2002) and described in Engelbrecht (2005). New neighborhoods are chosen every iteration. Each particle has two neighboring particles. In the original approach neighborhoods are determined on the basis of the simplest objective function f_1. Neighbors are the two closest particles based on the objective function value f_1. The different neighborhoods therefore overlap to allow information to flow through the swarm. The best neighborhood particle is determined by the second objective function f_2 and replaces the position of the gbest particle in equation 4.3 for particle i. The personal best of particle i is only updated when both objective function values have improved. This way the personal bests of the swarm converge towards the Pareto front.

$$v_d^i(t+1) = wv_d^i(t) + c_1 r_{1d}(t)[P_d^i(t) - x_d^i] + c_2 r_{2d}(t)[Nbest^i(t) - x_d^i], \tag{4.15}$$

The original dynamic neighborhood $MOPSO$ is sensitive to ordering of the objectives. Therefore the ordering was swapped every iteration to assure that both objectives are treated equally. The neighborhood particles were chosen on the basis of ranks, where the neighborhood of particle i of rank k, had one neighbor of rank $k-1$ and one neighbor of rank $k+1$. The particles at the outer edges were assigned neighbors of rank $n-1$ and $n-2$ for the worst particle, where n denotes the number of particles in the swarm, and the ranks 2 and 3 for the best particle. The pseudocode of the bi-objective PSO is presented in algorithm 2.

Algorithm 2 bi-objective PSO

```
procedure BI-OBJECTIVE PSO
    Uniformly initialize each particle in the swarm S
    Evaluate swarm S for f1 and f2
    Set order of objective function
    Rank swarm S based on first objective function
    repeat
        for each particle in i=1,...,n do
            Find neighbors based on rank
            Compute neighborhood best based on second objective function
            Update velocity according to Eq. 4.15
            Update position according to Eq. 4.2
            Evaluate fitness value for f1 and f2
            Only update personal best if both f1 and f2 improved
        Swap order of objective function
        Rank swarm S based on first objective function
    until stopping criterion is true
```

4.3 Teaching-Learning-Based Optimization

Generally, metaheuristic algorithms contain parameters that require tuning to work optimally. The $TLBO$ is an algorithm that was designed as a global search optimization algorithm without additional tuning parameters (Rao et al. 2012). The TLBO method is based on the philosophy of teaching and learning, where the teacher is considered a highly learned person (best solution) sharing knowledge with learners. Additionally, learners learn from interactions with others. To be in accordance with literature, we will use following analogies. Learners are the analogy for a current solution. The entire set of current solutions make up a class. The global or local best solution is the teacher.

4.3.1 Basic $TLBO$

The algorithm is divided into a teacher's and a learners' phase. The basic idea is that the teacher can change the mean of the class. In the teacher's phase the ith learner x_i is updated using

$$x_i(t+1) = x_i(t) + r(T(t) - T_F mean(t)). \tag{4.16}$$

with T is the teacher at time $t, r \sim U(0,1)$, T_F is a teaching factor and heuristically computed as $T_F = round[1 + r(0,1)\{2\ 1\}]$ and mean is the mean of each design variable.

In the learners' phase, two other random learners x_j, x_k are selected, where $x_i \neq x_j$. If x_j is better than x_i, x_i is updated

$$x_i(t+1) = x_i(t) + r(x_j - x_i) \tag{4.17}$$

and if x_j is better than x_i,

$$x_i(t+1) = x_i(t) + r(x_i - x_j). \tag{4.18}$$

The objective function is evaluated after each phase and the solution is only accepted when it has improved.

Algorithm 3 Generalized basic $TLBO$

```
1: procedure TLBO
2:     Uniformly initialize each learner in class C
3:     Evaluate class C
4:     Find teacher
5:     repeat
6:         for all learners in i=1,...,n do
7:             Calculate teaching factor TF = round(1 + r(0, 1))
8:             Update position according to Eq. 4.16
9:         Evaluate new learners
10:        Accept new solution if it's better than the old one
11:        for all learners in i=1,...,n do
12:            Randomly select different learner
13:            Update learner according to Eq. 4.17 or 4.18
14:        Evaluate new learners
15:        Accept new solution if it's better than the old one
16:    until stopping criterion is true
```

4.3.2 Implemented modified *TLBO*

Zou et al. (2015) proposed additional strategies for the teacher's and learners' phase to increase learning efficiency that outperformed the basic $TLBO$. The modified $TLBO$ algorithm with the learning experience of other learners, namely Learning experience Teaching-Learning Based Optimization ($LETLBO$) was used in this thesis.

During the teaching phase, Zou et al. (2015) introduced an area copying operator, which is also used in Producer-Scrounger model. This area copying operator is applied stochastically. Two randomly generated values a and b are compared, where $a, b \sim U(0, 1)$. If $a < b$, Eq. 4.16 is used. If $b > a$, another learner x_j is selected, where $x_j \neq x_i$, is chosen and the position is updated based on previous results of x_i and x_j. If $f(x_j) < f(x_i)$ the ith learner x_i is updated using

$$x_i(t+1) = x_i(t) + r(T(t) - x_j(t)) \tag{4.19}$$

and if $f(x_i) < f(x_j)$ the ith learner x_i is updated using

$$x_i(t+1) = x_i(t) + r(T(t) - x_i(t)) \tag{4.20}$$

where $r \sim U(0, 1)$.

Similarly to the teacher's phase, two learning strategies are applied stochastically. Two randomly generated values a and b are compared, where $a, b \sim U(0, 1)$. If $a < b$, Eq. 4.17 or 4.18 are used to update the ith learner. If $b > a$, two random learners x_j, x_k are selected,

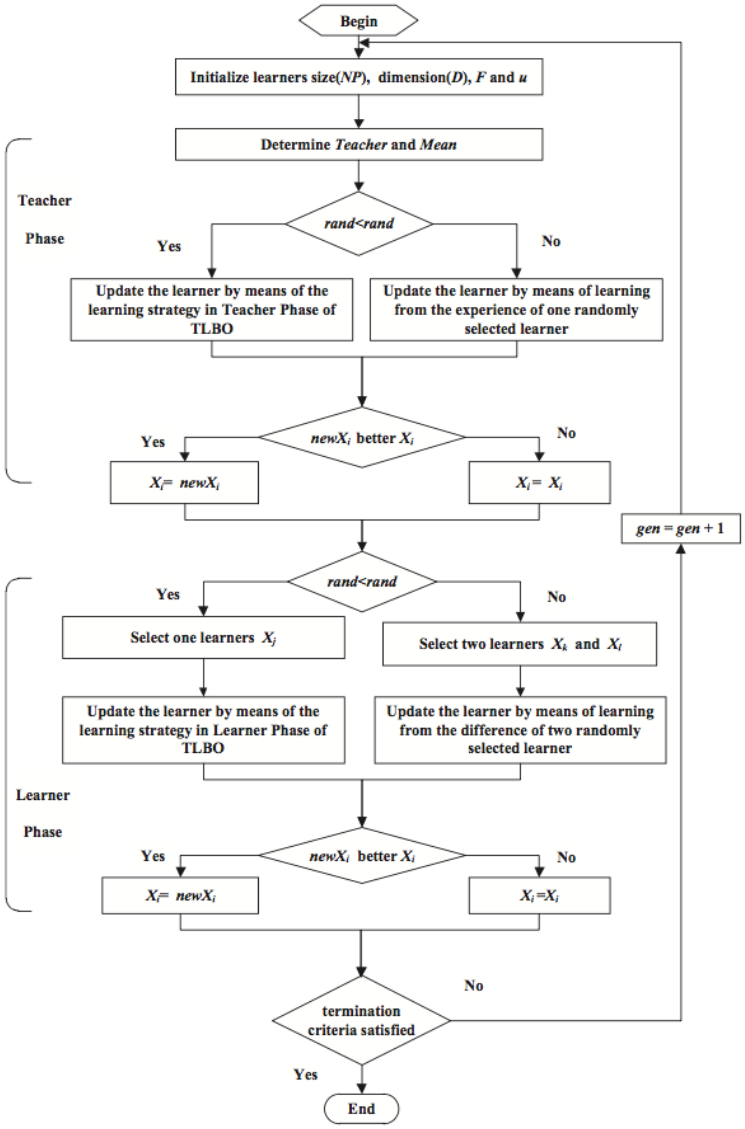


Figure 7: Flow chart of *LETLBO* (Zou et al. 2015).

where $x_i \neq x_j \neq x_k$. If x_j is better than x_k, x_i is updated with

$$x_i(t+1) = x_i(t) + r(x_j - x_k) \tag{4.21}$$

and if x_j is better than x_k, x_i is updated with

$$x_i(t+1) = x_i(t) + r(x_k - x_j). \tag{4.22}$$

Utilizing a difference operator can be found in other algorithms such as differential evolution.

4.4 Implemented variants

Two variants were developed and applied to modified PSO and $TLBO$ algorithms. The first variant aims to utilize different neighborhood topologies by dividing the population into complexes. The second variant aims to utilize the result space to improve the population by determining bad neighborhoods.

4.4.1 Shuffling complexes

Shuffling complexes is a well applied mechanism. It is part of the Shuffled Complex Evolution Approach developed by Duan et al. (1993), who combined elements of evolutionary algorithms with a partition of the particles into complexes. The algorithm developed by Duan et al. (1993) is often referred to as Shuffled Complex Evolution University of Arizona ($SCE-UA$). Each complex is treated as a separate population and evolved separately. Yan et al. (2007) developed a Shuffled Complex Evolution (SCE) algorithm for the PSO algorithm.

The clustering or the division of the population of complexes can be realized in different ways. The complexes can be created using random permutation or using rank information. Jakubcová et al. (2014) compared both shuffling variants in PSO algorithms and reported that rank based complexes performed better at most tested benchmark functions. In this thesis the complexes were created based on the rank.

The shuffling can occur after each generation or after a set number of generations. In this thesis a set generation of 10 was used. This means that for 10 generations the population is evaluated as one. Then the population is shuffled into complexes based on rank, where they would remain for 10 generations. This is depicted in Fig. 8.

4.4.2 Bad neighborhood approach

In inverse modeling, a function evaluation can take a very long time, which created the motivation to create a strategy that utilizes the result of the function evaluation to improve the location of particles. In this variant bad neighborhoods are determined. The bad neighborhoods are determined after each population evaluation based on the result of the function evaluation. The algorithm then performs the position update. With the updated position, the bad neighborhood strategy checks whether a particle landed in a bad neighborhood and if so, is motivated to move towards the global or complex best as follows

$$x_{new} = x_{inbadhood} + r(G - x_{inbadhood}) \tag{4.23}$$

where $r \sim U(0,1)$ and G denotes the global best or complex best.

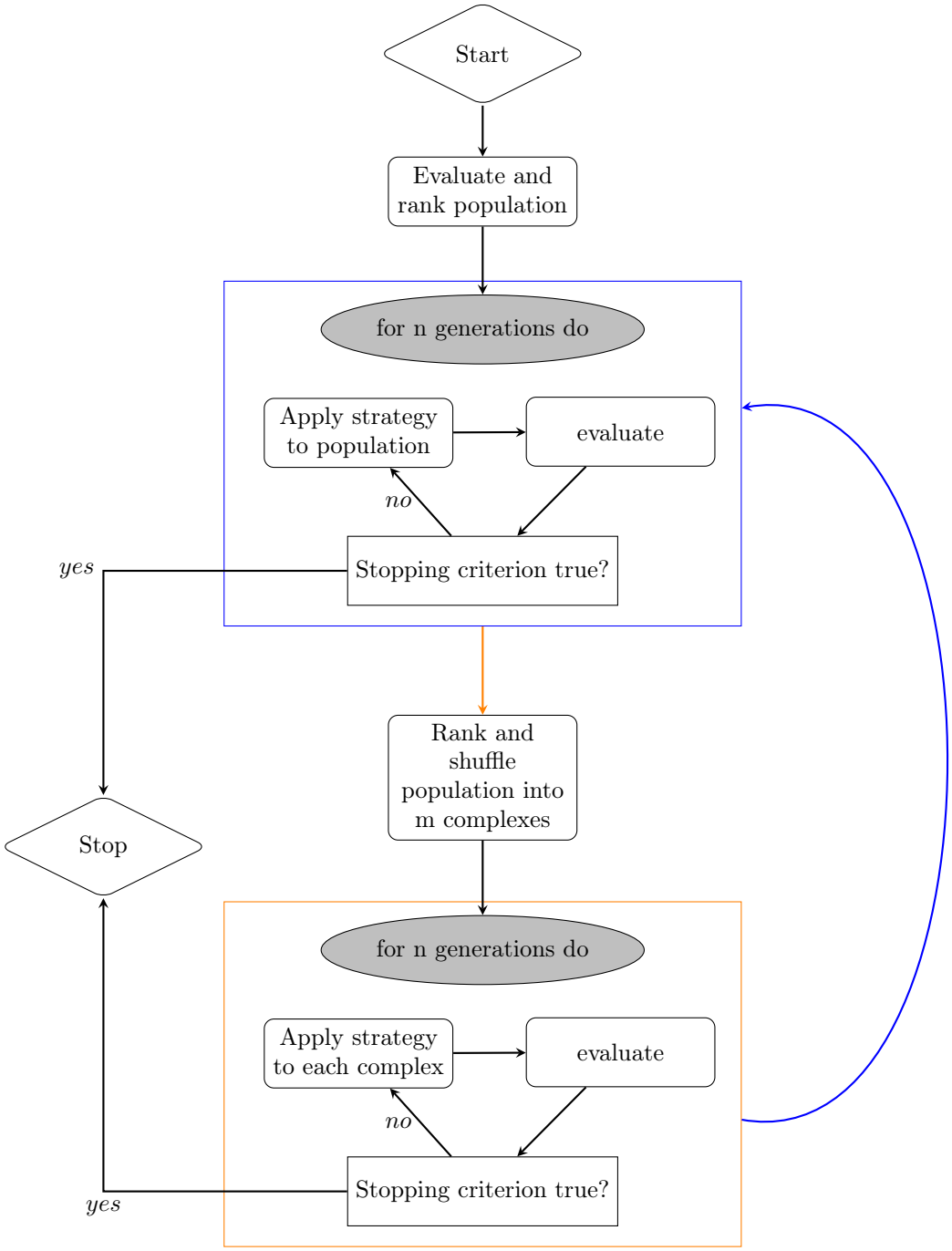


Figure 8: Simplified flow chart of implemented shuffling mechanism with switch between population as one complex and population shuffled into m complexes (= multi complex) after n generations.

This was thought to be a simple mechanism to improve a particle's or learner's location. The worst half of the population is considered to create a bad neighborhood. The neighborhood can be imagined as a bubble surrounding each bad particle. The worst particle creates the greatest bad neighborhood. The radius of the bad neighborhood decreases linearly with decreasing rank. In this implementation bad neighborhoods are allowed to overlap. If the neighborhood radius is too large, this mechanism facilitates a social component, where the entire population is drawn towards the global best, which can cause premature convergence. Whereas, if the neighborhood radius is too short, this mechanism has no effect. It was decided to start with greater bad neighborhood radii to leave unpromising areas quickly and subsequently decrease the bad neighborhood radii with each iteration. The decrease in bad neighborhood radii is hoped to counteract the likelihood of premature convergence.

Algorithm 4 bad neighborhood approach

procedure BAD NEIGHBORHOOD
    Uniformly initialize population
    Evaluate population
    Rank population
    Determine bad neighborhoods
    for all bad neighborhoods in j=1,...,n/2 **do**
        calculate bad neighborhood radius(j) for all design variables based on rank
        $sphere_{min}(j) = x_j - radius(j)$
        $sphere_{max}(j) = x_j + radius(j)$
    repeat
        for all particles in i=1,...,n **do**
            Update position according to algorithm strategy
            if x_i is in any bad neighborhood **then**
                Update position according to Eq. 4.23
        Rank population
        Determine bad neighborhoods
        for all bad neighborhoods in j=1,...,n/2 **do**
            calculate bad neighborhood radius(j) for all design variables based on rank and iteration
            $sphere_{min}(j) = x_j - radius(j)$
            $sphere_{max}(j) = x_j + radius(j)$
    until stopping criterion is true

4.4.3 Summary of variants

Eight different global search algorithms were implemented

- modified PSO (PSO)
- modified Particle Swarm Optimization with a bad neighborhood approach (PSO_{bn})
- modified Particle Swarm Optimization with a shuffling complexes approach (PSO_{sce})
- modified Particle Swarm Optimization with a bad neighborhood and shuffling complexes approach (PSO_{scebn})
- Learning experience $TLBO$ ($TLBO$)

- Learning experience Teaching-Learning Based Optimization with a bad neighborhood approach ($TLBO_{bn}$)
- Learning experience Teaching-Learning Based Optimization with shuffling complexes approach ($TLBO_{sce}$)
- Learning experience Teaching-Learning Based Optimization with a bad neighborhood and shuffling complexes approach ($TLBO_{scebn}$)

4.5 General implementation aspects

4.5.1 Algorithm implementation in R

The algorithms were developed in the R language (R Core Team 2015) in R studio (RStudio Team 2015). They are all functions that can be called with the following input parameters: population size, complexes, dimension of problem, minimum boundary, maximum boundary, generation, reinitialization limit and a printing option (total population or only global best). R is usually considered a slow language, however it also allows relatively quick development and debugging. There's a certain understanding that for-loops, particularly over data frames, are somewhat slow. In this thesis, attempts were made to preallocate vectors and matrices, to vectorize operations and to use the apply family instead of for-loops where appropriate to speed up the computation. There was also an attempt to use byte-code compilation using the compiler package and the cmpfun() function. After algorithms performance were very slow, system.time() analysis was conducted of the algorithms with and without usage of the cmpfun() function. It turned out that the cmpfun() function slowed down the computation tremendously and was therefore not used. At this stage, the implemented algorithms are set-up to solve minimization problems only.

4.5.2 Random seed

To avoid resulting differences between the optimization algorithms based on random variation of the number generation, the random seed was set before the start of each algorithm run. A list of random integer seeds was generated on www.random.org (Haahr and Haahr 2017). RANDOM.ORG is a true random number service that generates randomness via atmospheric noise. www.random.org has been used in peer-reviewed research Biggar et al. (2008). This means that for the first benchmark run, the random seed was set to k. The same seed k was set to all algorithms for the first benchmark run before the algorithm was called. This way, it can be assumed that only the design of the algorithm influenced the differences between the algorithms for the first run and not random variation. This is further important as it assures reproducibility of the presented results.

To generate (pseudo-)random uniform numbers, R's default Marsenne-Twister algorithm, developed by Matsumoto and Nishimura (1998) for generating uniform pseudorandom numbers, was used.

4.5.3 Boundary

There are several ways how to deal with particles leaving the search space. Particles leaving the search space can remain at their previous location, or the particles remain at the boundary of the search space or the particles bounce or reflect back into the search space. It was decided

that particles should be reflected back into the search space. For a particle that has left the boundary, it is reflected back into the search space before evaluation of the objective function so that

$$x_{i,reflected} = boundary - (x_{i,outside} - boundary). \tag{4.24}$$

If the upper boundary is 5, and the particle's position is $x_{i,outside} = 7$, the $x_{i,reflected} = 5 - (7 - 5) = 3$.

4.5.4 Reinitializing the population

Reinitializing can be a powerful tool to increase population diversity and avoid premature convergence. A reinitialization probability was introduced that affects the worst half of the population. For PSO algorithms the reinitialization does not affect the personal best found so far as this information is stored independent of the current location. However, for $TLBO$ algorithm, reinitialization overwrites the affected worst half of the population.

The user of the algorithm can set a reinitialization limit, which is reduced every generation. At the end of each generation, a random variable a is generated, where $a \sim U(0, 1)$. If $a >$ reinitialization limit, reinitialization takes place and the reinitialization limit is set to its initial value. If the user wants to avoid reinitialization, the reinitialization limit can be set extremely high, so that the reinitialization value, despite reduction, will never decrease below 1, so that the if statement never becomes true. On the contrary, if the user wishes reinitialization to take place every iteration, the reinitialization limit can be set to 0. During testing of the algorithms with benchmark functions, the algorithms were calibrated using different reinitialization limits.

4.5.5 Stopping criteria

The stopping criteria are usually a mix of maximum allowed function evaluations or iterations and convergence criteria. If the allowed number of iterations is reached, the algorithm stops. The convergence criteria are manifold. We can stop the algorithm when an acceptable solution was found. This can be considered the maximum allowed error. When the solution has converged on the basis of lack of change of the global best solution we can also stop the algorithm. This does not necessarily mean the solution has converged to a global optimum. Convergence can be assumed when following condition holds (Engelbrecht 2005):

$$f(x_{t-1}) - f(x_t) < \epsilon(1 + |f(x_t)|). \tag{4.25}$$

Stopping criteria for multi-objective problems are more difficult. In this work, bi-objective

optimization was stopped after a set number of generations.

4.5.6 Restart

Especially with limited computer or server capacity, or when one cannot estimate how many function evaluations are required, it is useful to be able to restart the computation with partly optimized values. Being able to restart an optimization also allows tuning parameters in the process. This was highly utilized with bi-objective optimization.

5 Benchmark functions

5.1 Introduction

Benchmark functions and benchmark problems are widely used to compare algorithms. The no-free-lunch theorem is a mathematical theorem that was first formalized by Wolpert and Macready (1997). They state, "*that for any algorithm, any elevated performance over one class of problems is offset by performance over another class.*" An algorithm can therefore perform better over a set of objective functions or benchmark functions but not over all possible problems. This is independent of the chosen performance metric. A lot of optimization algorithm research is based on competitiveness and providing better solutions than other optimization algorithms. However, with the no-free-lunch theorem it becomes clear that algorithm A cannot be universally better than algorithm B. Nonetheless, algorithm A might perform better than algorithm B on a set of objective functions or a real-life problem.

Simon (2013) (Appendix B) contains a very detailed section on the no-free-lunch theorem and its implications. Included are also examples showing that the usage of different performance metrics can lead to contradicting conclusions making it important to show a range of performance matrices. Benchmark functions are useful to compare metaheuristic algorithms. A wide range of benchmark functions and problems exist. Benchmark functions can be of different dimensions, uni-modal or multimodal, static or dynamic, single or multi-objective. Many benchmark functions are biased towards certain types of search spaces and so are certain optimization algorithms. They may be biased towards the center or towards feasible regions laying parallel to each other. It is therefore important when testing benchmark functions to have an understanding of the behavior of the algorithms. Biased benchmark functions can be adapted to unbias the search space by using offset of independent variables and rotation matrices such as in the problem definition of the CEC 2013 special session on real-parameter optimization (Liang et al. 2013). Using offsets and rotation matrices can make the benchmark function more difficult to solve (Simon 2013). This can be an advantage when benchmark functions are used as proxies for selecting the best algorithm for a real-world problem. It can also be a disadvantage when reviewing the performance of different optimization algorithms from different authors as the same names are used for offset benchmark functions and standard benchmark functions. Furthermore, often success rates in respect to certain benchmark functions are reported that are based on relatively high acceptance values and should thus be treated with a critical eye. If the benchmark functions are used to select algorithms for real-world problems, it becomes evident that the benchmark function should at least in part represent certain aspects of the real-world problem. A detailed list of benchmark definitions, including a discussion, can be found in Simon (2013) (Appendix C).

For the global optimization algorithms, 10 benchmark functions were selected from the CEC 2013 Real Parameter Optimization Competition and evaluated in 10D and 30D. Furthermore, two Dual Richards' Unsaturated Equation Solver ($DRUtES$) benchmark functions were evaluated presented. The bi-objective Particle Swarm Optimization (PSO) algorithm was tested on the bi-objective benchmark function DTLZ2 and reached sufficient convergence (not shown), but was not tested further on other benchmark functions.

5.2 CEC 2013 real-parameter optimization benchmark functions

5.2.1 Description of CEC 2013 real-parameter optimization benchmark functions

The CEC 2013 Real Parameter Optimization Competition provides popular benchmark functions that are published on several platforms, including as an R package, which made them easily available for testing algorithms in the R environment. In this thesis, ten CEC benchmark functions were selected to cover a range of different search spaces. The mathematical description of the selected benchmark functions and their optimal value can be found in Tab. 3 and Tab. 4. All functions are shifted to shifting operator $o = [o_1, ..o_D]$. M1 and M2 are rotation matrices. Λ^α is a diagonal matrix in D dimensions with the iith diagonal element as $\lambda_{ii} = \alpha^{\frac{i-1}{2(D-1)}}, i = 1, .., D$. T^β_{asy} is a conditional operator: If $x_i > 0, x_i = x_i^{1+\beta\frac{i-1}{D-1}\sqrt{x_i}}$. T_{osz} is an oscillation operator. While working on the algorithms, the author made use of different dimensions provided in the R package. Presented are the results from 30D benchmark function runs, as these provide challenging complexity, and 10D benchmark function runs. 30D benchmark functions are often used to test optimization algorithms. The search range for all functions is $[100,-100]^D$. The function description are kept in accordance with the CEC 2013 real-parameter optimization competition explained in Liang et al. (2013). Each benchmark function run was repeated 30 times. The 30D benchmark functions were run with 8 different calibration sets, where the initial reinitialization probability was changed, resulting in $8x30x10 = 2400$ optimization runs. The 10D benchmark functions were run with 3 different calibration sets, where the initial reinitialization probability was changed, resulting in $8x30x3 = 720$ optimization runs. One of the calibration runs used a very high reinitialization limit, so that no reinitialization took place. This was done to create a reference run. The maximum function evaluation was kept in accordance with the competition guideline and set to $eval_{max} = D \cdot 10000$, but stopping criteria were also applied, leading to less function evaluation if a) the global best position has not changed significantly for 100 iterations (10000 function evaluations) or b) if the error criterion was met. The solution was found when the error value obtained was less than 10^{-8}. This acceptance value in combination with the shifting operator and transformation

matrices is rather strict. Each run was started with a swarm size of 100. The *sce* variants were split into two sub complexes. The PSO variants could have been split into more complexes, however, the design of the Teaching-Learning Based Optimization ($TLBO$) algorithms in completely vectorized form caused an extreme increase of computational time for smaller sub-swarm sizes. To maintain comparability, a low number of complexes with 50 particles each was chosen.

Table 3: Summary of benchmark functions used from CEC 2013 real-parameter optimization f1-f5.

ID	Name	Function	$f_i^* = f(x_{opt})$
f_1	Sphere Function	$f_1(x) = \sum_{i=1}^{D} z_i^2 + f_1^*, z = x - o$	-1400
f_2	Rotated High Conditioned Elliptic Function	$f_2(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} z_i^2 + f_2^*, z = T_{osz}(M_1(x - o))$	-1300
f_3	Rotated Rosenbrock's Function	$f_3(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1}) + (z_i - 1)^2) + f_3^*$ $z = M_1(\frac{2.048(x-o)}{100}) + 1$	-900
f_4	Rotated Ackley's Function	$f_4(x) = -20exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}) - exp(\frac{1}{D}\sum_{i_1}^{D} cos(2\pi z_i)) + 20 + e + f_4^*$ $z = \Lambda^{10} M_2 T_{asy}^{0.5}(M_1(x - o))$	-700
f_5	Rotated Weierstrauss Function	$f_5(x) = \sum_{i=1}^{D} (\sum_{k=0}^{20} [0.5^k cos(2\pi 3^k (z_i + 0.5))]) - D\sum_{k=0}^{20} [0.5^k cos(2\pi 3^k 0.5)] + f_5^*$ $z = \Lambda^{10} M_2 T_{asy}^{0.5}(M_1 \frac{0.5(x-o)}{100})$	-600

Table 4: Summary of benchmark functions used from CEC 2013 special session on real-parameter optimization f6-f10.

ID	Name	Function	$f_i^* = f(x_{opt})$
f_6	Rotated Griewank's Function	$f_6(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{z_i}{\sqrt{i}}) + f_6^*, z = \Lambda^{100}(M_1 \frac{600(x-o)}{100})$	-500
f_7	Rastrigin's Function	$f_7(x) = \sum_{i=1}^{D}(z_i^2 - 10cos(2\pi z_i) + 10) + f_7^*, z = \Lambda^{10} T_{asy}^{0.2}(T_{osz} \frac{5.12(x-o)}{100})$	-400
f_8	Rotated Rastrigin's Function	$f_8(x) = \sum_{i=1}^{D}(z_i^2 - 10cos(2\pi z_i) + 10) + f_8^*$ $z = M_1 \Lambda^{10} M_2 T_{asy}^{0.2}(T_{osz} M_1 \frac{5.12(x-o)}{100})$	-300
f_9	Schwefel's Function	$f_9(x) = 418.9829D - \sum_{i=1}^{D} g(z_i) + f_9^*$ $z = M_1 \Lambda^{10} \frac{1000(x-o)}{1000} + 4.209687462275036e + 002$	-100
f_{10}	Composition function (g_1=Rotated expanded Griewank+Rosenbrock, g_2=Rotated Schaffers, g_3=Rotated Schwefel, g_4=Rotated expanded Schaffers, g_5=f_1)	$f_{10}(x) = \sum_{i=1}^{n}(\omega_i^*[\lambda_i g_i(x) + bias_i]) + f_{10}^*$ $w_i = \frac{1}{\sqrt{\sum_{j=1}^{D}(x_j - o_{ij})^2}} exp(-\frac{\sum_{j=1}^{D}(x_j - o_{ij})^2}{2D\sigma_i^2}), \omega_i = \frac{w_i}{\sum_i^n w_i}$ $n = 5, \sigma = [10, 20, 30, 40, 50], \lambda = [2.5, 2.5e^{-3}, 2.5, 5e^{-4}, 0.1], bias = [0, 100, 200, 300, 400]$	1400

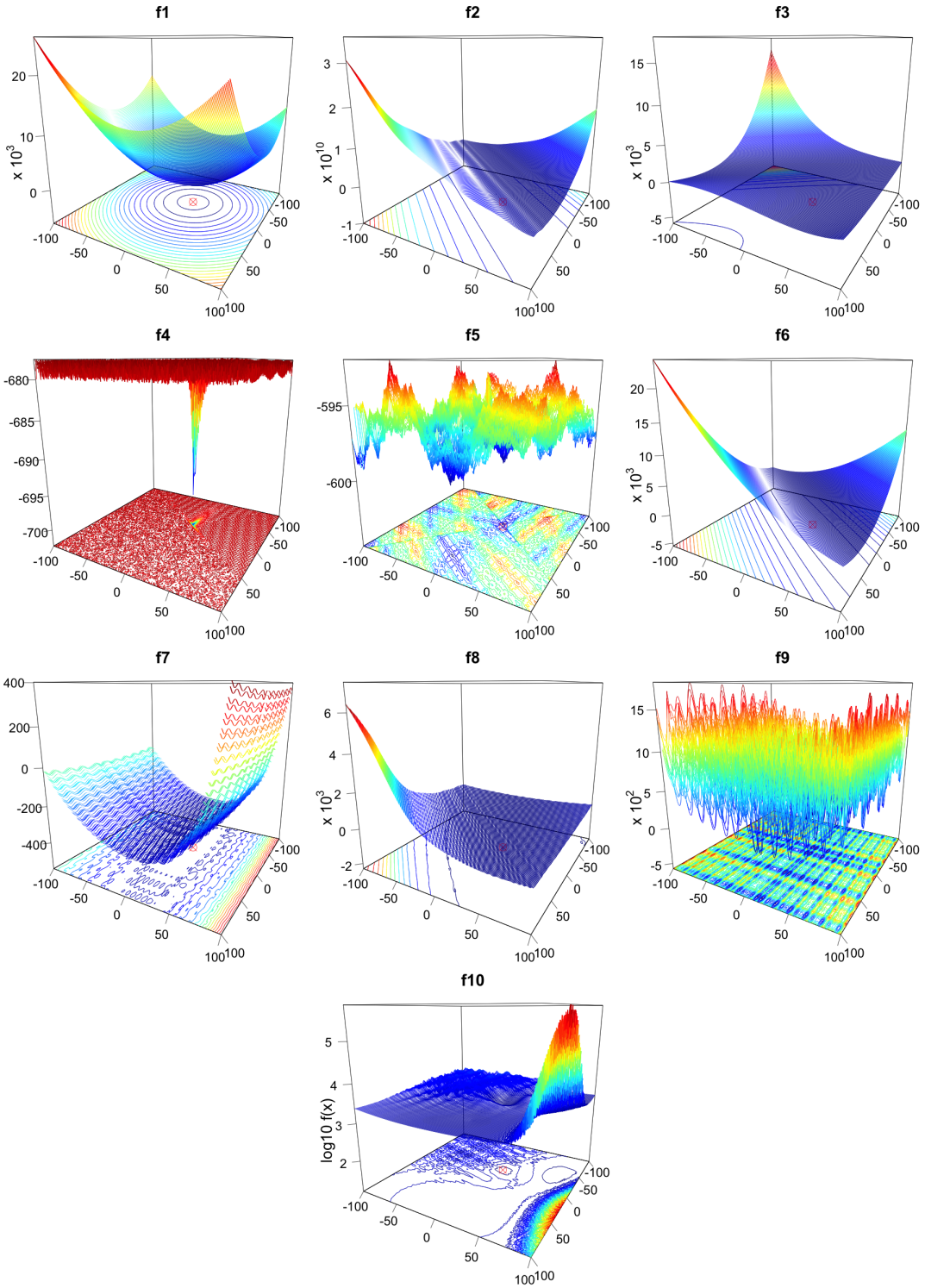


Figure 9: 3D maps of 2D benchmark functions

5.3 DRUtES benchmark functions

DRUtES benchmark functions were created to test the algorithms on the *DRUtES* environment. Different reference simulations were conducted and used to design simple inverse problems. It was tested how well the algorithms can find the parameters to lead to the known output. An inverse analysis can be described as a problem, where the output is known, i.e. a time series or measurements, so that

$$Output = Experiment(Input) \tag{5.1}$$

and using the inverse approach, the aim is to find the correct input

$$Input = Inverse(Output). \tag{5.2}$$

We obtain the correct input by minimizing an error function. The error function is the objective function, where we compare the output of our model with the estimated input parameters to the known reference output. A commonly used error function is the Root Mean Squared Error ($RMSE$). Inverse problems can be extremely challenging, when the model describing the input-output relation is not known or if the output is of very bad quality with high uncertainties.

The CEC benchmark functions have the same boundary in all dimensions, which is not the case for *DRUtES* benchmark functions. Therefore *DRUtES* benchmark functions are also important for testing the performance of algorithms on problems with different dimensional boundaries. Furthermore the boundaries of the dual permeability model benchmark function were set challenging with the boundary set to the real parameter, which makes the parameter difficult to identify. Additionally, we assume that we only know the water content data of the reference solution and that pressure head information is unknown. This was considered to improve the complexity of the created benchmark functions and make it more realistic, as the same conditions apply for the case study in this thesis. Overall, these benchmark functions are easier to solve than real problems as we assume perfect knowledge of initial and boundary conditions, and we know for certain that our artificial reference data can be described by the selected model. No noise was added to the reference simulation data, however one set of trial benchmark simulation runs were conducted with noise. All optimization algorithms performed equally well (not shown), which was assumed to be due to impossible parameter identification due to overambitious noise. Three *DRUtES* benchmark functions were created in 1D. The first and second benchmark functions were created using the standard total head model and the third benchmark function was created with the implemented dual permeability model.

5.3.1 Set-up of DRUtES benchmark functions

To execute the benchmark functions the algorithms had to be linked to *DRUtES* and a set-up had to be found to use several computer cores to evaluate several population members in parallel. Each algorithm was called from a set-up script feeding it population size, complexes, dimension, minimum and maximum boundary, maximum number of generations, reinitialization limit and number of available cores. The set-up also created numbered folders containing copies of *DRUtES* equal to the number of cores. When the algorithm evaluates the found parameters, R writes an input file *pars.in* with n parameter sets, where n denotes the number of cores. For the evaluation another R script is called. This was done to keep the algorithm itself as clean as possible and only link the evaluation script to the algorithm. The evaluation script makes a system call and executes a bash script. The bash script then reads the created file *pars.in* and uses the bash sed function to update a previously prepared configuration file with the new estimated parameters, before copying this new configuration file into the appropriate *drutes.conf* directory. *DRUtES* can then be executed. The bash script waits until all processes have finished. The R evaluation script then reads the *DRUtES* output file and compares it to reference data. The *DRUtES* benchmark functions were evaluated using the water content data only. The mean of the logged (base 10) $RMSE$ for selected observation points was chosen as the objective function criterion, which can be written as

$$logRMSE = \log \frac{\Sigma_{i=1}^{n}(Obs(i) - Sim(i))^2}{n}, \tag{5.3}$$

where Obs is the the observation at i, Sim is the simulation at i and n denotes the number of observations.

The result is then fed back to the algorithm to apply its strategy and start the next generation if the stopping criterion is not true. Each benchmark function was repeated 10 times for each algorithm per calibration run. Due to time limitations, only the more simple first benchmark function could be run with several different reinitialization probabilities, the second benchmark function was run with two different reinitialization probabilities including an extremely high limit and thus prohibiting any reinitialization. The dual permeability model benchmark was run with algorithm specific reinitialization limits that presented best guesses based on the 30D CEC benchmark function performance of each algorithm.

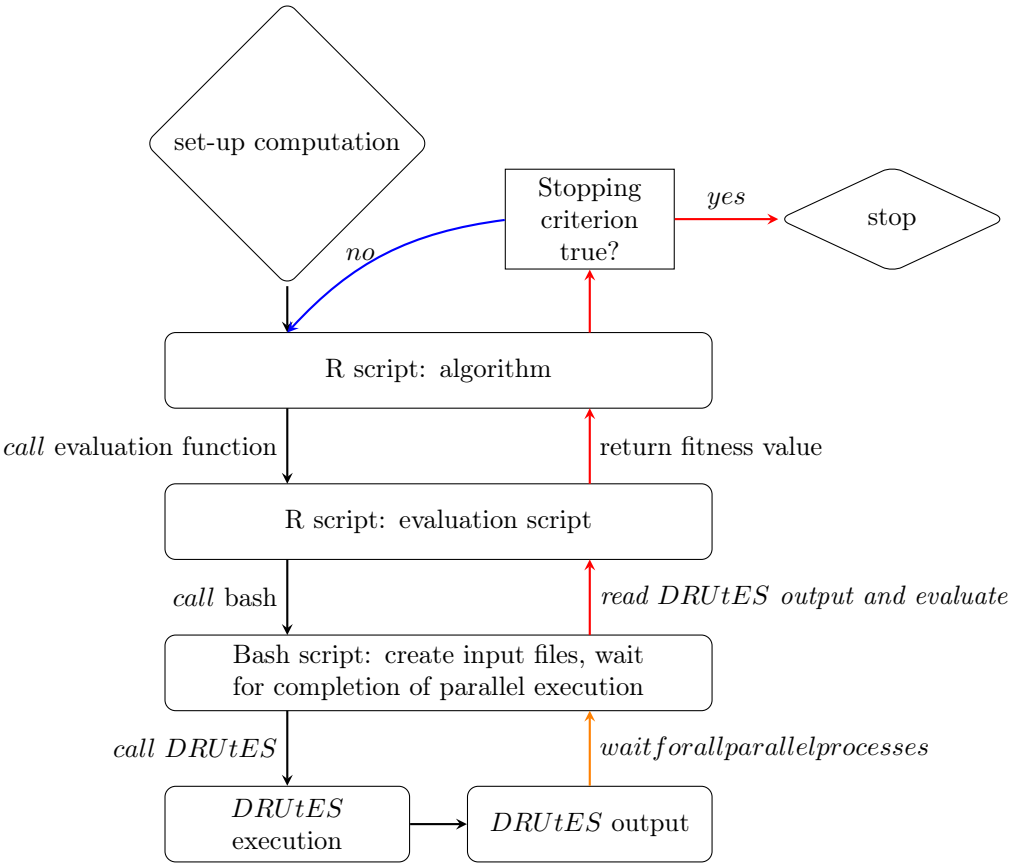


Figure 10: Simplified flow chart of the link between R scripts and *DRUtES*

Observation points were placed at a depth of 2.5, 5 and 7.5 cm. Only a small number of observation points were used to, again, create more realistic conditions. The Picard iteration criterion was set high as we were not interested in an accurate solution of the soil physics problem and because this generated output at exactly the same time steps. This assures that all parameter sets run quickly. Parameters θ_r and l were not optimized. m was calculated based on n. Other model set-ups are depicted in Tab. 5. For the 1D standard *DRUtES* benchmark function a scenario with a constant infiltration flux from (1) the bottom and (2) the top boundary with four unknowns was created, thereby creating low-dimensional 4-D benchmark problems. The maximum number of function evaluations was set to 32000. Tab. 6 shows the soil hydraulic properties and boundaries for the standard model benchmark functions. For the 1D dual permeability *DRUtES* benchmark function, a ponding scenario was chosen with an arithmetic mean as the exchange boundary conductivity in the coupling term (variant 1) and thirteen unknowns was created, which made it a medium-dimensional 13-D benchmark problem. However, for a 1D soil physics problem this is already a large number of parameters. The maximum number of function evaluations was set to 48000. Tab. 7 shows the soil hydraulic properties and boundaries for the dual permeability model benchmark function.

Table 5: Initial and boundary conditions used for *DRUtES* benchmark functions.

Parameter	**Symbol**		**Standard**		**Dual**	
Domain length	L		10 cm			
Spatial discretization	dx		0.1 cm			
Simulation time	t		2 days		0.5 days	
Minimum time step	dt_{min}		1e-7 days			
Maximum time step	dt_{max}		5e-3 days			
Initial pressure head	h_{ini}		-100 cm		-50 cm	
			top	bottom	top	bottom
Boundary condition		1	-100 cm	0.5 cm d^{-1}	0 cm	free drainage
		2	0.5 cm d^{-1}	-100 cm		

Table 6: Soil hydraulic properties used to generate reference data with standard model and minimum and maximum boundary values.

Parameter	**Symbol**	**Value**	**Min**	**Max**
Inverse of air entry value [cm^{-1}]	α	0.1	0.05	0.15
Shape parameter	n	2	1.5	2.5
Sat. water content [-]	θ_s	0.45	0.3	0.7
Residual water content [-]	θ_r	0	-	-
Sat. hydraulic conductivity [cm d^{-1}]	K_s	200	10	500

Table 7: Soil hydraulic properties used to generate reference data with dual permeability model and minimum and maximum boundary values.

Domain	**Parameter**	**Symbol**	**Value**	**Min**	**Max**
Matrix	inverse of air entry value [cm^{-1}]	α_m	0.005	0.001	0.01
	shape parameter [-]	n_m	1.5	1.3	1.8
	sat. water content [-]	$\theta_{s,m}$	0.5	0.4	0.5
	res. water content [-]	$\theta_{r,m}$	0.105	-	-
	sat. hydraulic conductivity [cm d^{-1}]	$K_{s,m}$	1.05	0.1	2
	matrix weight [-]	w_m [a]	0.5	-	-
Fracture	inverse of air entry value [cm^{-1}]	α_f	0.1	0.05	0.15
	shape parameter [-]	n_f	2.5	2	3
	sat. water content [-]	$\theta_{s,f}$	0.5	0.3	0.7
	res. water content [-]	$\theta_{r,f}$	0	-	-
	sat. hydraulic conductivity [cm d^{-1}]	$K_{s,f}$	600	500	800
	fracture weight [-]	w_f	0.5	0.1	0.5
Exchange boundary	distance to center of aggregate [cm]	a	1.0	0.4	0.6
	material parameter [-]	β	3	-	-
	material parameter [-]	γ	0.4	-	-
	sat. hydraulic conductivity [cm d^{-1}]	K_a	0.01	0.001	0.1
	inverse of air entry value [cm^{-1}]	$alpha_{ex}$	0.005	0.001	0.01
	shape parameter [-]	n_ex	1.5	1.3	1.8

[a] w_m was calculated based on w_f estimation

5.4 Benchmark results and discussion

For statistical analysis, the paired non-parametric Wilcoxon test was used to compare algorithms. It is analogous to the paired student T-test, but does not require the data to be normally distributed. It tests if the difference of the median between two algorithms is zero and thereby aims to detect significant differences between two algorithms. The R function wilcox.test() with the options paired=TRUE and alternative="two.sided" was used for the compute the paired two-sided Wilcoxon test. For this, the best algorithm based on the median value (indicated as best) was compared to all other algorithms. This was considered preferable to the overall lowest value, as the lowest value was often an outlier and the best median was hoped to represent a more robust solution. Algorithms which are not significantly different to the lowest median (at level p-value=0.05 and α=0.025 for two-sided test) are indicated in bold. The results were also evaluated graphically using box plots of the best calibration run and using convergence plots of the best run.

5.4.1 Results of CEC 2013 benchmark functions

Table 8 shows the best result of each algorithm over 30 simulation runs for each 10D benchmark function. Table 9 shows the Wilcoxon test of each 10D benchmark function. Figure 11 shows convergence plots of the best algorithm run and Fig. 12 shows boxplots. For 30D benchmark functions, table 10 shows the best result for each algorithm over 30 simulation runs, and table 11 shows the result of the Wilcoxon test for each 30D benchmark. The best values in Tab. 8 and 10 are also indicated in bold. For this, 8 decimal places were considered, but only 2 decimal places were printed in the tables and therefore not all of the same values are printed in bold. Figure 13 shows convergence plots of the best algorithm run and Fig. 14 shows boxplots of the best calibration. The optimum was only found for two 10D benchmark functions, f_1 and f_3, and two 30D benchmark functions, f_1 and f_6. The 10D benchmark functions were not calibrated as thoroughly and would probably perform better with ideal reinitialization limits.

As the CEC benchmark functions were used, it is difficult to compare the results with other results found in the literature. The aim of the thesis was also to compare the algorithms with common modifications to new variants. Therefore, we can consider the PSO and $TLBO$ variant as our base line. 10D CEC benchmark functions were used by Maca and Pech (2015) and the results are overall similar. For proper comparison, more data would need to be available to conduct statistical analysis. Simon (2013) recommends the use of statistical F-tests to test, whether differences between results are due to different random number generations. Each benchmark repetition was seeded with the same integer for all algorithms, which made this problem negligible. To be clear, different seeds were used for every repetition. The $TLBO$ variants significantly outperformed the PSO variants for

benchmark function f_2 in 10D, however the opposite effect is visible in 30D. The Teaching-Learning Based Optimization with a bad neighborhood and shuffling complexes approach ($TLBO_{scebn}$) algorithm was the overall most successful algorithm, particularly on 30D functions. The $TLBO$ variants showed overall faster convergence in most benchmark functions (not shown).

Table 8: Best result of all algorithms over 30 trial runs for 10D benchmark function.

ID	f(x_{opt})	PSO	PSObn	PSOsce	PSOscebn	TLBO	TLBObn	TLBOsce	TLBOscebn
f_1	-1400.00	**-1400.00**	**-1400.00**	**-1400.00**	**-1400.00**	-1400.00	**-1400.00**	**-1400.00**	**-1400.00**
f_2	-1300.00	2363.95	1829.05	-645.45	5115.35	-1101.02	-1299.80	**-1299.97**	-578.37
f_3	-900.00	**-900.00**	-899.99	**-900.00**	-899.98	-900.00	**-900.00**	-900.00	**-900.00**
f_4	-700.00	-679.78	-679.81	-679.72	**-679.85**	-679.82	-679.80	-679.77	-679.71
f_5	-600.00	-596.68	-595.21	-597.02	-595.98	-597.15	-596.93	-596.39	**-598.28**
f_6	-500.00	-499.77	-499.77	-499.85	-499.87	**-499.95**	-499.91	-499.91	-499.92
f_7	-400.00	-392.04	-380.10	-391.66	-390.05	-393.02	**-393.04**	-392.03	-390.05
f_8	-300.00	-290.99	-279.11	-289.91	-288.06	-293.98	-291.04	**-295.03**	-290.05
f_9	-100.00	28.58	351.60	**-81.43**	36.98	265.61	169.71	223.83	348.17
f_{10}	1400.00	1500.00	1500.00	**1500.00**	1500.30	1500.00	1500.00	1500.00	**1500.00**

Table 9: 10D benchmark functions: Non-parametric Wilcoxon test with best algorithm (based on best median value).

ID	PSO	PSObn	PSOsce	PSOscebn	TLBO	TLBObn	TLBOsce	TLBOscebn
f_1	6.91E-07	2.19E-03	2.61E-08	**4.78E-01**	1.86E-09	9.31E-09	2.05E-07	**best**
f_2	5.59E-09	6.15E-08	5.14E-06	1.86E-09	**7.32E-02**	**best**	**3.09E-01**	1.72E-03
f_3	3.84E-02	**7.67E-02**	9.93E-03	2.08E-02	4.73E-02	**best**	**7.61E-01**	**5.29E-01**
f_4	7.11E-03	**1.77E-01**	9.93E-03	1.89E-04	**4.16E-01**	**2.80E-01**	**best**	**4.77E-01**
f_5	2.69E-05	2.61E-08	2.99E-03	3.13E-04	5.01E-03	4.66E-03	9.98E-07	**best**
f_6	9.98E-07	9.31E-09	6.91E-07	1.30E-08	3.45E-05	7.61E-03	3.74E-03	**best**
f_7	**best**	3.54E-08	8.71E-03	7.06E-05	1.45E-02	**6.99E-02**	**1.77E-01**	2.62E-02
f_8	**8.55E-01**	2.35E-06	**2.45E-01**	1.97E-02	**7.15E-01**	**9.35E-01**	**best**	**7.77E-01**
f_9	**best**	6.15E-08	**9.19E-01**	**7.32E-02**	9.31E-09	1.30E-07	2.76E-06	1.68E-06
f_{10}	1.72E-03	2.99E-03	1.70E-04	3.45E-02	**7.15E-01**	2.62E-02	2.77E-02	**best**

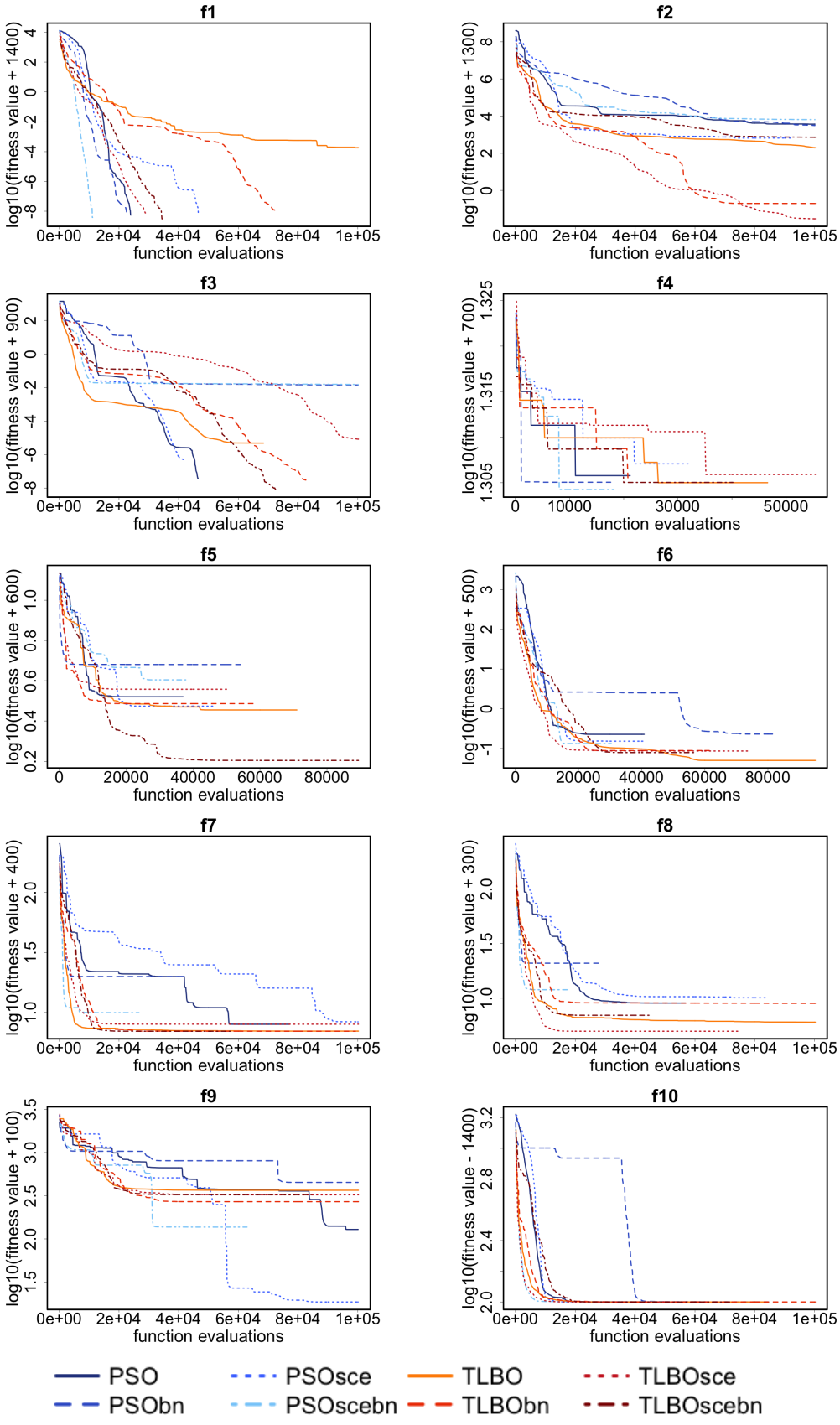


Figure 11: Convergence plots of the best run of each algorithm for 10D benchmark functions.

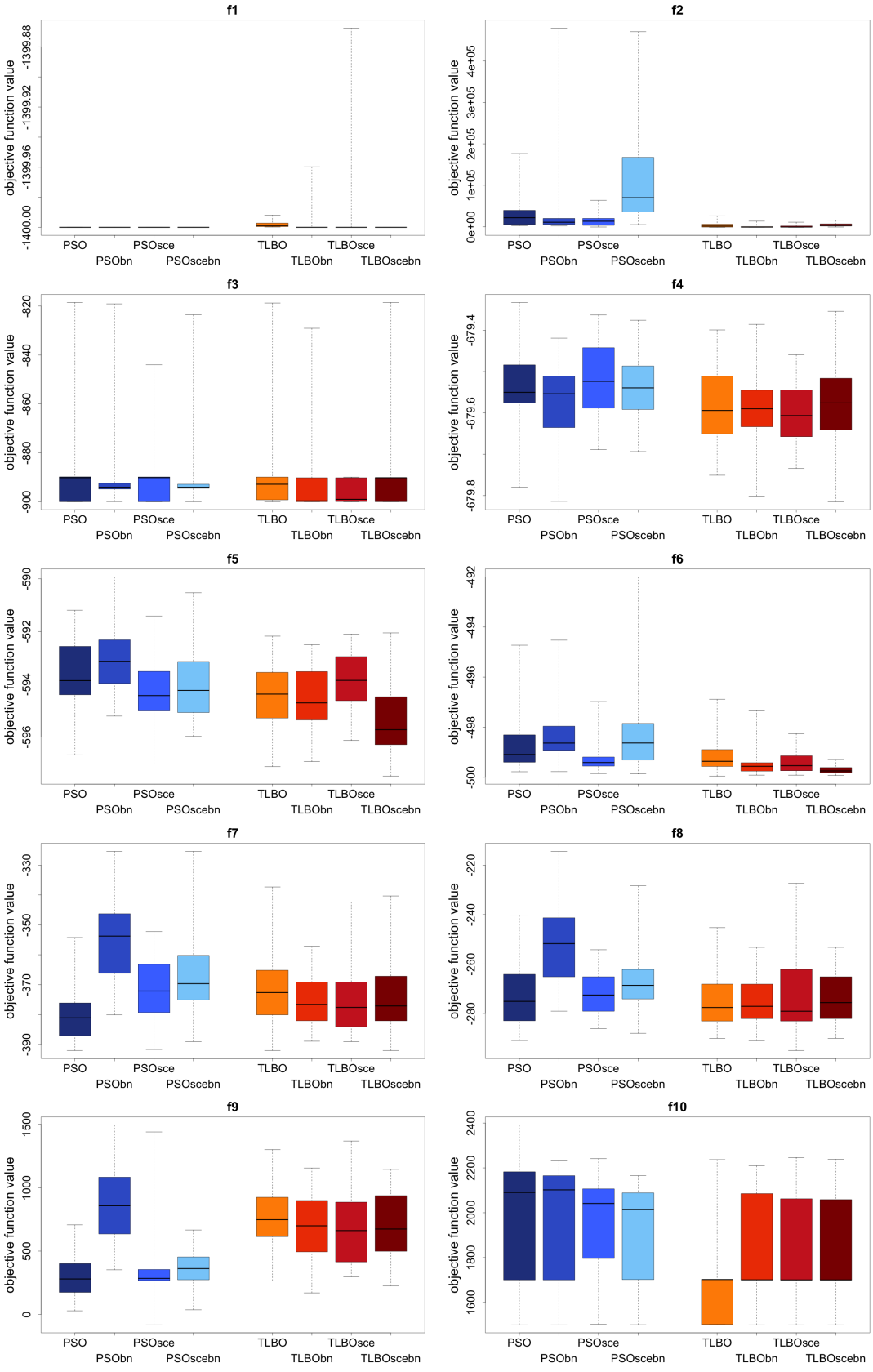


Figure 12: Boxplots of results over 30 trial runs for 10D benchmark functions.

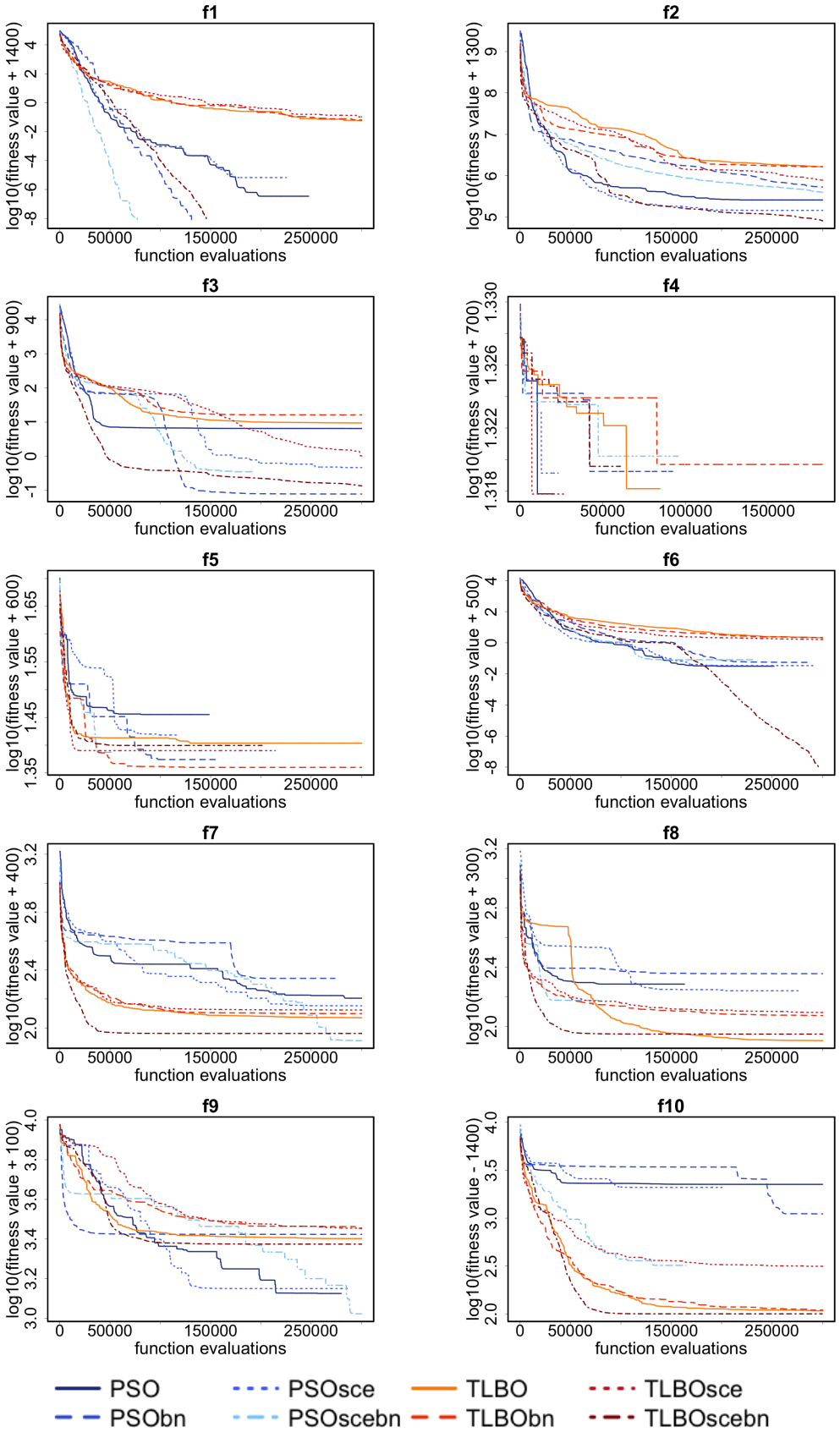


Figure 13: Convergence plots of the best run of each algorithm for 30D benchmark functions.

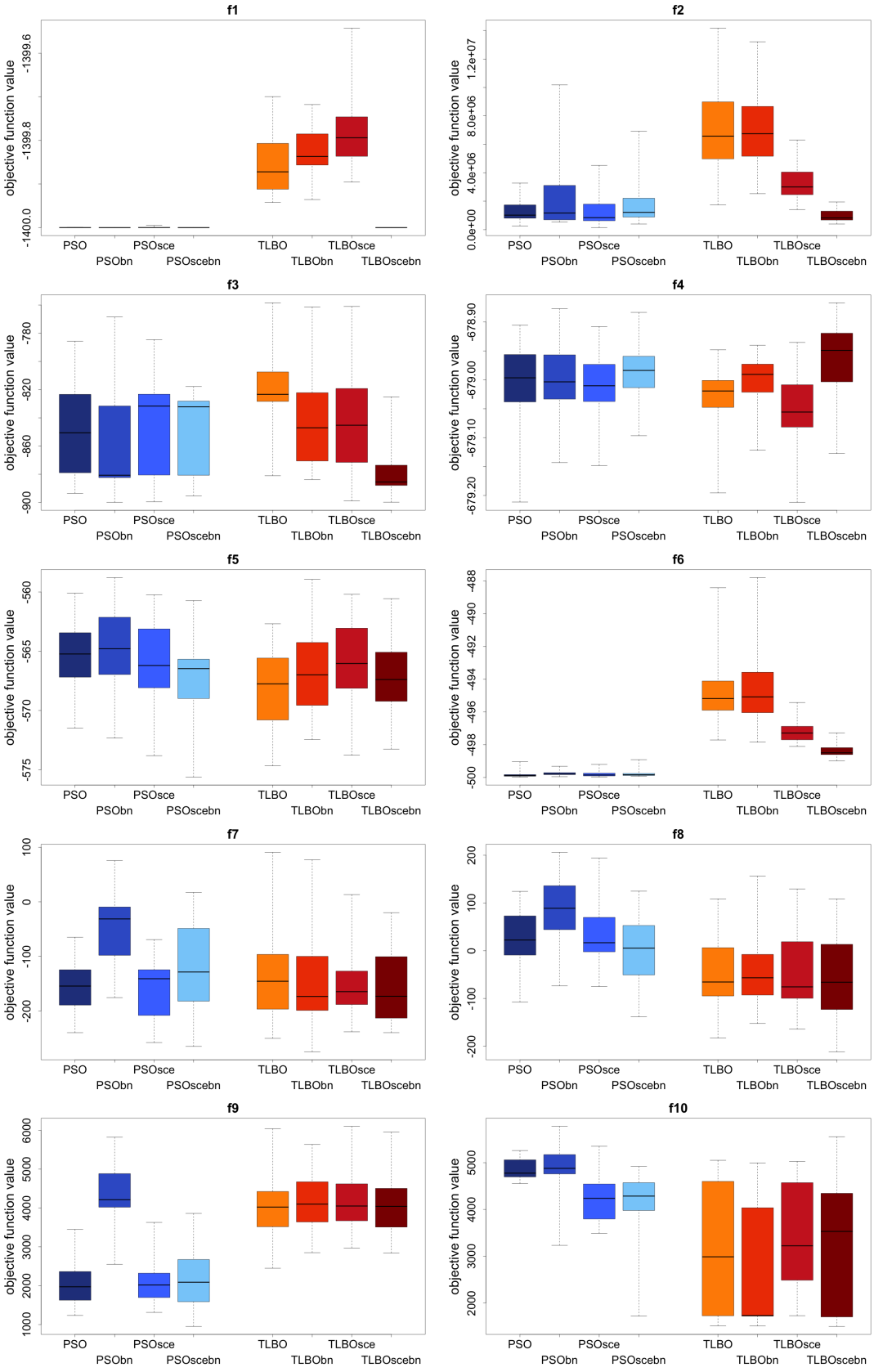


Figure 14: Boxplots of results over 30 trial runs for 30D benchmark functions.

Table 10: Best result of all algorithms over 30 trial runs for 30D benchmark function.

ID	f(x_{opt})	PSO	PSObn	PSOsce	PSOscebn	TLBO	TLBObn	TLBOsce	TLBOscebn
f_1	-1400.00	-1400.00	**-1400.00**	-1400.00	**-1400.00**	-1399.94	-1399.94	-1399.90	-1400.00
f_2	-1300.00	258826	534223	144859	396932	1650464	1659264	782187	**80531**
f_3	-900.00	-898.55	**-899.92**	-899.54	-899.65	-890.59	-883.80	-898.99	-891.89
f_4	-700.00	-679.21	-679.14	-679.15	-679.07	-679.20	-679.11	**-679.21**	**-679.13**
f_5	-600.00	-576.53	-576.33	-573.80	-575.61	-574.67	**-577.09**	-575.44	-574.91
f_6	-500.00	-499.97	-499.94	-499.96	-499.96	-497.89	-497.84	-498.37	**-500.00**
f_7	-400.00	-239.63	-180.11	-257.71	**-318.35**	-282.52	-274.55	-267.11	-308.46
f_8	-300.00	-107.41	-55.24	-126.30	-150.14	**-219.98**	-181.82	-176.17	-211.45
f_9	-100.00	**1233.22**	2550.82	1311.74	952.34	2423.44	2732.81	2748.53	2265.59
f_{10}	1400.00	3650.09	2505.11	3489.86	1720.62	1507.30	1509.45	1713.81	**1501.63**

Table 11: 30D benchmark functions: Non-parametric Wilcoxon test with best algorithm (based on best median value).

ID	PSO	PSObn	PSOsce	PSOscebn	TLBO	TLBObn	TLBOsce	TLBOscebn
f_1	1.86E-09	**1.31E-01**	1.86E-09	**best**	1.86E-09	1.86E-09	1.86E-09	**5.45E-02**
f_2	**6.67E-02**	8.71E-03	**2.29E-01**	1.34E-03	3.73E-09	1.86E-09	1.86E-09	**best**
f_3	1.42E-06	**6.06E-02**	2.32E-04	2.32E-04	9.31E-09	1.60E-05	7.99E-06	**best**
f_4	2.37E-03	2.02E-03	4.03E-03	3.80E-04	**1.84E-01**	2.02E-03	**best**	1.82E-05
f_5	3.22E-03	4.03E-03	4.27E-02	**3.49E-01**	**best**	**2.45E-01**	2.08E-02	**4.52E-01**
f_6	**best**	8.72E-04	**3.39E-01**	**7.32E-02**	1.86E-09	1.86E-09	1.86E-09	1.86E-09
f_7	**8.71E-01**	1.82E-05	**8.87E-01**	**2.05E-01**	**5.84E-01**	**best**	**9.84E-01**	**7.77E-01**
f_8	2.83E-04	6.91E-07	1.60E-05	9.93E-03	**9.52E-01**	**7.15E-01**	**best**	1.00E+00
f_9	**best**	3.73E-09	**9.03E-01**	**5.43E-01**	3.73E-09	1.86E-09	1.86E-09	5.59E-09
f_{10}	5.59E-09	1.86E-08	3.79E-06	6.29E-05	2.62E-02	**best**	2.48E-02	**1.35E-01**

5.4.2 Results of DRUtES benchmark functions

Fig. 15 shows boxplots of the best median calibration run and 16 shows convergence plots of the best run of each benchmark function with each algorithm. For the first benchmark function it may seem as if the $TLBO$ algorithm performed best and is robust, but it performed much worse in other calibration runs (not shown), whereas the Teaching-Learning Based Optimization with shuffling complexes approach ($TLBO_{sce}$) algorithm was overall much more robust and reached the acceptable error in every calibration run for at least some of the repetitions. Comparing the boxplots in Fig. 15, for the well calibrated first benchmark the algorithms showed much better performance than for benchmark function 2 and 3. This can also indicate that the correct reinitialization is very important for these problem.

Table 12: *DRUtES* benchmark functions: Non-parametric Wilcoxon test with best algorithm (based on best median value).

ID	**PSO**	**PSObn**	**PSOsce**	**PSOscebn**	**TLBO**	**TLBObn**	**TLBOsce**	**TLBOscebn**
standard 1	3.91E-03	1.95E-03	3.91E-03	1.95E-03	**7.70E-01**	1.95E-03	**best**	1.95E-03
standard 2	**2.03E-01**	**6.45E-02**	**1.93E-01**	1.95E-03	**3.75E-01**	3.91E-03	**best**	3.91E-03
dual	3.91E-03	3.91E-03	**7.42E-02**	**7.42E-02**	3.91E-03	3.91E-03	3.91E-03	**best**

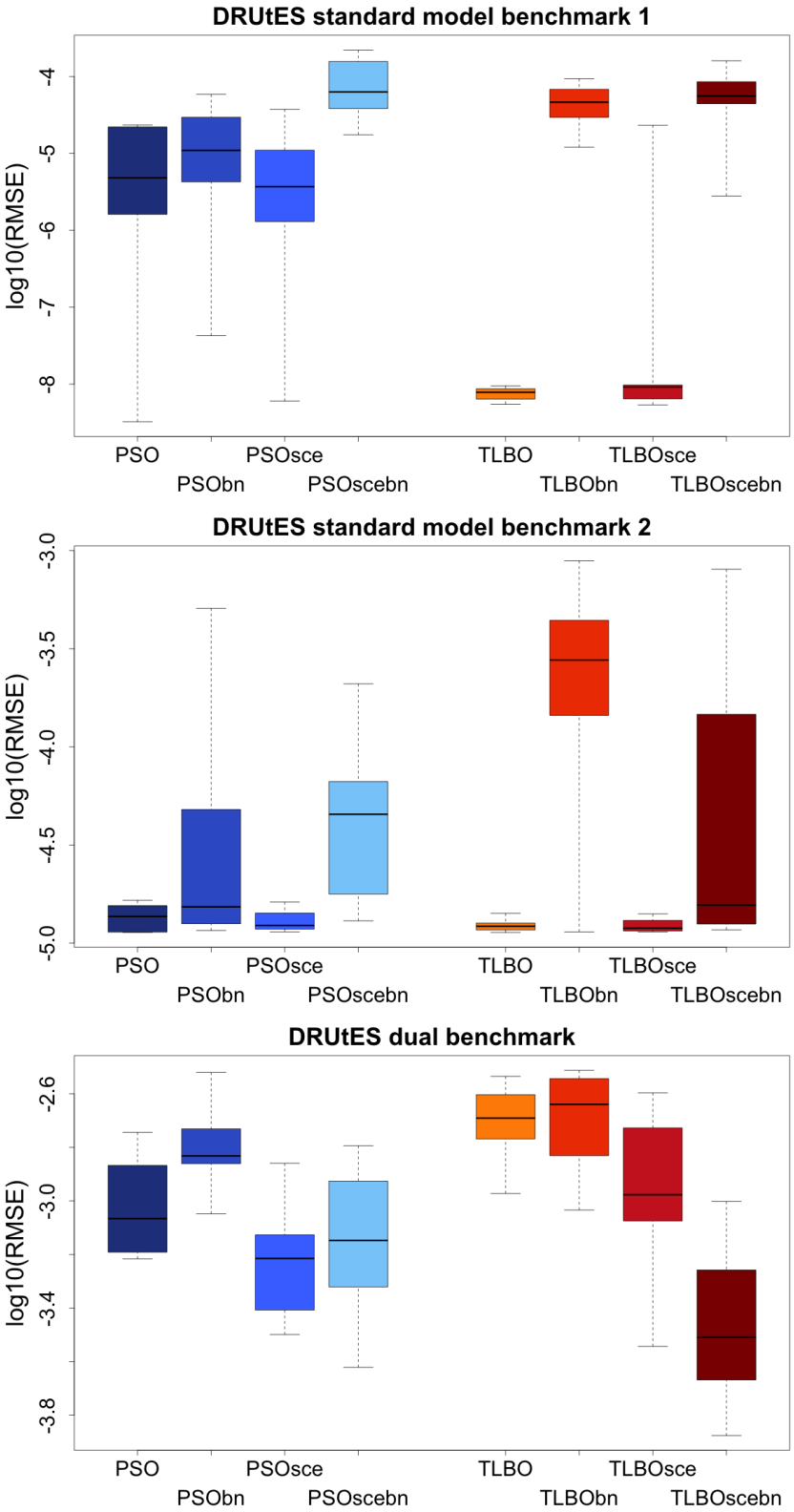


Figure 15: Boxplots of the best run of the *DRUtES* benchmark functions.

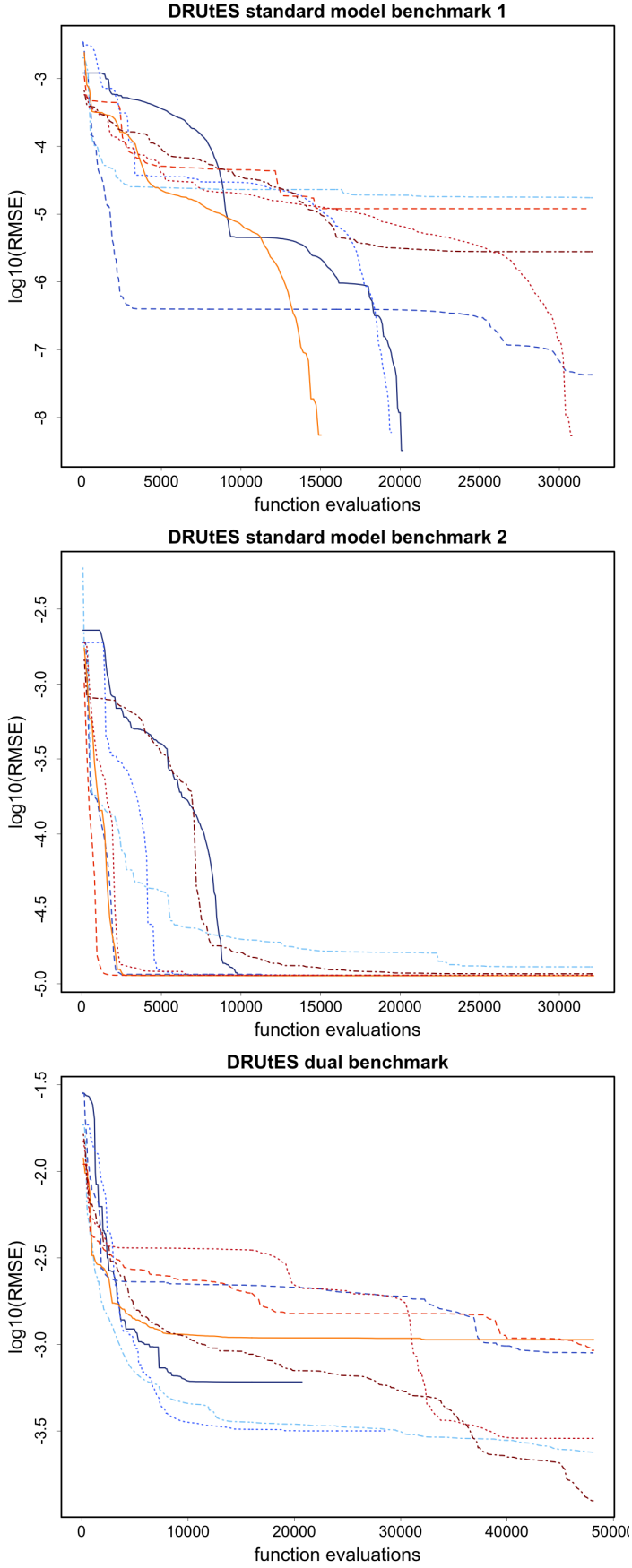


Figure 16: Convergence plot of the best run of *DRUtES* benchmark functions.

5.4.3 Benchmark summary

Tab. 13 translates the results into a scoreboard to gain an overview of the general performance of each algorithm. An algorithm gets a point for each benchmark that was not significantly different to the best performing algorithm in respect to the median. It becomes evident that for different dimensional problems and benchmark functions the algorithms' performances can differ significantly. Furthermore the performance of the algorithms on the CEC benchmark functions is not necessarily a reflection of the performance of the algorithms on the $DRUtES$ benchmark functions. It is therefore important to test algorithms on the $DRUtES$ environment to deduct which algorithm should be chosen for the actual inverse modeling task, even though the $DRUtES$ benchmark functions could be improved to include a greater variety of soil textures and uncertainty of reference data, initial and boundary conditions. Based on the results, it would be recommendable to use $TLBO_{sce}$ for low dimensional problems, but not for high dimensional problems. Throughout the calibration and algorithm testing, it became evident that reinitialization can be very useful in enhancing the performance. Fig. 27 (appendix B) shows boxplots of 30D CEC benchmark function without any reinitialization resulting in an overall worse performance. Overall, the $TLBO$ algorithms required more frequent reinitialization than PSO algorithms, except for $TLBO_{scebn}$, where it was problem dependent. This can be explained in the convergence structure of each class of algorithms. PSO algorithms permit particles to roam around in the search space, especially bad solutions, whereas $TLBO$ algorithms converge quickly towards the global best and only improved solutions are stored storing only improved solutions and discarding bad solutions. This presents the need for diversity enhancement, which is given by frequent reinitialization.

Table 13: Scoreboard of best algorithms that did not show significant differences in the best median.

Benchmark	**PSO**	**PSObn**	**PSOsce**	**PSOscebn**	**TLBO**	**TLBObn**	**TLBOsce**	**TLBOscebn**
10D f_1	0	0	0	1	0	0	0	1
10D f_2	0	0	0	0	1	1	1	0
10D f_3	0	1	0	0	0	1	1	1
10D f_4	0	1	0	1	1	1	1	1
10D f_5	0	0	0	0	0	0	0	1
10D f_6	0	0	0	0	0	0	0	1
10D f_7	1	0	0	0	0	1	1	0
10D f_8	1	0	1	0	1	1	1	1
10D f_9	1	0	1	1	0	0	0	0
10D f_{10}	0	0	0	0	1	0	0	1
Σ	3	2	2	3	4	5	5	7
30D f_1	0	0	0	1	0	0	0	1
30D f_2	1	0	1	0	0	0	0	1
30D f_3	0	1	0	0	0	0	0	1
30D f_4	0	0	0	0	1	0	1	0
30D f_5	0	0	0	1	1	1	0	1
30D f_6	1	0	1	1	0	0	0	0
30D f_7	1	0	1	1	1	1	1	1
30D f_8	0	0	0	0	1	1	1	1
30D f_9	1	0	1	1	0	0	0	0
30D f_{10}	0	0	0	0	0	1	0	1
Σ	4	1	4	5	4	4	3	7
DRUtES standard 1	0	0	0	0	1	0	1	0
DRUtES standard 2	1	1	1	0	1	0	1	0
DRUtES dual	0	0	1	1	0	0	0	1

6 Case study

6.1 Introduction

Data from three water content AQUA-TEL-TDR (Automata, Inc., now McCrometer CONNECT) sensors were used in this study. They were installed horizontally at depth of 10, 20 and 30 cm. The field soil was loamy Chernozem on a carbonate-rich loess substrate. Sensor installation, data sampling and Time Domain Reflectometer (TDR) data calibration were not part of this thesis and were provided from previous field research conducted in Prague-Suchdol (50°8′N, 14°23′E, 286 m a.s.l). A detailed description of the site, field measurements and TDR calibration can be found in Doležal et al. (2012) and Doležal et al. (2015). The TDR readings resulted in anomalously high water content readings during rainfall events. The effects were very pronounced in the upper TDR readings. During the installation of the TDR probe, a slurry was prepared with local native soil. This slurry was applied around the TDR probes in order to close the gaps between the TDR probes and the soil. When the slurry dried and shrank, cracks appear around the TDR probes. As a result, Doležal et al. (2012, 2015) and Kogelbauer et al. (2015) hypothesized that a very thin layer of extremely high porosity, an artificial macroporous envelope, surrounded the TDR probe (Fig. 17). It was concluded that the high TDR readings during the passing of the wetting front were caused by local soil hydraulic properties. The steep increase was assumed to be due to preferential flow. There were previous attempts to simulate and quantify percolation fluxes with the measured data with two different manually calibrated models with moderate success. A more detailed description of previous simulations can be found in Kogelbauer et al. (2015). Furthermore, Kogelbauer et al. (2015) describe that during dry spells, cracks at the surface 1 to 3 mm wide appear. The preferential flow was possible due to a connected network of macropores that exists in the upper soil layer, which was also confirmed by visual inspection during the uninstallation of the TDR sensors (Doležal et al. 2012).

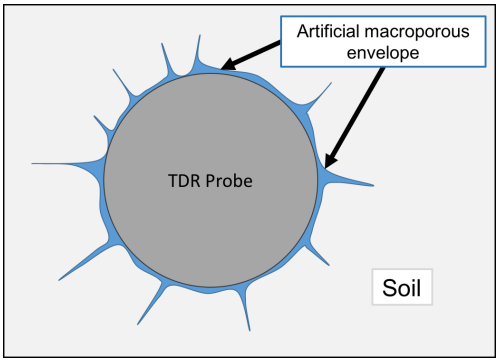


Figure 17: Conceptual model of the artificial macroporous envelope causing anomalous TDR readings during rainfall.

6.2 Mesh optimization

The presented problem requires a 2D description, which has to represent processes of different spatial magnitude. The overall domain can be represented by a soil bock of 75 cm width and 100 cm depth. The heterogeneities around the TDR are only assumed to be 2.5 mm wide and are assumed to require additional soil hydraulic properties (SHP) description. The mesh therefore has to be of quite fine resolution close to the TDR. The finer the mesh, the more unknowns have to be solved numerically, which increases the computational time. It is therefore desirable to find a mesh that reduces the number of degrees if freedom, while maintaining high accuracy. This optimization problem is somewhat contradictory as higher mesh resolution will lead to higher accuracy and coarser mesh resolution will lead to a smaller number of degrees of freedom and less unknowns to be solved numerically. Therefore a set of optimal solution sunder constraint conditions is desired, which can be obtained using multi-objective optimization. The modified dynamic neighborhood Multi-objective Particle Swarm Optimization ($MOPSO$) algorithm was used to solve this multi-objective problem.

6.2.1 Set-up of mesh optimization

First, a very fine reference mesh was created using Gmsh (Geuzaine and Remacle 2009) covering the main properties of the problem. Only the upper 35 cm were of interest, which was reflected by the non-uniform discretization of the created mesh. Using the standard total head model, a simple infiltration simulation was constructed (Tab. 14). The great advantage of Gmsh is the ability to create a mesh with parameter inputs from a configuration file (mesh.geo), which can be compiled to create a mesh file (mesh.msh). The configuration file contains contains geometry information and initial mesh size information, but also information on mesh progressions along transfinite lines, which can overwrite the initial mesh size information. With a transfinite line, the number of nodal points along a line can be defined, and with a progression parameter the distribution of these lines can be defined, where 1 represents a uniform distribution. Basic uniform meshes can thus be easily refined where necessary and complex meshes can be generated. The geometry as depicted in Fig. 18, meaning the size of the soil box, the position and size of the TDR probes, anomalies and helper lines, was considered static.

Seven parameters were identified defining the mesh. The optimization task thus became a 7D problem. Fig. 18 shows the schematic of the mesh creation. The domain box was defined through 6 points (1-6). The TDRs (gray) and the anomaly borders (red) were defined by 4 arches creating circles, which were constructed through 4 points. To make sure the same number of elements surround each anomaly, helper circles (orange) were constructed close to the anomaly.

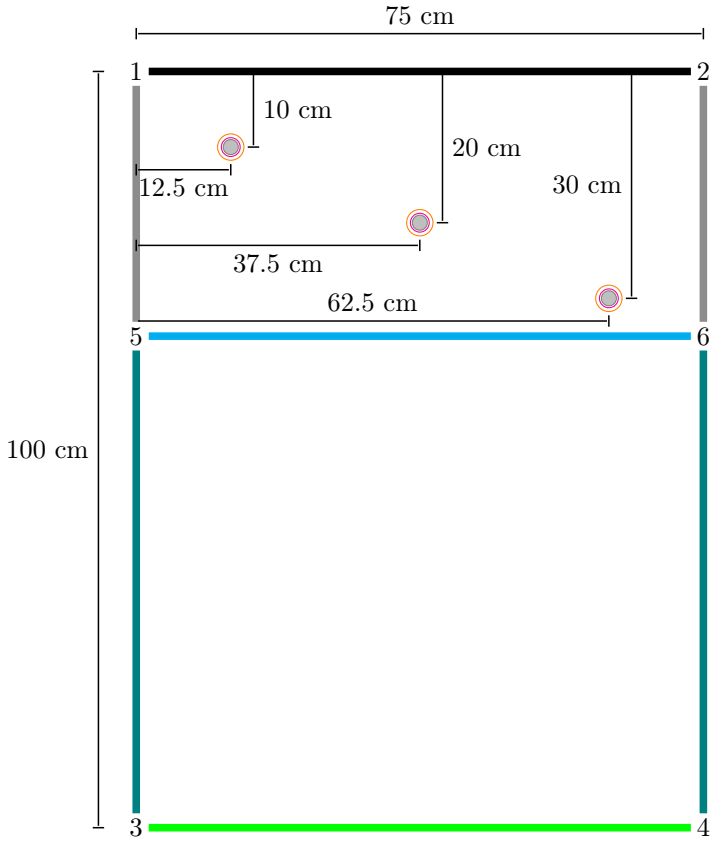
Each point in Gmsh is defined by a coordinate and a mesh size. Point 1, Point 2, TDRs,

anomaly borders and their helpers were assigned a parameter *mesh size 1* with their point definition. For TDRs, anomalies and helpers, this definition was overwritten using transfinite lines creating three further parameters *TDR*, *anomaly* and *helper* to refine the mesh and to specify the number of nodal points in each arch. Points 3 and 4 were assigned parameter *mesh size 2* and points 5 and 6 were assigned parameter *mesh size 3*. The top line indicated in black was also overwritten using transfinite line definition creating the last parameter *top*.

These definitions were sufficient to create a sufficiently complex mesh. The effects of the parameters can be described as follows. Lines connecting two points are automatically assigned the points' mesh size definition as long as no other mesh definition overwrites this statement. If the two points have different definitions, the mesh size along the line is linearly increasing or decreasing. The green line therefore automatically gets assigned *mesh size 3*, which is quite coarse. The cyan line connecting point 5 and 6 automatically gets assigned *mesh size 2*, which is smaller than *mesh size 3*. The actual element size of the mesh between the light blue line and green line is linearly increasing towards the bottom. The top transfinite line also decreases the size of mesh elements near the top. The element size decreases towards the helper circles and more so towards the anomaly. The mesh is finest in the area defining the anomalies.

The computational set-up is similar to the Dual Richards' Unsaturated Equation Solver ($DRUtES$) benchmark function depicted in Fig. 10, except that in this case, $Gmsh$ configuration files were created and new meshes were computed based on the optimization output. The optimization was constrained by a time limit that was set upon the $DRUtES$ execution. Only meshes needing less than 5 minutes for the given simulation were considered further. This was realized with a punishment function, which created artificially high Root Mean Squared Error ($RMSE$) values for mesh parameter sets taking longer. The mesh optimization was started on two computers, each containing 8 cores, to test the influence of population size. One mesh optimization was started with a population size of 48 and another with a population size of 80. The accuracy of each simulation was computed using the $RMSE$ of the water content data of the simulation output and the reference simulation. For this, three observation points directly above the TDRs were chosen. The degrees of freedom were estimated using the number of nodes created in each $Gmsh$ mesh. The personal best information was saved after each generation, which made it possible to follow the optimization output constantly and to restart optimization when necessary. The optimization would only stop if the maximum number of generations was reached. As it was unknown how many generations would suffice, the number of generations was set quite low. The result of the last generation was then used to restart the next run.

Geometry description of case study problem for mesh optimization



TDR and surrounding anomaly boundary and meshing helper circle

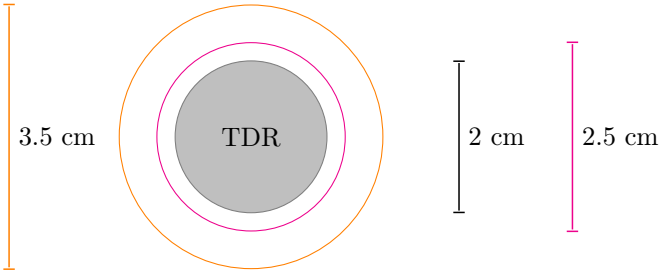


Figure 18: Schematic domain geometry set-up for Gmsh mesh optimization (top) and additional close-up of TDR probe (bottom) with surrounding anomaly (magenta) and meshing helper circle (orange).

Table 14: Initial and boundary conditions used in mesh optimization scenario.

Model	Sim. time	Min dt	Max dt	Initial h_{pres}	Top bc	Bottom bc	Sides bc	Picard criterion
standard	100 d	0.5 d	0.5 d	-100 cm	30 cm d^{-1}	-100 cm	0 cm d^{-1}	1e10

6.2.2 Results of mesh optimization

After a number of restarts, it became apparent that the optimization run with a population of 80 coverged much more slowly, despite significantly greater number of function evaluations (not shown). Although a greater population assures that a greater area of the search space

Table 15: Number of maximum generations set for each restart.

Run	Initial	1	2	3-5	6-13	14-17	18-23	24-25	Σ gen.	Σ eval.
48 particles	100	200	50	50	25	15	25	50	1010	48480
80 particles	100	200	50	28 [a]	-	-	-	-	378	30240

[a] Restart 3 only

is covered initially, it takes much longer to transfer information to distant neighborhoods. This is due to the low number of neighbors in a neighborhood. The implemented switch in the objective function order may actually help counteract this affect as the basis for the neighborhood formation changes. The mesh optimization was discontinued after two and a half weeks due to time limitations of this project. This was considered acceptable as only more potential meshes joined the Pareto front, but the Pareto front itself did not seem to move. Fig. 19 depicts the evolution of the mesh optimization. Note, comparing the result after 800 and 1010 generations may seem like a deterioration at first sight, but the effect is false as simply more members of the population joined the suitable search space. In the beginning, first none and then only a small number of particles have reached the non-punishable search space. As the optimization commences, more particles find mesh solutions within the time constraints and join the Pareto front. The Pareto front itself appears to be non-convex, although as the entire population has not fully converged this cannot be said for certain. It may seem as if the particles seem to converge towards attractors. Interestingly, the mesh set-up was unintentionally set-up so that different combinations of the chosen design parameters can in fact result in meshes with a very similar number of nodes and similar accuracy. This may in part be due to the chosen observation points. Mentioning this, it should be stressed that this work was not conducted to find the single-best mesh or to claim the existence of such, but to find an appropriate mesh that has the ability to reduce the computational time while maintaining high accuracy, which certainly allows the existence of different combination of design parameters to create similar meshes.

With the Pareto front a set of optimal solutions was found. It is now up to the scientist or engineer to choose the most suitable solution. In this case, 4 meshes were selected. Namely, the meshes with the highest (id 40) and the second highest accuracy (id 47), a mesh from the middle (id 38) and the mesh with the lowest number of nodes (id 43). These were run again to obtain time series, as only parameter sets are stored during the mesh optimization, but not the actual output. The time series were compared to the reference mesh (Fig. 20). Simple paired T-test and Wilcoxon test were conducted to see if the differences between the selected meshes and the reference mesh were statistically significant. Only the first observation point presented significant differences between the meshes (not shown). At this stage, the scientist or engineer has to make a decision on which objective function outweighs the other and it

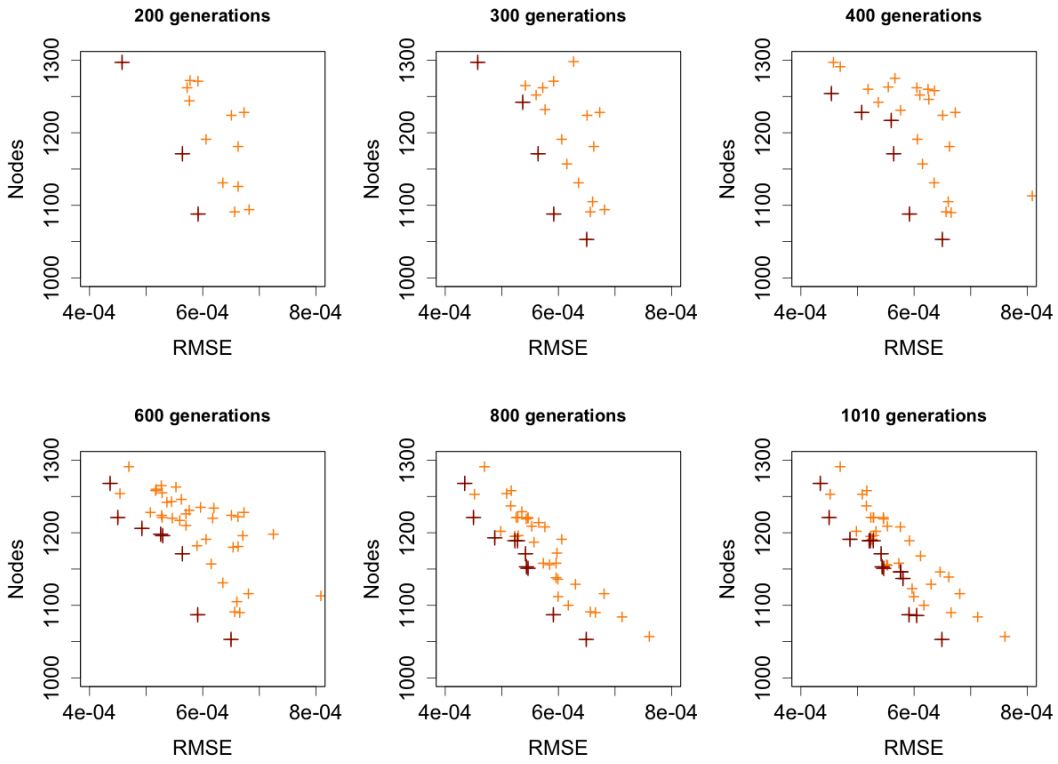


Figure 19: Evolution of mesh optimization after 200, 300, 400, 600, 800 and 1010 generations. The larger dark red crosses indicate the non-dominated Pareto front.

may also be perfectly valid to conclude that a decrease in computational time outweighs small differences. Nonetheless, it was decided to use the mesh with the highest accuracy (Fig. 21). The parameters identified for the mesh considered most appropriate can be found in Tab. 16

Table 16: Design variables for mesh optimization, assigned boundaries and identified value.

Parameter	**Symbol**	**Kind**	**Min**	**Max**	**Identified value**
Mesh size 1	$ms1$	size	5	10	6.02
Mesh size 2	$ms2$	size	5	10	7.05
Mesh size 3	$ms3$	size	15	20	16.42
Helper	$help$	number of nodes[a]	6	12	6.613
Anomaly boundary	$anom$	number of nodes[a]	10	20	10.60
TDR boundary	TDR	number of nodes[a]	10	15	11.37
Top boundary	top	number of nodes	20	50	27.03
RMSE	-	-	-		4.3e-4
Number of nodes	-	-	-		1268

[a] per quarter circle arch. Number of nodes for entire circle is four times greater.

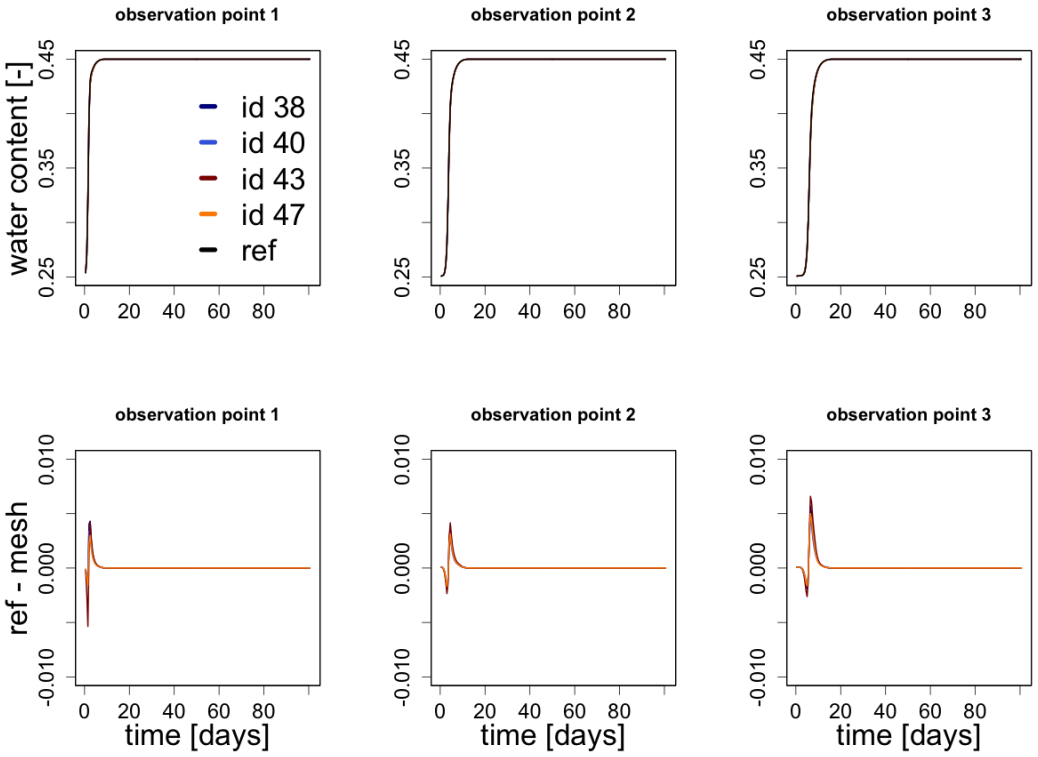


Figure 20: Comparison of four selected meshes from the Pareto front showing the time series (top) and the difference to the reference mesh (bottom). Mesh id 40 (light blue) resulted in highest accuracy, mesh id 47 (orange) resulted in second highest accuracy, mesh id 38 (dark blue) represents the middle with medium accuracy and number of nodes and mesh id 43 (dark red) with lowest number of nodes and worst accuracy.

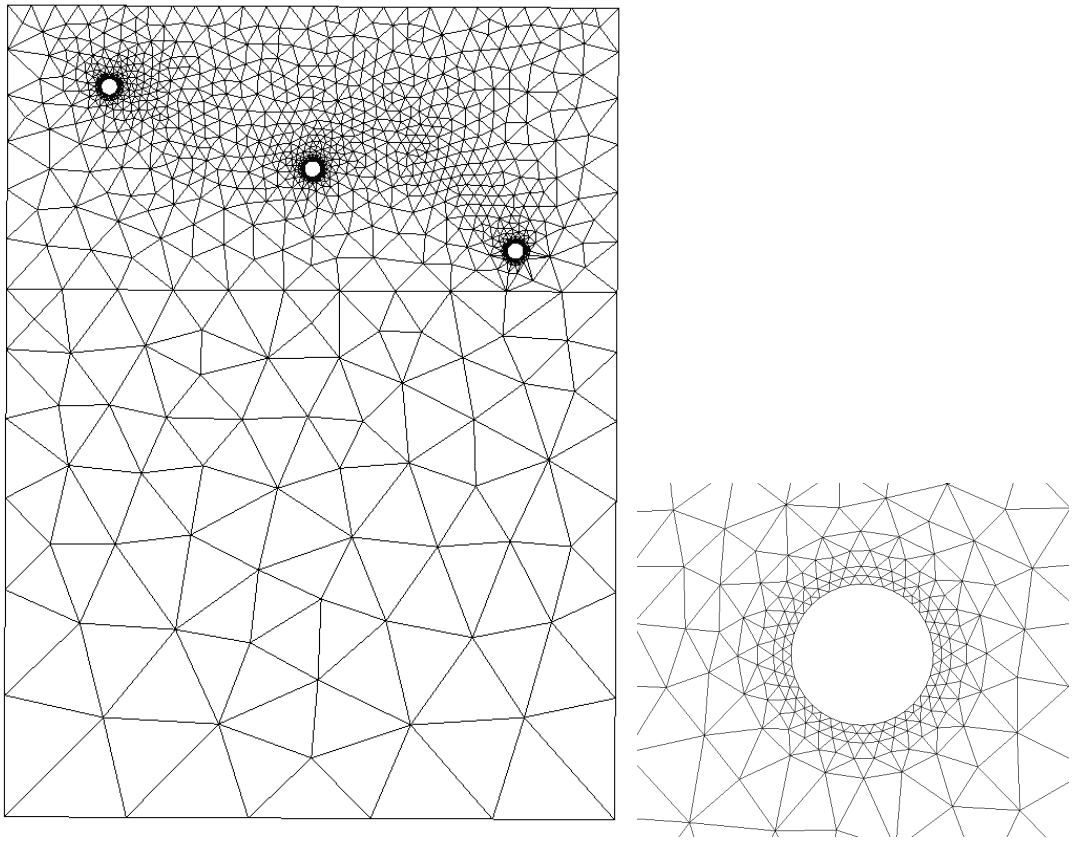


Figure 21: Optimized mesh and close-up of mesh around TDR

6.3 Inverse modelling

6.3.1 Project design and data preparation

Estimating initial condition

The calibration data made available covered rainfall events from the 16th until the 21st November 2010 and contained precipitation data and water content TDR measurements. Later, data over two years were made available that were used to update assumptions on the presented problem. One major unknown was the initial condition. To obtain an acceptable initial condition, it was decided to use the meteorological data available from the weather station on the grounds of the Czech University of Life Sciences, Prague-Suchdol, to perform an initialization run starting two weeks prior to the actual start of the calibration time period. As the anomalous heterogeneities were considered negligible to obtain initial pressure head values, the initialization run was set-up in 1D. The final vertical pressure head distribution was mapped on to the 2D domain and used as the initial condition. This caused the initial condition and the pressure head range to be different for every parameter set.

Data preparation for atmospheric boundary

As the initialization run was of considerable length, evaporation could not be neglected. The meteorological data from the weather station was also used to calculate hourly potential evaporation fluxes using the Penman's Monteith equation according to Allen et al. (1998), with assumptions concerning the soil heat flux and resistance terms recommended by the Food and Agriculture Organization (FAO) for hourly estimates. The Penman-Monteith form of the evaporation flux ET is

$$ET = \frac{\Delta(R_n - G) + \rho_a c_p \frac{e_s - e_a}{r_a} fac}{(\Delta + \gamma(1 + \frac{r_s}{r_a}))\lambda}, \tag{6.1}$$

where R_n is net radiation [MJ h^{-1} m^{-2}], G is soil heat flux [MJ h^{-1} m^{-2}], ρ_a is the mean air density [kg m^{-3}], c_p is specific heat of the air 1.013e-3 kJ kg^{-1} K^{-1}, Δ is the slope of the saturation vapor pressure temperature relationship[kPa K^{-1}], γ is the psychometric constant [kPa K^{-1}], $e_s - e_a$ is the vapor pressure deficit [kPa], r_a is the aerodynamic resistance [s m^{-}] approximated by $\frac{208}{u_2}$ with u_2 as the wind speed at two meters height, r_s is the bulk surface resistance assumed to be 70 s m^{-1}, fac is a conversion factor to obtain the correct time units and λ is latent heat 2.45 MJ kg^{-1}. The unit used in this simulation were cm for length and hours for time. To convert the wind dependent part of the ET flux to the desired time units fac was set to 3600 s h^{-1}. Obviously, if the time unit conversion is not accounted for, the wind effect is hugely underestimated. With the above equation this results in m^{-2} h^{-1} kg. Dividing by the liquid water's density ($\sim$ 1000 kg m^{-3}) and multiplying by 1000 converts

length units from m to mm, and results inresults in more commonly used units for the ET flux, despite not having stated this explicitly in the equation. The computed ET flux was divided by 10 to obtain cm h^{-1}. According to Allen et al. 1998, for hourly values the soil heat flux G can be approximated using R_n data with

$$G = \begin{cases} 0.1 \cdot R_n, & during \quad daytime \\ 0.5 \cdot R_n, & during \quad nighttime \end{cases} \tag{6.2}$$

where daytime was assumed to last from 7 am to 6 pm in October and from 8 am to 5 pm in November. For all other equations concerning the ET flux I refer to Allen et al. 1998.

Project design

When modeling infiltration, there is a risk to create an ill-posed or even singular problem when only assigning Neumann conditions. The inverse problem can become ill-posed when the soil becomes near-saturated resulting in the retention capacity becoming extremely small and possibly vanishing. This causes the system to lose connection to the initial condition defined at t=0. When the retention capacity term is zero, the system becomes singular and no exact solution can be found. When the connection to the initial condition is lost and only Neumann conditions are assigned, the system is only defined by first derivatives and therefore not fully defined. To avoid this, a Dirichlet condition, which assigns the solution to a boundary, was assigned to the bottom boundary. As the bottom boundary was located relatively far away from the area of interest, it was hoped this wouldto not falsify the solution significantly. To obtain the best possible Dirichlet condition for the 2D simulation, the 1D domain of the initialization was extended to cover 200 cm, which was initialized hydrostatically with $H_{init} = 0$ cm resulting in a linear pressure head gradient $h_{top} = -200$ cm at the top and $h_{bottom} = 0$ cm.

The inverse modeling was conducted using both the standard uni-modal van Genuchten model and the dual permeability model. This was done to see if the dual permeability model with additional parameters improves the standard model's parameterization of the problem. This resulted in 2 inverse problems of different dimensionality. The Teaching-Learning Based Optimization with a bad neighborhood and shuffling complexes approach ($TLBO_{scebn}$) algorithm was selected for both optimization tasks with a population of 64 and 2 complexes. Despite $TLBO_{scebn}$ being a global optimizer, population-based metaheuristics offer evaluation of and information from the entire population. $TLBO_{scebn}$ compared to Particle Swarm Optimization (PSO) is quite strong at getting the entire population of potential solutions into feasible space within a few iterations. The total number of iterations was set to 1000, but the state of the optimization was saved after each iteration and the approximated simulated water content time series was saved after each

function evaluation.

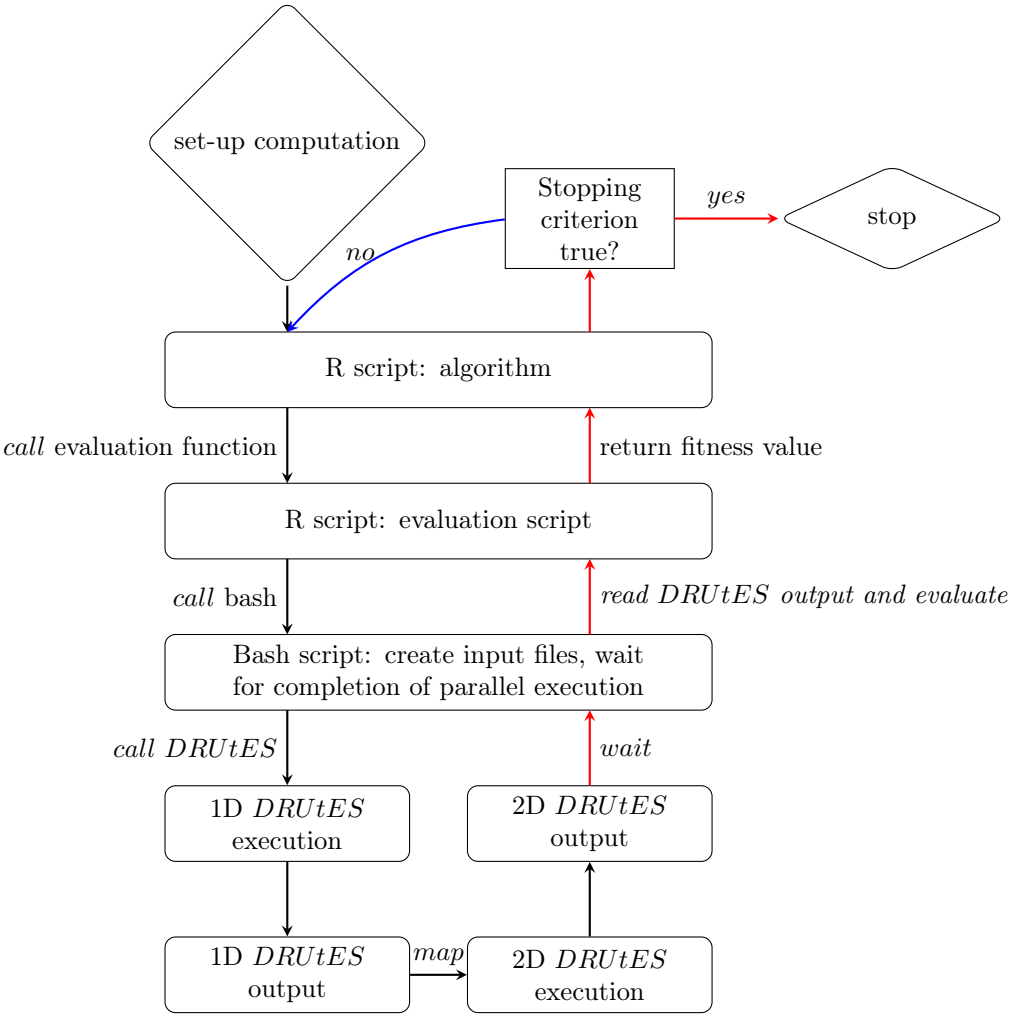


Figure 22: Simplified flow chart of the link between R scripts and *DRUtES* for inverse modeling.

Domain description and number of estimated soil materials

Fig. 23 shows the domain setup with boundary conditions and assumed soil material distribution. Soil 1, covering the top 15 cm, was considered to contain a dual pore system and therefore has the potential to create bypass or preferential flow. Soil 2, covering the rest of the soil, was described as compacted and originally only covered a depth of 15 to 25 cm. Based on the calibrated TDR data, it was assumed that the compaction affects the third TDR as well and that it is therefore acceptable to extend the soil definition. No dual pore system is assumed for soil 2. Soil 3, covering a thin layer of 2.5 mm surrounding the TDR, was assumed to be described by macropores with extremely high porosity. After preliminary runs and access to more data, it was concluded that two descriptions of the anomalous regions are necessary, namely anomaly 1 and anomaly 2. If only one anomalous region is defined, the mean water content of TDR 1 is underestimated with correct estimation of the

water contents of TDR 2 and TDR 3, or alternatively the mean water contents of TDR 2 and TDR 3 are overestimated while the mean water content of TDR 1 is correctly estimated. Therefore, one set of parameters was estimated to describe the upper TDR anomalous region surrounding the TDR at -10 cm depth (anomaly 1) and another set of of soil hydraulic parameters were estimated to describe the TDR anomalous regions at -20 and -30 cm depth (anomaly 2). The fracture and matrix definitions were identical for soil 2 and the anomaly regions, respectively, with K_a set relatively high in these uni-modal regions to achieve instant equilibrium in the dual permeability run in these soils. Note, the exchange boundary conductivity in soil 1 was set low, so that non-equilibrium caused by the preferential flow in the upper soil layer is still present.

Table 17: Computational set-up of 1D and 2D runs.

Parameter		**Dual**		**Standard**	
		1D	2D	1D	2D
Domain	Ω	100 cm	75 cm x 100 cm	100 cm	75 cm x 100 cm
Discretization	dx	1.5 cm	variable	0.5 cm	variable
Initial time step	dt	1e-5 h			
Minimum time step	dt_{min}	1e-7h		1e-5 h	
Maximum time step	dt_{max}	0.1 h	0.15 h	0.05 h	0.02 h
Simulation time	t	326 h	26 h	710 h	26 h

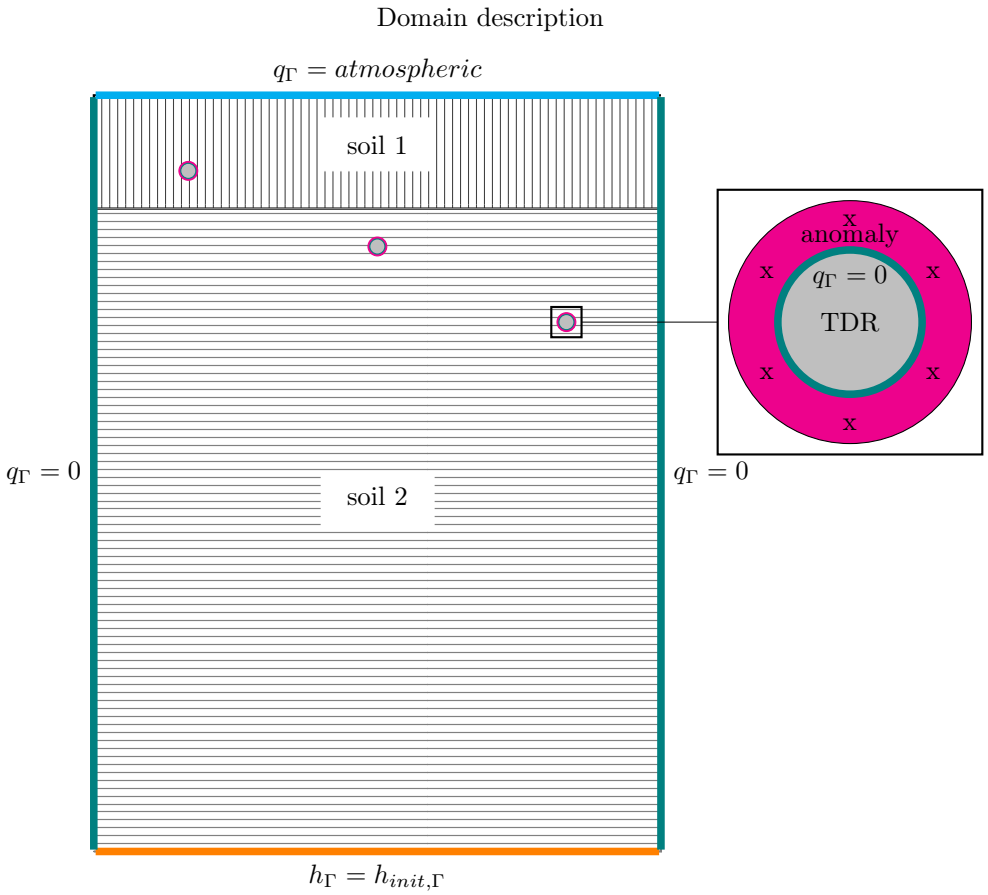


Figure 23: Boundary conditions and material distribution for inverse modeling. The x's in the anomaly surrounding the TDR probes indicate 6 observation points, which were placed in the middle of the anomalous soil shown in magenta. The anomalous soil was magnified in the close-up and is not true in scale.

Table 18: X,Z coordinates [cm] of TDRs and observation points in each anomaly.

	10 cm depth	**20 cm depth**	**30 cm depth**
TDR	12.500, -10.000	37.500, -20.000	62.500, -30.000
Observation point 1	12.500, -8.875	37.500, -18.875	62.500, -28.875
Observation point 2	12.500, -11.125	37.500, -21.125	62.500, -31.125
Observation point 3	11.530, -10.563	36.530, -20.563	61.530, -30.563
Observation point 4	11.530, -9.438	36.530, -19.438	61.530, -29.438
Observation point 5	13.470, -10.563	38.470, -20.563	63.470, -30.563
Observation point 6	13.470, -9.438	38.470, -19.438	63.470, -29.438

Finding search space ranges

Certain assumptions concerning the range of the estimated regions were partly based on Kogelbauer et al. (2015) soil description and the TDR data.

1. The saturated hydraulic conductivity was reported to vary over several orders of magnitude between 6x10^{-4} and 0.4 cm min^{-1}, where the lower end seems extremely slow and the higher end seems extremely fast for a loamy soil. The lower end would be more representative of a clay soil and the higher end is more representative of a sand. The soil was, however, reported to be highly heterogeneous.

2. The mean value of the of the soil's porosity is $\epsilon = 0.457$ cm^3 cm^{-3}. This was assumed to be a good approximation of the saturated water content θ_s for both soil 1 and soil 2.

3. The calibrated TDR data time series were used to estimate the saturated water content θ_s of the anomalies and to identify *knees* ,in order to estimate the water content at field capacity. To avoid making grave assumptions on whether field capacity is more representative at pF 1.8 or 2.5, the range of parameters was chosen to be flexible enough to allow for both. The saturated water content was estimated to be $\theta_{s,anomaly1}$=0.7 [-] with water content at field capacity (FC) at approximately $\theta_{anomaly1}(FC)$=0.48 [-] for anomaly 1 and saturated water content $\theta_{s,anomaly2}$=0.5 [-] for anomaly 2.

4. The fine earth of the soil contains 22-32.5% sand, 39.5-54% silt and 22-28% clay.

To estimate ranges, it is essential to understand the physical meaning of the retention curve. Hillel (2004) explains that if a slight suction is applied to water in a saturated soil, no outflow may occur until a critical value is exceeded, at which point the largest surface pore begins to empty and its water content is displaced by air corresponding to the air-entry value of a soil. We assume capacity retention function is proportional to the particle size distribution. Coarser soils tend to be more uniformly distributed with greater pores. This causes the air entry point to be at low suction and the coarser soils to exhibit steeper retention properties, where a significant proportion of the water drains suddenly at a certain suction due to the the uniform particle size distribution. Finer soils with smaller pores tend to exhibit flatter retention properties and tend to hold water at greater tensions, as capillary action is greater in capillaries of smaller diameter. Looking at the TDR data, we observe a rapid increase and decrease in water content in TDR 1, which suggests that a significant part of the anomalies pores are filled and drained rapidly indicating steep retention properties. However the anomaly during the November event does not fall below $\theta_{min,anomaly1} = 0.48$. As the anomalous regions are assumed to be extremely thin, these changes in water content do not necessarily indicate changes in the overall water content of the entire soil system. The first inverse run indicated that the variance cannot be predicted in the assumed ranges (set-up 1, Tab. 19). This suggests that the retention curve should be even steeper. However, keeping the field capacity constraint in mind, this also suggests an air entry at greater suction, resulting in rather soil-unusual van

Genuchten parameterization for the anomaly (set-up 2, Tab. 19). Overall, the search space ranges are quite difficult to determine as no pressure head information is available.

Evaluation criteria

Each TDR water content reference data was compared to the mean of the water content of six observation points placed within each anomaly (Tab. 18). This was thought to sufficiently represent the anomaly, creating three mean water content time series. The coordinates of each observation point can be found in Tab. 18 and are depicted in Fig. 23. The objective function chosen for the optimization was the the mean of the logged $RMSE$. This means that each TDR's time series was treated as equally important. As the $RMSE$ only serves as an error criterion and does not serve as an indicator for the shape of the solution, several other criteria were additionally evaluated and stored in output files. Each anomaly was evaluated separately using logged $RMSE$, the Pearson's correlation coefficient, the ratio of the means and the ratio of the variance. R functions cor(), mean() and var() from the stats and base packages were used to evaluate the correlation coefficient, the means and the variances. Before evaluation took place, the simulated output was approximated by linear interpolation to match the hourly TDR output times using R's approx() function from the stats package. This assured that the number of compared values are the same and that there is no bias due to changing time steps.

The ratio of the means can be expressed as

$$ratio_{means} = \frac{\Sigma_{i=1}^{t} Sim_i}{\Sigma_{i=1}^{t} Obs_i} = \frac{\overline{Sim}}{\overline{Obs}} \tag{6.3}$$

where $\overline{Sim}$ is the mean of the simulated value, Sim_i is the simulated value at time i, $\overline{Obs}$ is the mean of the observed value and Obs_i is the observed value at time i. The mean is usually the sum of all values divided by the number of values. Since the number of values is identical, they cancel each other out. The ratio of the variance can be expressed as

$$ratio_{variance} = \frac{\Sigma_{i=1}^{t} (Sim_i - \overline{Sim})^2}{\Sigma_{i=1}^{t} (Obs_i - \overline{Obs})^2} = \frac{\sigma_{Sim}^2}{\sigma_{Obs}^2}, \tag{6.4}$$

where σ_{Sim}^2 is the variance of the simulated data and σ_{Obs}^2 is the variance of the observed data. Again, the number of values is identical and therefore cancel each other out in this expression.

The Pearson's correlation coefficient for our case is

$$r = \frac{\Sigma_{i=1}^{t} (Sim_i - \overline{Sim})(Obs_i - \overline{Obs})}{\sqrt{\Sigma_{i=1}^{t} (Sim_i - \overline{Sim})^2} \sqrt{\Sigma_{i=1}^{t} (Obs_i - \overline{Obs})^2}}. \tag{6.5}$$

6.3.2 Results and Discussion

The inverse modeling task could not be fully completed as simulation time of the dual permeability model took too long and due to necessary modifications of the project design. However, the preliminary results and the preliminary runs already raise interesting questions. Fig. 24 depicts the results for the optimized standard model with two different parameter ranges. It was decidednot to show the dual permeability estimates, as these were still too preliminary to provide meaningful information on the fit of the model. The standard model with set-up 1 (std 1 depicted in blue) has already converged and the global best does not change, however standard model with set-up 2 (std 2 depicted in blue) is still improving and finding better parameter combinations to fit the observations. The results indicate that the presented case study with physical van Genuchten parameters cannot be simulated with a uni-modal model. The adapted boundary did improve the fit for TDR1 slightly, however the identified van Genuchten parameters might not be physical. Nonetheless, it raises the question: Do artificial slurries made of native soil and exhibiting cracks modify physically meaningful ranges of van Genuchten parameterization? Moreover, it also raises the question: How do water retention properties change? More precisely, is water in cracks within these slurries, which contain a moderate clay content, held longer or held at greater tensions, when compared to water in pores of equivalent diameter in natural soils? And would this explain unusual van Genuchten parameterization? Furthermore, despite the different set-ups, very similar parameters were identified for soil 1 and soil 2. In both set-ups the saturated hydraulic conductivity in the upper soil was identified to be 24.53 and 22.76 cm h^{-1} (588 and 546 cm d^{-1}). Although these values are still in the upper range suggested by Kogelbauer et al. (2015), it is unlikely that the soil, a loamy Chernozem, truly exhibits these conductivities. It can be seen as another indication that fractures and cracks must play an important role in the water transport. Results presented in Kogelbauer et al. (2015) using a mobile-immobile dual porosity model showed a much better fit.

As the inverse modeling with the dual permeability model is still on-going, we cannot yet say whether the dual permeability model is able to improve the simulation outcome. To increase variance in the artificial envelope, the fracture definition of the upper soil can be influential. With high infiltration weights, the fracture in this set-up represents the main conducting domain in the dual permeability approach. It still needs to be analyzed if the fracture definition in the top soil is adequate. The location of the population of potential solutions can be an indication that the optimal values lie outside of the chosen range. If the majority of the population, the set of potential solutions, is very close to the boundary the range should be adapted. However, the optimization has not progressed far enough to be able to tell.

Furthermore, since the shape of the water content seemed to be the biggest challenge, it might be advisable to use a different, more holistic, optimization criterion, which also takes

variance into account, such as the Kling-Gupta-Efficiency (KGE). The KGE is a maximization criterion, so either the negative KGE would need to be optimized or the algorithms need to be restructured in order to allow for maximization problems as well. It is already extremely helpful to evaluate different evaluation criteria.

The TDR sensor readings were calibrated with water content data from undisturbed soil samples. Only samples under medium wet conditions were considered to not violate the hypothesis of preferential flow and anomalous TDR readings under wet conditions. An F-test was conducted to test if the variance of the water content measurements of the undisturbed samples and the TDR readings are significantly different. The F-test did not result in significant differences, which lead to the conclusion that the slope of the linear TDR calibration function is best described by unity (Doležal et al. 2012, 2015). Considering the low sample size (4-7) and the limited moisture range covered by the calibration, the conclusion that the TDR readings and the water content of undisturbed samples are best related to each other with a slope of unity for the entire moisture range may have to be revisited. Especially when deviating form physical-based parameterization ranges, there is a certain risk of simulating a TDR sensor artifact rather than true soil hydraulic properties.

The rain event in November was not particularly strong and with additional data, it would be more interesting to calibrate more intense rain events. It is out of the question that after calibration is completed, the results require validation. If the dual permeability model is significantly superior in other calibration events, it would be useful to calculate percolation fluxes as suggested in Pirastru and Niedda (2010) and compare these to the saturated hydraulic conductivity in the matrix to analyse the importance of bypass flow compared to matrix driven flow.

To reduce the simulation time of the dual permeability model, attempts were made to change the project design and numerics. For the 1D simulation, implemented changes reduced the simulation time tremendously. Although more work is required to reduce the simulation time for 2D runs, some progress has been made to reduce the simulation time in 2D. The calibration time was reduced to only cover one rain event instead of a series of three rain events in November. The Picard iteration criterion was set high (M. Kuraz, oral communication). It was assumed that the high Picard criterion can help to find more appropriate ranges for later optimization runs with a lower Picard iteration criterion. Nonetheless, the high Picard criterion lead to unrealistic results and some oscillation. With a low Picard criterion the simulation takes several hours, with some exceptions taking more than 12 hours with the optimized mesh. Not all parameter sets require a long period of time, however the algorithm cannot commence before all simulations have finished. Computers with 32 cores were used and the population was set to 64. The minimum population should not be lower than 40 for $TLBO_{scebn}$. Each evaluation call therefore takes two sets of parallel runs. The algorithm

requires two evaluation for each iteration, which would have resulted in an iteration taking 48 hours. Even with a low iteration number of 100, this would have resulted in a simulation time of several months. Despite intensive efforts, the issues could not be resolved in time and were out of scope for this thesis. An ideal set-up is yet to be found and will be part of future research. The uni-modal problem, in contrast, was run on computers with 8 cores. Despite the limited number of cores, the computation time of one iteration with the standard model is significantly faster. This allowed the use of a low Picard criterion, a finer 1D mesh, a longer pre-initialization time and smaller maximum time step. Generally, we can conclude that with the present set-up, 2D non-coupled problems are more suitable when using population-based metaheuristics and feasible ranges of parameters can be found quite quickly.

Changing the evaluation set-up may also help reduce the simulation time. To improve the simulation, a time punishment factor for computations exceeding a maximum time can be introduced. If the maximum simulation time is reached and the simulation has not finished, the error criterion can be set artificially high. For this it is necessary to know what simulation time can be considered appropriate. The introduction of a punishment factor may cause good solutions to be punished when the maximum simulation time is set inappropriately. This approach is therefore not recommended when the ranges of the design parameters are not known at all.

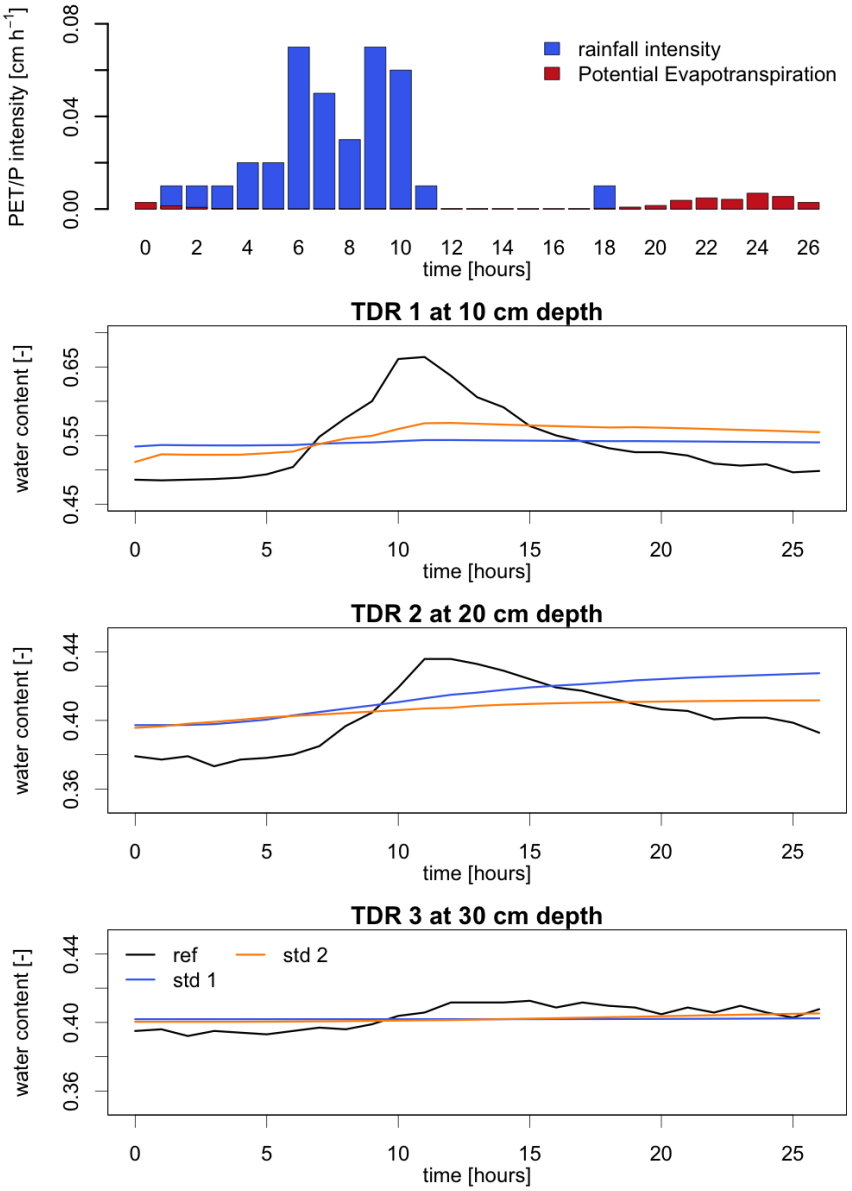


Figure 24: Optimized results with uni-modal standard model and input rainfall intensity and potential evapotranspiration data

Table 19: Design variables for inverse modeling, assigned boundaries and identified value for dual permeability model set-up 1.

Material	Parameter	Symbol	Min	Max	Identified value	
					std 1	std 2
Soil 1 dual	inverse of air entry value in matrix [cm^{-1}]	α_m	0.005	0.05	-	-
	shape parameter in matrix [-]	n_m	1.3	2.3	-	-
	sat. water content in matrix [-] [a]	$\theta_{s,m}$	-	-	-	-
	res. water content in matrix [-]	$\theta_{r,m}$	-	-	0	
	sat hydraulic conductivity of matrix [cm h^{-1}]	$K_{s,m}$	0.1	5	-	-
	inverse of air entry value in fracture [cm^{-1}]	α_f	0.01	0.15	-	-
	shape parameter in fracture [-]	n_f	2	3	-	-
	sat. water content in fracture [-]	$\theta_{s,f}$	0.3	0.5	-	-
	res. water content in fracture [-]	$\theta_{r,f}$	-	-	0	
	sat hydraulic conductivity of fracture [cm h^{-1}]	$K_{s,f}$	15	30	-	-
	fracture weight	w_f	0.05	0.3	-	-
	infiltration weight	w_{inf}	0.5	0.95	-	-
	distance from center of aggregates [cm]	a	0.5	1.5	-	-
	exchange boundary conductivity [cm h^{-1}]	K_a	1e-7	1e-4	-	-
Soil 1 uni-modal	inverse of air entry value [cm^{-1}]	α	0.01	0.1	0.01003	0.01003
	shape parameter[-]	n_m	1.8	2.5	2.472	2.217
	sat. water content [-]	θ_s	-	-	0.457	
	res. water content [-]	θ_r	-	-	0	
	sat hydraulic conductivity [cm h^{-1}]	K_s	0.1	25	24.53	22.76
Soil 2 [b]	fracture weight [c]	w_f	-	-	0.5	
	distance from center of aggregates [cm] [c]	a	-	-	1	
	exchange boundary conductivity [cm h^{-1}] [c]	K_a	-	-	0.1	
	inverse of air entry value [cm^{-1}]	α	0.01	0.1	0.076	0.037
	shape parameter	n [-]	5	10	1.500	1.506
	sat. water content [-]	θ_s	-	-	0.457	
	res. water content [-]	θ_r	-	-	0	
	sat hydraulic conductivity [cm h^{-1}]	K_s	0.1	20	6.23	4.12
Anomaly 1 [b]	fracture weight [-] [c]	w_f	-	-	0.5	
	distance from center of aggregates [cm] [c]	a	-	-	1	
	exchange boundary conductivity [cm h^{-1}] [c]	K_a	-	-	0.1	
set-up 1	inverse of air entry value [cm^{-1}]	α	0.01	0.07	0.0154	-
set-up 2	inverse of air entry value [cm^{-1}]	α	0.003	0.015	-	0.0056
set-up 1	shape parameter [-]	n	1.3	2	1.3167	-
set-up 2	shape parameter [-]	n	2	4.5	-	4.45
set-up 1 dual	inverse of air entry value [cm^{-1}]	α	0.01	0.07	-	-
set-up 2 dual	inverse of air entry value [cm^{-1}]	α	0.005	0.03	-	-
set-up 1 dual	shape parameter [-]	n	1.3	2	-	-
set-up 2 dual	shape parameter [-]	n	2.5	3.5	-	-
	sat. water content [-]	θ_s	0.68	0.72	0.6831	0.71
	res. water content [-]	θ_r	-	-	0	
	sat hydraulic conductivity [cm h^{-1}]	K_s	0.1	10	8.312	8.912
Anomaly 2 [b]	fracture weight [-] [c]	w_f	-	-	0.5	
	distance from center of aggregates [cm] [c]	a	-	-	1	
	exchange boundary conductivity [cm h^{-1}] [c]	K_a	-	-	0.1	
set-up 1	inverse of air entry value [cm^{-1}]	α	0.01	0.07	0.045	-
set-up 2	inverse of air entry value [cm^{-1}]	α	0.005	0.05	-	0.046
set-up 1	shape parameter [-]	n	1.3	2	1.306	-
set-up 2	shape parameter [-]	n	1.3	2.5	-	1.307
	sat. water content [-]	θ_s	0.45	0.52	0.52	0.519
	res. water content [-]	θ_r	-	-	0	
	sat hydraulic conductivity [cm h^{-1}]	K_s	0.1	10	9.962	9.484

[a] computed from $\theta_{s,f}$ and total θ_s=0.457, where $\theta_{s,m} = \frac{\theta_s - \theta_{s,f} \cdot w_f}{1 - w_f}$

[b] Fracture and matrix domains have same properties to simulate uni-modal soil

[c] Only dual

7 Conclusion

In this thesis the dual permeability model was implemented in the open free software Dual Richards' Unsaturated Equation Solver (*DRUtES*) and test simulation were presented to show when preferential bypass flow can occur. Additionally, population-based metaheuristic algorithms were presented that can be used for global optimization and bi-objective optimization. Due to numerical difficulties and long simulation time of the dual permeability model, the inverse modeling could only be set-up, but not completed. Nonetheless, important questions could be raised on the basis of preliminary results.

The main further research aims include:

- Better understanding of when simulations with the dual permeability become ill-conditioned and how to treat this.
- Better numerical treatment of the dual permeability model implementation in 2D to decrease computational time and thus increase usability for inverse modeling.
- Enhanced verification of the dual permeability model with more appropriate real data.
- The creation of tutorials on how to use *DRUtES* and the mesh generator in combination with *DRUtES* to increase accessibility and usability of *DRUtES*.
- Improved understanding of retention properties of artificial macropores with moderate clay content.
- Validation of the presented case study.

References

Allen, Richard G., L. S. Pereira, D. Raes, and M. Smith. 1998. *FAO Irrigation and drainage paper No. 56*. FAO - Food / Agriculture Organization of the United Nations. ISBN: 92-5-104219-5.

Amaran, Satyajith, Nikolaos V. Sahinidis, Bikram Sharda, and Scott J. Bury. 2015. "Simulation optimization: a review of algorithms and applications". *Annals of Operations Research* 240, no. 1 (): 351–380. ISSN: 0254-5330, 1572-9338. doi:10.1007/s10479-015-2019-x.

Beven, Keith, and Peter Germann. 2013. "Macropores and water flow in soils revisited". *Water Resources Research* 49, no. 6 (): 3071–3092. ISSN: 1944-7973. doi:10.1002/wrcr.20156.

Bianchi, Leonora, Marco Dorigo, Luca Maria Gambardella, and Walter J. Gutjahr. 2009. "A survey on metaheuristics for stochastic combinatorial optimization". *Natural Computing* 8 (2): 239–287. ISSN: 1567-7818, 1572-9796. doi:10.1007/s11047-008-9098-4.

Biggar, Paul, Nicholas Nash, Kevin Williams, and David Gregg. 2008. "An Experimental Study of Sorting and Branch Prediction". *J. Exp. Algorithmics* 12 (): 1.8:1–1.8:39. ISSN: 1084-6654. doi:10.1145/1227161.1370599.

Blum, Christian, and Andrea Roli. 2003. "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison". *ACM Comput. Surv.* 35, no. 3 (): 268–308. ISSN: 0360-0300. doi:10.1145/937503.937505.

Boussaïd, Ilhem, Julien Lepagnot, and Patrick Siarry. 2013. "A survey on optimization metaheuristics". *Information Sciences*, Prediction, Control and Diagnosis using Advanced Neural Computations, 237:82–117. ISSN: 0020-0255. doi:10.1016/j.ins.2013.02.041.

Buckingham, Edgar. 1907. "Studies on the movement of soil moisture".

Bäck, Thomas, and Hans-Paul Schwefel. 1993. "An Overview of Evolutionary Algorithms for Parameter Optimization". *Evolutionary Computation* 1, no. 1 (): 1–23. ISSN: 1063-6560, visited on 10/12/2016. doi:10.1162/evco.1993.1.1.1. http://dx.doi.org/10.1162/evco.1993.1.1.1.

Cheney, Elliott Ward, and David Ronald Kincaid. 2008. *Numerical Mathematics and Computing*. 6th ed. Brooks/Cole. ISBN: 978-0-495-38472-4.

Czachor, Henryk. 2011. "Laminar and Turbulent Flow in Soils". In *Encyclopedia of Agrophysics*, ed. by Jan Gliński, Józef Horabik, and Jerzy Lipiec, 413–413. Encyclopedia of Earth Sciences Series. DOI: 10.1007/978-90-481-3585-1_80. Springer Netherlands. ISBN: 978-90-481-3584-4 978-90-481-3585-1.

Darcy, Henry. 1856. *Les fontaines publiques de la ville de Dijon.*

Das, S., and P. N. Suganthan. 2011. "Differential Evolution: A Survey of the State-of-the-Art". *IEEE Transactions on Evolutionary Computation* 15, no. 1 (): 4–31. ISSN: 1089-778X. doi:10.1109/TEVC.2010.2059031.

Doležal, František, Svatopluk Matula, and J. M. Moreira Barradas. 2012. "Percolation in macropores and performance of large time-domain reflectometry sensors". *Plant, Soil and Environment* 58:503–507.

– . 2015. "Rapid percolation of water through soil macropores affects reading and calibration of large encapsulated TDR sensors". *Soil and Water Research* 10 (3): 155–163. doi:10.17221/177/2014-SWR.

Dréo, Johann. 2011. *Different classifications of metheuristics*. Visited on 10/16/2016. https://commons.wikimedia.org/wiki/File:Metaheuristics_classification.svg.

Duan, Q. Y., V. K. Gupta, and S. Sorooshian. 1993. "Shuffled complex evolution approach for effective and efficient global minimization". *Journal of Optimization Theory and Applications* 76 (3): 501–521. ISSN: 0022-3239, 1573-2878, visited on 10/31/2016. doi:10.1007/BF00939380.

Durner, Wolfgang, and Hannes Flühler. 2005. "Soil hydraulic properties". *Encyclopedia of Hydrological Sciences*.

Eberhart, RC, and J Kennedy. 1995. "A new optimizer using particle swarm theory, Proceedings of the Sixth International Symposium on Micro Machine and Human Science: Oct 1995".

Engelbrecht, Andries P. 2005. *Fundamentals of Computational Swarm Intelligence*. Wiley. ISBN: 978-0-470-09191-3.

Genuchten, M.Th. van. 1980. "Closed-form equation for predicting the hydraulic conductivity of unsaturated soils". *Soil Science Society of America Journal* 44 (5): 892–898.

Gerke, H. H., and M. T. van Genuchten. 1993a. "A dual-porosity model for simulating the preferential movement of water and solutes in structured porous media". *Water Resources Research* 29 (2): 305–319.

– . 1993b. "Evaluation of a first-order water transfer term for variably saturated dual-porosity flow models". *Water Resources Research* 29, no. 4 (): 1225–1238. ISSN: 1944-7973. doi:10.1029/92WR02467.

Gerke, Horst H. 2011. "Bypass Flow in Soil". In *Encyclopedia of Agrophysics*, ed. by Jan Gliński, Józef Horabik, and Jerzy Lipiec, 100–105. Encyclopedia of Earth Sciences Series. DOI: 10.1007/978-90-481-3585-1_23. Springer Netherlands. ISBN: 978-90-481-3584-4 978-90-481-3585-1.

Germann, Peter F. 1985. "Kinematic wave approach to infiltration and drainage into and from soil macropores". *Transactions of the ASAE* 28 (3): 745–0749.

Germann, Peter F., and Keith Beven. 1985. "Kinematic Wave Approximation to Infiltration Into Soils With Sorbing Macropores". *Water Resources Research* 21, no. 7 (): 990–996. ISSN: 1944-7973. doi:10.1029/WR021i007p00990.

Geuzaine, Christophe, and Jean-François Remacle. 2009. "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities". *International Journal for*

Numerical Methods in Engineering 79, no. 11 (): 1309–1331. ISSN: 1097-0207. doi:10.1002/nme.2579.

Gogna, Anupriya, and Akash Tayal. 2013. "Metaheuristics: review and application". *Journal of Experimental & Theoretical Artificial Intelligence* 25, no. 4 (): 503–526. ISSN: 0952-813X. doi:10.1080/0952813X.2013.782347.

Haahr, Mads, and Sven Haahr. 2017. *RANDOM.ORG - True Random Number Service*. Visited on 03/04/2017. https://www.random.org/.

Hillel, Daniel. 2004. *Soil and Water: Physical Principles and Processes*. New York: American Press. ISBN: 978-0-323-15670-7.

Hu, Xiaohui, and R. Eberhart. 2002. "Multiobjective optimization using dynamic neighborhood particle swarm optimization". In *Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC '02*, 2:1677–1681. doi:10.1109/CEC.2002.1004494.

Iden, S. C., and W. Durner. 2007. "Free-form estimation of the unsaturated soil hydraulic properties by inverse modeling using global optimization" [**inlang**en]. *Water Resources Research* 43, no. 7 (): W07451. ISSN: 1944-7973, visited on 10/12/2016. doi:10.1029/2006WR005845.

Jakubcova, Michala, Petr Maca, and Pavel Pech. 2015. "Parameter Estimation in Rainfall-Runoff Modelling Using Distributed Versions of Particle Swarm Optimization Algorithm". WOS:000363210700001, *Mathematical Problems in Engineering*: 968067. doi:10.1155/2015/968067.

Jakubcová, Michala, Petr Máca, and Pavel Pech. 2014. "A comparison of selected modifications of the particle swarm optimization algorithm". *Journal of Applied Mathematics* 2014.

Kar, Arpan Kumar. 2016. "Bio inspired computing – A review of algorithms and scope of applications". *Expert Systems with Applications* 59 (): 20–32. ISSN: 0957-4174. doi:10.1016/j.eswa.2016.04.018.

Kogelbauer, Ilse, Kamila Bát'ková, František Doležal, Svatopluk Matula, and Willibald Loiskandl. 2015. "Preferential percolation quantified by large water content sensors with artifactual macroporous envelopes". *Hydrological Processes* 29, no. 19 (): 4325–4338. ISSN: 1099-1085, visited on 10/04/2016. doi:10.1002/hyp.10491.

Kuraz, Michal, and Petr Mayer. 2008. *DRUtES – an opensource library for solving coupled nonlinear convection-diffusion-reaction equations*. http://www.drutes.org.

Liang, J. J., B. Y. Qu, P. N. Suganthan, and Alfredo Hernándes-Diaz. 2013. *Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization*. Tech. rep.

Maca, Petr, and Pavel Pech. 2015. "The Inertia Weight Updating Strategies in Particle Swarm Optimisation Based on the Beta Distribution". WOS:000353804300001, *Mathematical Problems in Engineering*: 790465. doi:10.1155/2015/790465.

Matsumoto, Makoto, and Takuji Nishimura. 1998. "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8 (1): 3–30.

Mualem, Y. 1976. "A new model for predicting the hydraulic conductivity of unsaturated porous media". *Water Resources Research* 12 (3): 513–522. doi:10.1029/WR012i003p00513.

Naka, S., T. Genji, T. Yura, and Y. Fukuyama. 2003. "A hybrid particle swarm optimization for distribution state estimation". *IEEE Transactions on Power Systems* 18, no. 1 (): 60–68. ISSN: 0885-8950. doi:10.1109/TPWRS.2002.807051.

Nieder, Rolf. 2011. *Bodenkunde I. Grundlagen der Bodenkunde für 3. Semester Geoökologie und Umweltingenieurwesen*. TU Braunschweig, Institut für Geoökologie, Abteilung für Bodenkunde und Bodenphysik.

Piotrowski, Adam P., Maciej J. Napiorkowski, Jaroslaw J. Napiorkowski, Marzena Osuch, and Zbigniew W. Kundzewicz. 2017. "Are modern metaheuristics successful in calibrating simple conceptual rainfall–runoff models?" *Hydrological Sciences Journal* 62, no. 4 (): 606–625. ISSN: 0262-6667. doi:10.1080/02626667.2016.1234712. http://dx.doi.org/10.1080/02626667.2016.1234712.

Pirastru, Mario, and Marcello Niedda. 2010. "Field monitoring and dual permeability modelling of water flow through unsaturated calcareous rocks". *Journal of Hydrology* 392, no. 1–2 (): 40–53. ISSN: 0022-1694. doi:10.1016/j.jhydrol.2010.07.045.

R Core Team. 2015. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Rank, Ernst, Casimir Katz, and Heinrich Werner. 1983. "On the importance of the discrete maximum principle in transient analysis using finite element methods". *International Journal for Numerical Methods in Engineering* 19, no. 12 (): 1771–1782. ISSN: 1097-0207. doi:10.1002/nme.1620191205.

Rao, R. V., V. J. Savsani, and D. P. Vakharia. 2012. "Teaching–Learning-Based Optimization: An optimization method for continuous non-linear large scale problems". *Information Sciences* 183, no. 1 (): 1–15. ISSN: 0020-0255. doi:10.1016/j.ins.2011.08.006.

Ratnaweera, Asanga, Saman K. Halgamuge, and Harry C. Watson. 2004. "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients". *IEEE Transactions on evolutionary computation* 8 (3): 240–255.

RStudio Team. 2015. *RStudio: Integrated Development Environment for R*. Boston, MA: RStudio, Inc. http://www.rstudio.com/.

Shi, Yuhui, and Russell Eberhart. 1998. "A modified particle swarm optimizer". In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, 69–73. IEEE.

Simon, Dan. 2013. *Evolutionary Optimization Algorithms*. Wiley. ISBN: 978-0-470-93741-9.

Sörensen, Kenneth. 2015. "Metaheuristics—the metaphor exposed". *International Transactions in Operational Research* 22, no. 1 (): 3–18. ISSN: 1475-3995. doi:10.1111/itor.12001.

Weyland, Dennis. 2010. "A Rigorous Analysis of the Harmony Search Algorithm: How the Research Community can be Misled by a "Novel" Methodology". *International Journal of Applied Metaheuristic Computing* 1 (2): 50–60. ISSN: 1947-8283, 1947-8291. doi:10.4018/jamc.2010040104.

Wolpert, D. H., and W. G. Macready. 1997. "No free lunch theorems for optimization". *IEEE Transactions on Evolutionary Computation* 1, no. 1 (): 67–82. ISSN: 1089-778X. doi:10.1109/4235.585893.

Yan, Jiang, Hu Tiesong, Huang Chongchao, Wu Xianing, and Gui Faling. 2007. "A Shuffled Complex Evolution of Particle Swarm Optimization Algorithm". In *Adaptive and Natural Computing Algorithms*, ed. by Bartlomiej Beliczynski, Andrzej Dzielinski, Marcin Iwanowski, and Bernardete Ribeiro, 341–349. Lecture Notes in Computer Science 4431. DOI: 10.1007/978-3-540-71618-1_38. Springer Berlin Heidelberg. ISBN: 978-3-540-71589-4 978-3-540-71618-1.

Zou, Feng, Lei Wang, Xinhong Hei, and Debao Chen. 2015. "Teaching–learning-based optimization with learning experience of other learners and its application". *Applied Soft Computing* 37 (): 725–736. ISSN: 1568-4946. doi:10.1016/j.asoc.2015.08.047.

Šimůnek, J., M Šejna, and M. Th van Genuchten. 2012. *The DualPerm Module for HYDRUS (2D/3D) Simulating Two-Dimensional Water Movement and Solute Transport in Dual-Permeability Porous Media, Version 1.0*. Prague: PC Progress.

Šimůnek, Jirka, Nick J. Jarvis, M. Th. van Genuchten, and Annemieke Gärdenäs. 2003. "Review and comparison of models for describing non-equilibrium and preferential flow and transport in the vadose zone". *Journal of Hydrology*, Soil Hydrological Properties and Processes and their Variability in Space and Time, 272, no. 1–4 (): 14–35. ISSN: 0022-1694. doi:10.1016/S0022-1694(02)00252-4.

8 Appendix

8.1 Appendix A

Comparison of dual permeability model with standard model

The dual permeability model with all five hydraulic conductivity variants were compared to the standard model by setting the fracture and matrix domain the same with parameterization in Tab. 20. Fig. 25 shows the results of the scenarios. The dual permeability models outputs are identical to the standard output. Simulation output in Fig. 25 make it hard to identify any gray lines indicating the dual permeability model as the alignment with the standard model is too good. The exchange boundary conductivity K_a is non-zero for scenario 1, 2 and 3. Fig. 26 shows a ponding scenario with free drainage (scenario 4) at the bottom for the set up as in Tab. 20. The output times reflect the beginning of the simulation, where deviations between the standard model and the dual permeability model are noticeable. The modified set-up included a decreased Picard iteration criterion, decreased maximum time-step and direct evaluation of the constitutive equations. The modified set-up improved the simulation. In both set-ups the dual permeability model converges towards the standard model.

Table 20: Soil hydraulic properties and domain description used in test simulations of the dual permeability model with the standard model.

Parameter	**Symbol**	**Value**
Inverse of air entry value [cm^{-1}]	α	0.05
Shape parameter	n	2
Sat. water content [-]	θ_s	0.45
Residual water content [-]	θ_r	0
Sat. hydraulic conductivity [cm d^{-1}]	K_s	200
Domain length [cm]	L	10
Spatial discretization [cm]	dx	0.1
Simulation time [days]	t	2
Minimum time step [days]	dt_{min}	5e-8
Maximum time step [days]	dt_{max}	0.1

Table 21: Initial and boundary conditions used for test simulations of the dual permeability model with the standard model in *DRUtES*.

Scenario	**Initial h_{pres}**	**Top bc**	**Bottom bc**
1	-100 cm	-50 cm	-50 cm
2	-100 cm	0.5 cm d^{-1}	-100 cm
3	-10 cm	-0.2 cm d^{-1}	-10 cm
4	-50 cm	0 cm	free drainage

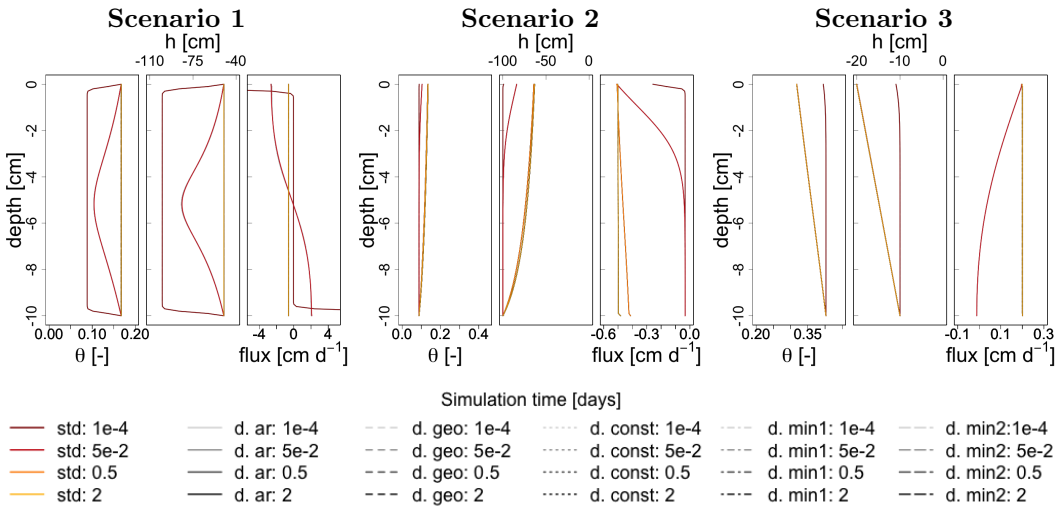


Figure 25: Simulation output after 1e-4, 5e-2, 0.5 and 2 days of test scenarios 1, 2 and 3. The gray lines cannot be recognized because the dual variants and the standard model align perfectly.

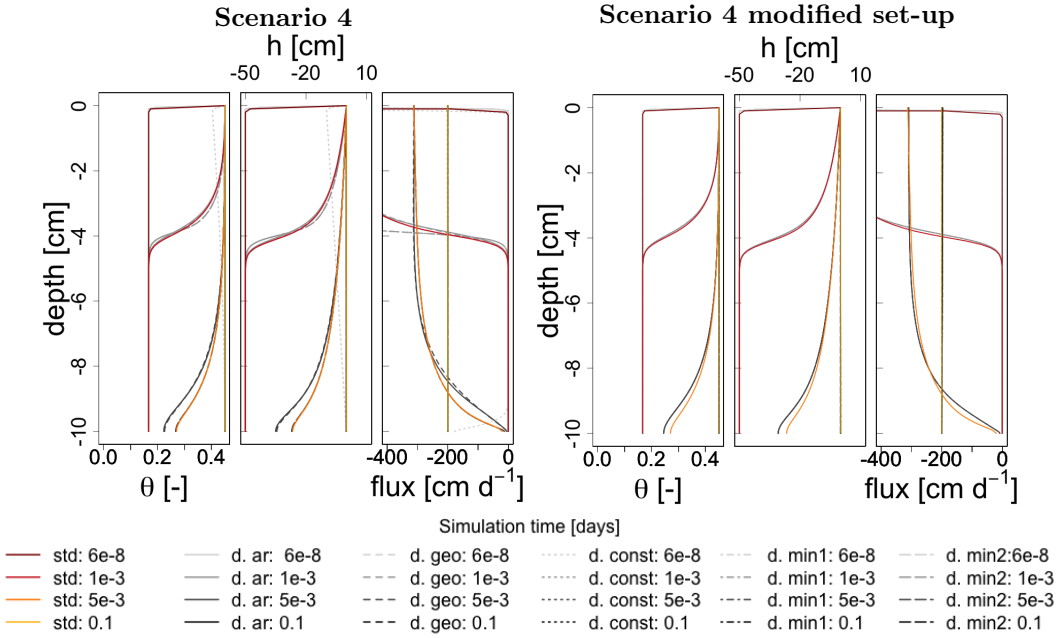


Figure 26: Simulation output after 6e-8, 1e-3, 5e-3 and 0.1 days of test scenarios 4 with different set-up

8.2 Appendix B

Fig. 27 depicts boxplots of 30D CEC Benchmark functions without any reinitialization during the optimization. Comparing these to calibrated optimization output, it becomes evident that, generally, reinitialization can enhance the optimization result immensely in most cases.

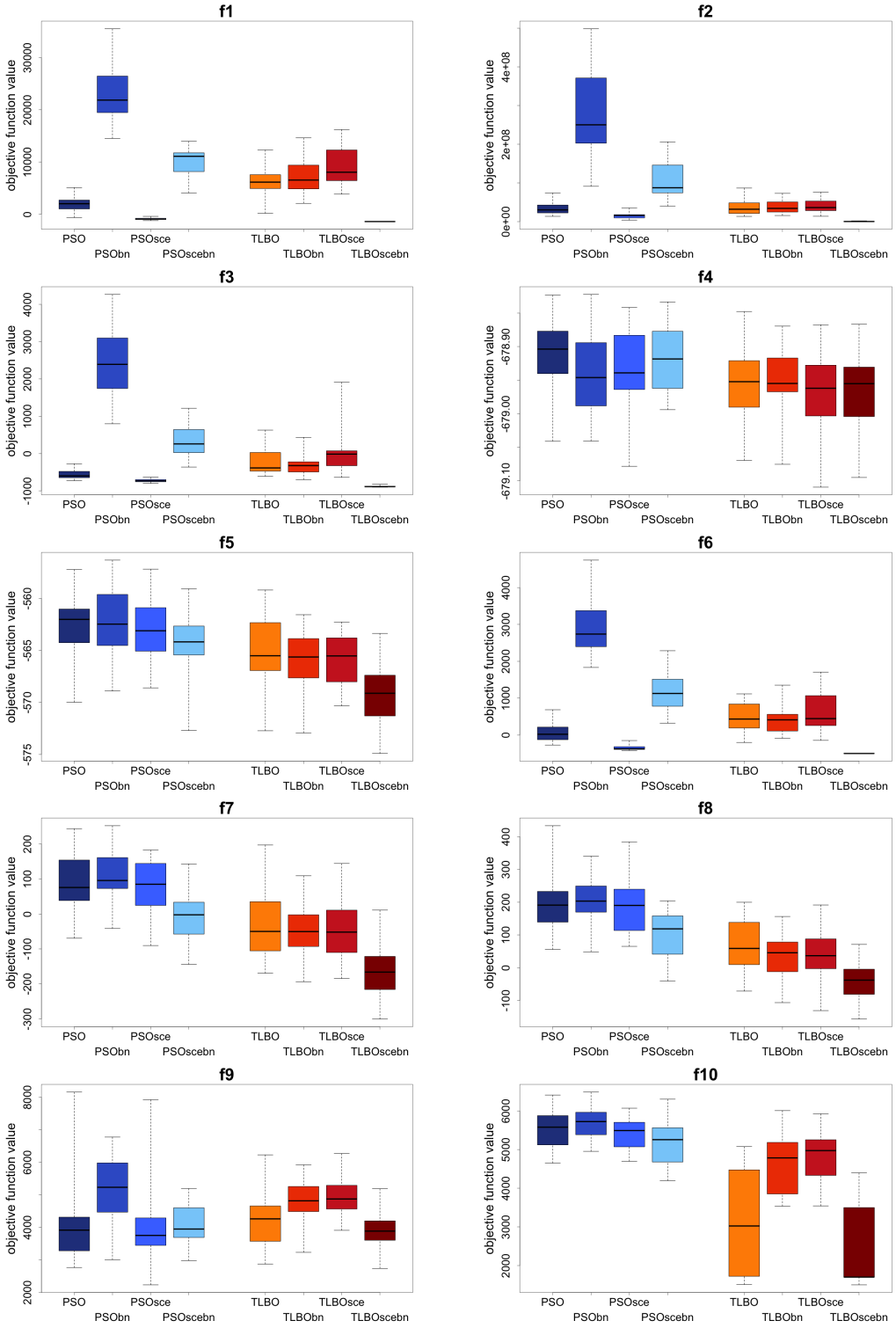


Figure 27: Boxplots of 30 runs of 30D benchmark functions without reinitialization.

8.3 Appendix C

The Github repositories were solely created with the aim to make code used in this thesis accessible.Following codes are accessible on Github.

1. The *DRUtES* version containing the dual permeability model implementation: https://github.com/Jorub/dual_permeability
2. The CEC benchmark function set-up: https://github.com/Jorub/CEC_opti
3. The mesh optimization set-up: https://github.com/Jorub/mesh_opti
4. The *DRUtES* benchmark set-up: https://github.com/Jorub/bench_drutes_setup
5. The *DRUtES* inverse modeling set-up for the dual permeability model: https://github.com/Jorub/inverse_dual