Tutorial for interfacing *hydroPSO** with SWAT-2005 and MODFLOW-2005

Rodrigo Rojas, PhD
Mauricio Zambrano-Bigiarini, PhD

December 17, 2012

*for hydroPSO >= 0.2-0

Contents

List of Tables

List of Figures

Preface

The next tutorial shows how to use the *hydroPSO* R package to calibrate different model codes. Two of the main properties of *hydroPSO* are the independence from the model to be calibrated and the simple interfacing between the model code and the calibration-engine, i.e. *hydroPSO*. These two properties are illustrated through the calibration of real-world case studies, which should be the starting point for a user planning to implement *hydroPSO* for her/his own model calibration exercise.

First, we illustrate the main features of *hydroPSO* when used to calibrate any objective function fully defined within the R environment. Second, we show how to use *hydroPSO* to calibrate a semi-distributed hydrological model and a steady-state groundwater flow model. The first is implemented in SWAT-2005 (*Neitsch et al.*, 2005) whereas the second is implemented in MODFLOW-2005 (MF2005) (*Harbaugh*, 2005). Both model codes are widely-used programs to simulate surface water and groundwater flow problems. At the same time, for illustrative purposes we use the program ZONEBUDGET (ZB) (*Harbaugh*, 1990) to obtain groundwater flows at particular aquifer cross sections. The purpose of including ZB is to illustrate how to interface *hydroPSO* with a modelling application that involves more than one executable code. In the latter case, however, only MF2005 is calibrated against observed data, and including other observations can be easily implemented in a similar way as described here.

Both programs SWAT-2005 and MF2005/ZB are public domain and accessible from `http://swatmodel.tamu.edu/software/swat-model/` and `http://water.usgs.gov/nrp/gwsoftware/modflow.html`, respectively. For simplicity, we must assume some basic knowledge from part of the reader about the file naming convention used in SWAT-2005, MF2005/ZB, as well as the setting up of the internal options for these programs.

As an integral part of this tutorial, the reader can download from `http://www.rforge.net/hydroPSO/` two files MF2005.zip and SWAT2005.zip including all the required files to run the SWAT-2005 and MF2005 applications, sample R scripts to interface *hydroPSO* with SWAT-2005 and MF2005, as well as several auxiliary files.

For assistance, bugs report, comments, and suggestions please contact the authors of the *hydroPSO* package at: mzb.devel@gmail.com and/or Rodrigo.RojasMujica@gmail.com.

1 Introduction

hydroPSO is an R package implementing the Particle Swarm Optimisation (PSO) algorithm (*Kennedy and Eberhart*, 1995; *Eberhart and Kennedy*, 1995). PSO is a population-based global optimisation technique inspired by social behaviour of bird flocking, which shares few similarities with other evolutionary optimisation techniques such as Genetic Algorithms (GA) (*Poli et al.*, 2007). In PSO, however, the solution-space is explored on the basis of individual and neighbourhood best-known "particle positions" with no presence of evolution operators (e.g. mutation or crossover) as in GA. PSO has recently received a surge of attention in literature given its flexibility, simplicity of implementation (programming), low memory and computational requirements, low number of adjustable parameters, and efficiency (see, e.g., *Eberhart and Shi*, 1998; *Shi and Eberhart*, 1999; *Eberhart and Shi*, 2001; *Poli et al.*, 2007).

The main motivations for developing the *hydroPSO* package are:

1. bring this powerful optimisation/calibration technique to the attention of practitioners and scientists working on environmental modelling,

2. provide a model-independent software package allowing the user to calibrate (different) environmental models without having to invest considerable programming efforts in customizing the calibration engine to the environmental model, and

3. allow the PSO research community to explore alternative configurations and recent developments of the standard PSO using a versatile single-package software.

Unlike other R packages recently developed for similar purposes (see e.g. *hydromad* *Andrews et al.* 2011 and *R-SWAT-FME* *Wu and Liu* 2012), *hydroPSO* is not restricted to a limited number of hard-coded models, can be interfaced with any model with a relatively low programming effort, is fully compatible with calibration tools employing PEST-like template files (see *Doherty*, 2010), and is easily parallelizable.

hydroPSO is also capable of performing sensitivity analysis using the Latin Hypercube One-factor-At-a-Time (LH-OAT) method (see *van Griensven et al.*, 2006) and provides detailed information about the evolution of PSO's performance. In addition, advanced plotting capabilities and a complete family of built-in functions contained in the R language (*R Development Core Team*, 2011) are available to visualise, analyse and summarise the calibration results. At the same time, *hydroPSO* features a suite of controlling options and PSO variants for fine-tuning and improving the performance of PSO,

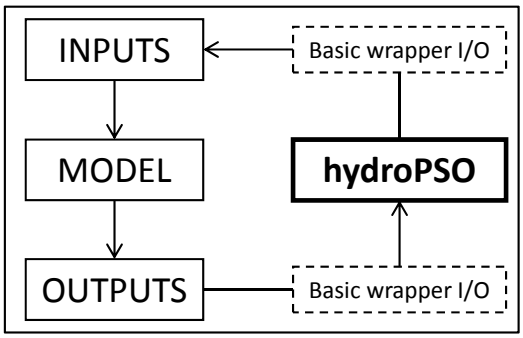

Figure 1: Flow chart of the implementation/interaction between *hydroPSO* and the model code to be calibrated. Dashed-line boxes represent basic I/O functions to read/write model files (after *Zambrano-Bigiarini and Rojas*, 2012).

thus, allowing the user to adapt the calibration/optimisation engine to different modelling problems. For a full description of the PSO enhancements included in the *hydroPSO* package the reader is referred to Section 2.

In principle, *hydroPSO* only needs to know *which* model parameters need to be calibrated, *where* they need to be written, and *from where* and *how* to read the main model output. Then, it will take control of the model(s) to be calibrated until either a maximum number of iterations or an error tolerance in the objective function are reached: these two being problem-specific and user-defined. The basic interaction between *hydroPSO* and the model to be calibrated is shown in Figure 1. For models with numerous input and output files or cascading models (i.e. serial modelling applications), several I/O functions can be combined to interface *hydroPSO* and the model.

In this tutorial we show first how to interface *hydroPSO* with SWAT-2005 (*Neitsch et al.*, 2005) to calibrate a semi-distributed hydrological model for the Ega River basin in Spain. That section illustrates three main points: 1) basic interfacing between *hydroPSO* and the model through the definition of ASCII files `ParamRanges.txt` and `ParamFiles.txt`, used to define which model parameters need to be calibrated and where they need to be written; 2) advanced sensitivity analysis using LH-OAT and the use of *hydroPSO* with a pre-defined goodness-of-fit measure as objective function, and 3) the set up for calibrating a model output variable in time and static in space (transient application). Second, we show how to interface *hydroPSO* with MF2005 (*Harbaugh*, 2005) to calibrate a groundwater flow model for the regional aquifer of the Pampa del Tamarugal basin in Chile. That section, in turn, illustrates: 1) advanced interfacing between *hydroPSO* and the model through the implementation of user-defined I/O wrapper functions written

Table 1: Distinctive features of the SWAT-2005 and MF2005 applications presented in this tutorial.

Application feature	SWAT-2005	MODFLOW-2005
OS	GNU/Linux	Windows 7
Type of model	Semi-distributed, surface hydrology	Fully-distributed, groundwater
hydroPSO-model interface	Basic through ASCII files	Advanced through user-defined R functions
Executable model code	Single file (swat2005.out)	Sequential batch file (*.bat)
Simulated model outputs	Transient (1962-1970), single observation	Steady-state (1960), multiple observations
Goodness-of-fit measure	Pre-defined Nash-Sutcliffe efficiency	User-defined Gaussian likelihood

in R, 2) how to define/use a customised goodness-of-fit measure as objective function, and 3) how to set up *hydroPSO* for a model output fixed in time but variable in space (steady-state application). Details about both applications are summarised in Table 1.

We therefore believe that this tutorial cover two standard-problems commonly found by the modelling community. We must note, however, that these tutorials deal with surface water and groundwater modelling applications as these two are the authors' areas of expertise. However, based on the flexibility of the *hydroPSO* package and the benefits added by programming it in R, we believe this package can be implemented to a wider range of environmental models requiring some form of parameter estimation. To the date of writing this tutorial, we have provided support to users calibrating models such as: SWAT-2005, MF2005, LISFLOOD, AGNPS and HBV-96.

In Section 2 we present a brief description of PSO, including the main algorithm, discussion on topologies, PSO variants and fine-tuning options available in *hydroPSO*. Section 3 briefly describes the main package functions as well as the basic requirements to interface *hydroPSO* with a given model code. An introductory application of *hydroPSO* to optimise test functions commonly used to assess the performance of optimisation algorithms is described in Section 4. The procedure to interface *hydroPSO* with SWAT-2005 and MF2005 as well as the corresponding calibration results, are reported in Sections 5 and 6, respectively.

2 Particle Swarm Optimisation (PSO)

2.1 Canonical PSO Algorithm

Particle Swarm Optimisation (PSO) is a population-based global search algorithm developed by *Kennedy and Eberhart* (1995) and *Eberhart and Kennedy* (1995). A feature distinguishing PSO from evolutionary algorithms is the lack of genetic operators, instead each individual of the population, termed *particles* in PSO terminology, adjusts its *flying trajectory* around the multi-dimensional search-space according to its own flying experience and the one of all neighbouring particles in the swarm (*Eberhart and Shi*, 1998).

Considering a D-dimensional search-space, *position* and *velocity* for the i-th particle are represented by $\vec{X}_i = x_{i1}, x_{i2}, \ldots, x_{iD}$ and $\vec{V}_i = v_{i1}, v_{i2}, \ldots, v_{iD}$, respectively. The performance of each particle is assessed through a problem-specific "goodness-of-fit" measure, which is the basis for calculating $\vec{X}_i$, thus, reflecting the quality of the particle's position. The best-known position of the i-th particle (known as *personal best*) is represented by $\vec{P}_i = p_{i1}, p_{i2}, \ldots, p_{iD}$, whereas the best-known position for the neighbouring particles (known as *local best*) is represented by $\vec{G} = g_1, g_2, \ldots, g_D$. Velocity and position of the i-th particle are updated according to the following equations,

$$\vec{V}_i^{t+1} = \omega \vec{V}_i^t + c_1 \vec{U}_1^t \otimes (\vec{P}_i^t - \vec{X}_i^t) + c_2 \vec{U}_2^t \otimes (\vec{G}^t - \vec{X}_i^t) \tag{1}$$

or (*Clerc and Kennedy*, 2002),

$$\vec{V}_i^{t+1} = \chi \left[\vec{V}_i^t + c_1 \vec{U}_1^t \otimes (\vec{P}_i^t - \vec{X}_i^t) + c_2 \vec{U}_2^t \otimes (\vec{G}^t - \vec{X}_i^t) \right] \tag{2}$$

$$\vec{X}_i^{t+1} = \vec{X}_i^t + \vec{V}_i^{t+1} \tag{3}$$

where $i = 1, 2, \ldots, N$, with N equal to the swarm size, and $t = 1, 2, \ldots, T$, with T equal to the maximum number of iterations. ω is defined as the *inertia weight*, which was introduced in early PSO variants to avoid particles flying around their best-known position without converging to it. c_1 and c_2 are the *cognitive* and *social* acceleration coefficients, which control the influence of the *personal* and the *local best*. These two parameters influence the trade-off between the *local exploitation* and the *global exploration* search abilities of the algorithm (*Shi and Eberhart*, 1998). $\vec{U}_1$ and $\vec{U}_2$ are independent and uniformly distributed D-dimensional random vectors in the range $[0, 1]$ used to maintain the "diversity" of the population (note that $\otimes$ denotes element-wise vector multiplication). χ is the constriction factor defined by

$$\chi = \frac{2\kappa}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|} \tag{4}$$

where $\kappa \in [0, 1]$ and $\varphi = c_1 + c_2 > 4$, with typical values of $\kappa = 1$, $c_1 = c_2 = 2.05$ and $\varphi = 4.1$. It should be noted that equations 1 and 2 are mathematically equivalent for appropriate values of the coefficients ω, c_1, and c_2.

The inclusion of the inertia weight (ω) or the constriction factor (χ) in equations 1 or 2 aims to prevent *swarm explosion*, i.e. an uncontrolled increase of the magnitude of velocities (particle displacement) (*Poli et al.*, 2007). In addition to this, *Eberhart and Shi* (2000) suggest constraining particle velocity to within the range $[-\vec{V}^{max}, \vec{V}^{max}]$ with $\vec{V}^{max} = \vec{X}^{max}$, and $\vec{X}^{max}$ as the limits of the search space. In case this constraint is violated, the d-dimensional component of the velocity for the i-th particle is set directly to the closest velocity bound as follows

$$v_{id} = \begin{cases} v_d^{max} & , v_{id} > \quad v_d^{max} \\ -v_d^{max} & , v_{id} < \quad -v_d^{max} \end{cases} \tag{5}$$

Equations 1 to 5 can be considered as the *canonical* PSO algorithm, with the whole swarm as a single neighbourhood (see *Kennedy*, 2006), which is summarised as follows

Algorithm 1 Canonical PSO algorithm.

1: **for** $i = 1$ to N **do** {for each particle in the swarm}
2: Initialise particles' positions ($\vec{X}_i$) and velocities ($\vec{V}_i$)
3: Initialise personal best, $\vec{P}_i$, and local best, $\vec{G}$
4: **repeat**
5: **for** $i = 1$ to N **do**
6: Pick random vectors $\vec{U}_1$ and $\vec{U}_2$
7: Update particle's velocity using equations 1 or 2
8: Update particle's position using equation 3
9: **if** $f(\vec{X}_i) < f(\vec{P}_i)$ **then** {minimization of f}
10: Update particle's best-known position $\vec{P}_i = \vec{X}_i$
11: **if** $f(\vec{P}_i) < f(\vec{G})$ **then** {minimization of f}
12: Update the neighbourhood's best-known position $\vec{G} = \vec{P}_i$
13: **until** [nr. of iterations (T) or tolerance error is met]

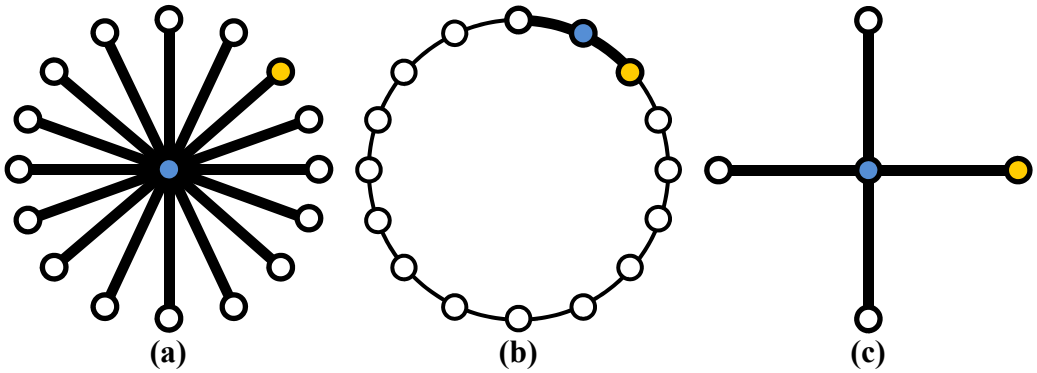

Figure 2: Examples of topologies for an i-th particle (blue) interacting with its neighbourhood and the local best (yellow): (a) *gbest*; (b) *lbest*; and (c) *von Neumann* (2D-projected).

2.2 Topologies

Particles in the swarm interact by defining a common set of links. These links control the exchange of information, i.e. they inform each particle of the best-known position of neighbouring particles, and are defined as the *swarm topology* (see Figure 2). The set of particles "informing" the i-th particle at the t-th iteration is termed the particle's *neighbourhood*, and includes the particle itself as a member (*Clerc*, 2007).

A vast number of swarm topologies has been investigated and reviewed in the literature (see, e.g., *Kennedy*, 1999; *Kennedy and Mendes*, 2002, 2003; *Mendes*, 2004; *Poli et al.*, 2007). Two of the most common correspond to the so-called *gbest* and *lbest*. In the *gbest* topology all particles are interconnected and, thus, the best-known particle influences all the remaining particles in the swarm. This topology is generally recognised to have a fast convergence but is highly vulnerable to sub-optimal solutions and premature convergence (see, e.g., *Mendes*, 2004; *Clerc*, 2006; *Poli et al.*, 2007). In the *lbest* topology, each particle is connected to two immediate neighbours and, thus, exchange of information with the best-known particle is restricted only to the particle's immediate neighbourhood. In general, the particle itself is included as a member of its neighbourhood (*Mendes*, 2004; *Clerc*, 2006). The *lbest* topology shows the advantage of allowing parallel searches in different regions of the search-space (*Poli et al.*, 2007), which results in a more thorough search strategy. Accordingly, *lbest* shows a slower convergence and less sensitivity to sub-optimal solutions compared to *gbest*.

Kennedy and Mendes (2002) investigate generalisations of the *lbest* topology. One of their main findings is the relative superiority of the *von Neu-*

mann structured neighbourhood, i.e. four neighbours. This topology is more densely connected than *lbest* but less densely connected than *gbest*, thus, it shows some parallelism with *lbest* but benefits from a larger neighbourhood (*Poli et al.*, 2007).

Finally, *Clerc* (2006) propose the so-called *random* topology. In this topology, each particle informs k particles (and itself) chosen randomly, with the parameter k usually set to 3 (*Clerc*, 2012). This results in each particle informing at least one (itself) and at most $k+1$ (including itself) particles, and being informed by any number of particles between 1 and N. The *random* topology can be considered as a special case of the *lbest* topology where particles have k *informants* most of the time and the "connections" between particles randomly change when the global optimum shows no improvement. The *random* topology is used in the "Standard PSO 2011" algorithm, which has become the benchmark in recent years.

2.3 PSO Variants and Fine-Tuning Options

Notwithstanding the inclusion of V_{max} and ω aimed at overcoming premature convergence to sub-optimal solutions, improvements to the canonical PSO are still an active area of research (*Poli et al.*, 2007). To date, a vast collection of enhancements/variants has been suggested in the literature (see, e.g., *Clerc*, 2006; *Kennedy*, 2006; *van den Bergh and Engelbrecht*, 2006; *Zhao*, 2006; *Poli et al.*, 2007; *Poli*, 2008; *Chen and Chi*, 2010), resulting in a large group of algorithms, which we cannot fully review or implement into a single piece of software.

This section describes the PSO variants and state-of-the-art enhancements implemented in the calibration engine of *hydroPSO*, which are further summarised in Table 2.

2.3.1 PSO Variants

In equations 1 and 2 only two effective sources of "influence" are considered for updating the particles' velocities, namely, the position of the particle's personal best, $\vec{P}$, and the position of the local best, $\vec{G}$. *Mendes et al.* (2004) propose a PSO variant, which they refer to as Fully Informed Particle Swarm (FIPS), where information drawn from all the particles in the neighbourhood (i.e. not only the best one) contributes to adjusting the particle's velocity. FIPS is implemented as follows,

$$\vec{V}_i^{t+1} = \chi \left[\vec{V}_i^t + \frac{1}{K_i} \sum_{n=1}^{K_i} \vec{U}_1^t(0,\varphi) \otimes (\vec{P}_{nbr_n}^t - \vec{X}_i^t) \right] \tag{6}$$

where K_i is the number of neighbours for the i-th particle, and $\vec{P}_{nbr_n}$ is the position of the particle's n-th neighbour. Additionally, *Mendes et al.* (2004) propose a second PSO variant, which they refer to as weighted FIPS (wFIPS). In wFIPS the contribution of each neighbour to the adjustment of a particle's velocity is weighted by the corresponding goodness-of-fit of its personal best. Performance of both FIPS and wFIPS is highly sensitive to the topology employed (*Poli et al.*, 2007), however, for topologies with few neighbours they outperform the canonical PSO algorithm (*Mendes et al.*, 2004; *Kennedy*, 2006).

A third PSO variant implemented in *hydroPSO* corresponds to the Improved Particle Swarm Optimisation (IPSO) by *Zhao* (2006). In IPSO the term representing the social influence (i.e., $\vec{G}^t - \vec{X}_i^t$, in equations 1 or 2) is enhanced by using information drawn from a subset containing the n_{gb} best performing particles in the neighbourhood. According to *Zhao* (2006) this variant follows the analogy where a group of leaders, i.e. the best performing particles for an iteration, could influence positively better decisions for society (swarm) compared to a case when a single leader is followed. IPSO is implemented as follows

$$\vec{V}_i^{t+1} = \omega \vec{V}_i^t + c_1 \vec{U}_1^t \otimes (\vec{P}_i^t - \vec{X}_i^t) + \sum_{j=1}^{n_{gb}} c_{2,j} \vec{U}_{2,j}^t \otimes (\vec{G}_j^t - \vec{X}_i^t) \tag{7}$$

where coefficient $c_{2,j}$ is defined by

$$c_{2,j} = c_2 \hat{g}_j \tag{8}$$

where $\hat{g}_j$ is a weighting factor based on the values of the goodness-of-fit for the n_{gb} particles.

2.3.2 Fine-Tuning Options

In this section several options used to tailor the calibration engine to specific user needs are described. We should note here that the effectiveness of these options has generally been tested against the "canonical" PSO algorithm.

1. PSO Parameters
   The PSO algorithm includes a few parameters that have to be carefully selected in order to avoid swarm explosion and to ensure proper convergence of the algorithm. These parameters include: a) inertia weight (ω) or the Clerc's constriction factor (χ), b) the cognitive acceleration coefficient (c_1), and c) the social coefficient (c_2).

Several definitions for the *inertia weight* (ω) have been extensively investigated and, hence, will not be further developed here. *Shi and Eberhart* (1998) initially propose a linearly decreasing variation for ω. This strategy forces an extensive exploration of the search-space at initial iterations, which gradually shifts to a more dissipative ("exploitative") search at later iterations (*Poli et al.*, 2007). Alternative ω definitions implemented in *hydroPSO* include: a) linear (*Shi and Eberhart*, 1998), b) non-linear (*Chatterjee and Siarry*, 2006), c) adaptive factor (*Liu et al.*, 2005), d) global-local best ratio (*Arumugam and Rao*, 2008), and e) fully random (*Eberhart and Shi*, 2001). These alternative definitions for ω allow the modeller to fine-tune the "exploitation/exploration" properties of the algorithm for different calibration problems.

Depending on the nature of the optimisation/calibration problem, the definition of the acceleration coefficients c_1 and c_2 may have a drastic impact on the algorithm's performance. *Ratnaweera et al.* (2004) suggest that time-varying acceleration coefficients (TVAC) may reduce premature convergence at the first stages of the search and improve convergence at later iterations. For example, a large value of c_1 and a small value of c_2 at initial iterations allow the particles to "fly" around the search-space instead of moving towards their personal best. For later iterations, in turn, a small value of c_1 and a large value of c_2 allow the particles to converge to the global attraction zone. Based on this, linear, non-linear, and global-local best ratio (only for c_1) TVAC are implemented in *hydroPSO* to fine-tune the coefficients c_1 and c_2.

2. Boundary Conditions
   In many optimisation/calibration problems it is a good practice to confine particle position to physically meaningful parameter ranges. In *hydroPSO* this is done by using a D-dimensional vector defining the range of the search-space ($\Psi = \{(\psi_1^{min}, \psi_1^{max}), (\psi_2^{min}, \psi_2^{max}), \ldots, (\psi_D^{min}, \psi_D^{max})\}$). The behaviour of the particles at these boundaries may influence the exploration of the search-space (*Robinson and Rahmat-Samii*, 2004). Generally, four types of boundary condition may be imposed on a particle reaching the boundary of the search-space: a) *absorbing*, where the particle's position is set to the boundary value and the velocity is set to zero; b) *invisible*, where the particle is allowed to "fly" beyond the boundaries without limiting either its position or velocity, but its goodness-of-fit is not evaluated, c) *reflecting*, where the sign of the velocity in the overshot dimension is changed and the particle is reflected back to the search-space; and d) *damping*, where part (1-f) of the veloc-

ity is absorbed by the boundary and part (f) is reflected back, with f as a random damping factor in [0,1] (see *Robinson and Rahmat-Samii*, 2004; *Huang and Mohan*, 2005; *Clerc*, 2012).

3. Regrouping Strategy
   In the case of premature convergence (i.e. when the proposed solution approximates a local rather than a global optimum) or to validate the global attraction zone at final iterations, a suitable mechanism allowing particles to escape from sub-optimal solutions is required. *Evers and Ghalia* (2009) propose a strategy that avoids stagnation of particles by automatically triggering swarm regrouping (see Table 2 for details). At each iteration the *swarm radius* (δ^t) is computed as the maximum Euclidean distance between the position of the swarm optimum ($\vec{G}$) and any given particle ($\vec{X}_i$). Then, regrouping is triggered when the *normalised swarm radius* (δ^t_{norm}) is less than a user-defined *stagnation threshold* (ϵ). The swarm is regrouped around the swarm optimum by using an updated range ($range_d(\Psi^r)$ where r is the *swarm regrouping index*) calculated as the minimum between (i) the original range of the search-space on dimension d, and (ii) the product of the *regrouping factor* (ρ) with the maximum distance along dimension d between any particle and the swarm optimum. This regrouping method defines a reduced search-space (Ψ^r), which is small enough for an efficient search and large enough to allow particles to escape from sub-optimal solutions.

4. Other Options
   In *hydroPSO* sampling of initial particle position ($\vec{X}_{ini}$) and velocity ($\vec{V}_{ini}$) might be improved by including a Latin Hypercube sampling (*McKay et al.*, 1979) over the full D-dimensional search-space in order to ensure a fair initialisation of the D exploration.

   To enhance a faster propagation of the best solutions found throughout the swarm, *Mussi et al.* (2009) discuss two strategies for updating the particle and local best position. In the *synchronous* strategy the local best is updated only after computing and updating the position and personal best of all particles in the swarm. In the *asynchronous* strategy, the local best is updated immediately after each particle's position is updated.

   For later stage iterations, oscillation of the particles around the local best may be restricted by using a time-varying $\vec{V}_{max}$. This reduces the search-space for latter stage iterations and, thus, complements the

effect of (ω). This is suggested by the authors of *hydroPSO* following *Chatterjee and Siarry* (2006).

Finally, *hydroPSO* (>=0.3-0) offers **parallel capabilities** to speed up the optimisation process in multi-core machines or cluster networks, which are easily configured.

2.4 Standard PSO 2011 (SPSO 2011)

In a series of technical reports (see *Clerc*, 2007, 2009, 2012), a clear need to define a "Standard PSO algorithm" has been identified. This stems mainly from the need to establish a common benchmark to assess the performance of the numerous PSO improvements proposed in the literature. To date, three versions of the standard PSO algorithm exist: SPSO 2006, SPSO 2007, and SPSO 2011 (*Clerc*, 2012). We should point out that SPSO 2011 has in fact been designed as a benchmark for algorithm performance and is not intended to be the best available PSO algorithm in the literature (*Clerc*, 2012).

The most recent benchmark proposed, SPSO 2011, starts by defining a centre of gravity (Gr) around the current ($\vec{X}_i^t$), previous best ($\vec{p}_i^t$), and local best positions ($\vec{l}_i^t$) as follows,

$$\vec{p}_i^t = \vec{X}_i^t + c_1 \vec{U}_1^t \otimes (\vec{P}_i^t - \vec{X}_i^t) \tag{9a}$$

$$\vec{l}_i^t = \vec{X}_i^t + c_2 \vec{U}_2^t \otimes (\vec{G}^t - \vec{X}_i^t) \tag{9b}$$

$$\vec{Gr}_i^{\,t} = \frac{(\vec{X}_i^t + \vec{p}_i^t + \vec{l}_i^t)}{3} \tag{9c}$$

then, a random point is defined in the hypersphere $\mathcal{H}(\vec{Gr}_i^{\,t}, ||\vec{Gr}_i^{\,t} - \vec{X}_i^t||)$, and velocity is updated as follows,

$$\vec{V}_i^{t+1} = \omega \vec{V}_i^t + r\mathcal{H}(\vec{Gr}_i^{\,t}, ||\vec{Gr}_i^{\,t} - \vec{X}_i^t||) - \vec{X}_i^t \tag{10}$$

where r is a random number in $[0, 1]$. The particle's position is updated following equation (3).

Despite the fact that SPSO 2011 is an improvement to the previous SPSO 2007 in terms of accounting for rotational invariance, SPSO 2007 could perform better than SPSO 2011 for separable and (less likely) for non-separable functions (Maurice Clerc, developer SPSO 2007 & 2011, personal communication, 2012). This is the main reason for including both SPSO 2007 & SPSO 2011 in the *hydroPSO* R package.

Table 2: Main features for customising the *hydroPSO* calibration engine.

Feature	*hydroPSO* argument	Description	References
1. PSO Variants	`method="spso2011"`	1.1. Standard PSO 2011	*Clerc* (2012)
	`method="spso2007"`	1.2. Standard PSO 2007	*Clerc* (2012)
	`method="fips"`	1.3. Fully Informed Particle Swarm	*Mendes et al.* (2004)
	`method="wfips"`	1.4. Weighted FIPS	*Mendes et al.* (2004)
	`method="ipso"`	1.5. Improved PSO (IPSO)	*Zhao* (2006)
	`method="canonical"`	1.6. Canonical PSO	*Clerc* (2007)
2. Topologies	`topology="random"`	2.1. Random ($k = 3$)	*Clerc* (2012)
	`topology="lbest"`	2.2. *lbest*	*Kennedy and Mendes* (2002)
	`topology="vonNeumann"`	2.3. von Neumann ($k = 4$)	*Kennedy and Mendes* (2002)
	`topology="gbest"`	2.4. *gbest*	*Kennedy and Mendes* (2002)
3. Boundary conditions	`boundary.wall="absorbing2011"`	3.1. Absorbing	*Clerc* (2012)
	`boundary.wall="absorbing2007"`	3.2. Absorbing	*Clerc* (2012)
	`boundary.wall="reflecting"`	3.3. Reflecting	*Robinson and Rahmat-Samii* (2004)
	`boundary.wall="invisible"`	3.4. Invisible	*Robinson and Rahmat-Samii* (2004)
	`boundary.wall="damping"`	3.5. Damping	*Huang and Mohan* (2005)
4. Definition of X_{ini}	`Xini.type="random"`	4.1. Random	*Eberhart and Kennedy* (1995)
	`Xini.type="lhs"`	4.2. Latin Hypercube Sampling (LHS)	*hydroPSO*
5. Definition of V_{ini}	`Vini.type="random2011"`	5.1. Random following SPSO 2011	*Clerc* (2012)
	`Vini.type="lhs2011"`	5.2. LHS following SPSO 2011	*Clerc* (2012)
	`Vini.type="random2007"`	5.3. Random following SPSO 2007	*Clerc* (2012)
	`Vini.type="lhs2007"`	5.4. LHS following SPSO 2007	*Clerc* (2012)
	`Vini.type="zero"`	5.5. Zero velocity	*hydroPSO*
6. Definition of inertia weight ω	`use.IW=TRUE, IW.w=1/(2*log(2))`	6.1. Linear variation	
	`IW.type="linear"`	$\omega_{iter} = \left[\frac{iter_{max}-iter}{iter_{max}}\right](\omega_{ini}-\omega_{fin})+\omega_{fin}$ [a]	*Shi and Eberhart* (1998)
		6.2. Non-linear variation	
	`IW.type="non-linear"`	$\omega_{iter} = \left[\frac{iter_{max}-iter}{iter_{max}}\right]^n(\omega_{ini}-\omega_{fin})+\omega_{fin}$ [a]	*Chatterjee and Siarry* (2006)
		6.3. Adaptive Inertia Weight Factor (AIWF)	
	`IW.type="aiwf"`	$\omega_{iter} = \begin{cases} \left[\frac{(\omega_{max}-\omega_{min})(f-f_{min})}{f_{avg}-f_{min}}\right]+\omega_{min} & , f \le f_{avg} \\ \omega_{max} & , f > f_{avg} \end{cases}$ [b]	*Liu et al.* (2005)
		6.4. Global-Local best ratio inertia weight	
	`IW.type="GLratio"`	$\omega_{iter} = 1.1 + \frac{G_d^t}{P_{id}^t}$ [c]	*Arumugam and Rao* (2008)
		6.5. Random inertia weight	
	`IW.type="runif"`	$\omega_{iter} = 0.5 + \frac{rnd(\omega_{ini},\omega_{fin})}{2}$ [d]	*Eberhart and Shi* (2001)
7. Time-varying acceleration coefficients	`use.TVc1=FALSE, c1=0.5log(2)+`	7.1.1. Linear variation	
	`TVc1.type="linear"`	$c1_{iter} = \left[\frac{iter}{iter_{max}}(c1_{fin}-c1_{ini})\right]+c1_{ini}$ [e]	*Ratnaweera et al.* (2004)
	`TVc1.type="non-linear"`	7.1.2. Non-linear variation	
		$c1_{iter} = \left[\frac{iter}{iter_{max}}(c1_{fin}-c1_{ini})\right]^n+c1_{ini}$ [e]	*hydroPSO*
	`TVc1.type="GLratio"`	7.1.3. Global-Local best ratio	
		$c1_{iter} = 1.1 + \frac{G_d^t}{P_{id}^t}$	*Arumugam and Rao* (2008)
	`use.TVc2=FALSE, c2=0.5log(2)+`	7.2.1. Linear variation	
	`TVc2.type="linear"`	$c2_{iter} = \left[\frac{iter}{iter_{max}}(c2_{fin}-c2_{ini})\right]+c2_{ini}$ [e]	*Ratnaweera et al.* (2004)
		7.2.2. Non-linear variation	
	`TVc2.type="non-linear"`	$c2_{iter} = \left[\frac{iter}{iter_{max}}(c2_{fin}-c2_{ini})\right]+c2_{ini}$ [e]	*Ratnaweera et al.* (2004)
8. Time-varying maximum velocity	`use.TVlambda=FALSE`	$\vec{V}_{max}^{iter} = \lambda^{iter}\vec{X}_{max}$	
		8.1. Linear variation	
	`TVlambda.type="linear"`	$\lambda^{iter} = \left[\frac{iter}{iter_{max}}(\lambda^{fin}-\lambda^{ini})\right]+\lambda^{ini}$ [g]	*Chatterjee and Siarry* (2006)
		8.2. Non-linear variation	
	`TVlambda.type="non-linear"`	$\lambda^{iter} = \left[\frac{iter}{iter_{max}}(\lambda^{fin}-\lambda^{ini})\right]^n+\lambda^{ini}$ [g]	*hydroPSO*
9. Regrouping	`use.RG=FALSE`	9.1. *swarm radius* $= \delta^t = max\|\vec{X}_i^t - \vec{G}^t\|$ [f]	*Evers and Ghalia* (2009)
	`RG.thr=`ϵ	*normalised swarm radius* $= \delta_{norm}^t = \frac{\delta^t}{diam(\Psi)} < \epsilon$	
		$diam(\Psi) = \|range_d(\Psi)\|$	
		$range_d(\Psi) = X_d^U - X_d^L$	
	`RG.r=`ρ	$range_d(\Psi^r) = min\left\{range_d(\Psi^0), \rho\ max\|X_{id}^{r-1} - G_d^{r-1}\|\right\}$	
	`RG.miniter=`r	$\vec{X}_i = G^{r-1} + \left[\vec{r_3} \cdot range_i(\Psi^r) - \frac{1}{2}range_i(\Psi^r)\right]$	
		$V_d^{max} = \lambda range_d(\Psi^r)$	
10. Update of positions and velocities	`best.update="sync"`	10.1. Synchronous	*Mussi et al.* (2009)
	`best.update="async"`	10.2. Asynchronous	*Mussi et al.* (2009)
11. Normalisation	`normalise=FALSE`	11.1. Standard PSO 2011	*Clerc* (2012)
12. Parallelisation	`parallel="none"`		*hydroPSO*
	`par.nnodes=NA`		*hydroPSO*
	`par.pkgs=c()`		*hydroPSO*

[a] ω_{iter} is the current inertia weight, ω_{ini} and ω_{fin} are the initial and final inertia weights, and n is the non-linear modulation index.

[b] f is the current objective value of the particle, f_{avg} is the average objective value for all the particles, f_{min} is the minimum objective value of all the particles, ω_{max} and ω_{min} are user-defined maximum and minimum values for ω.

[c] It sets a minimisation problem where G_d^t and $\overline{P_{id}^t}$ are the local best and the average of all personal bests during iteration t, respectively.

[d] $rnd()$ is a random number in $[\omega_{ini}, \omega_{fin}]$.

[e] $c1_{ini}$, $c1_{fin}$, $c2_{ini}$, $c2_{fin}$ are constants.

[f] $\vec{X}_i^t$ are particles' positions, $\vec{G}^t$ is local best location, ϵ is the stagnation threshold, $diam(\Psi)$ is the diameter of the search-space, $range_d(\Psi)$ is the length of the search-space along dimension d, X_d^U and X_d^L are the upper and lower bounds of the search-space, Ψ^r is the regrouped search-space, r is the *swarm regrouping index*, Ψ^0 is the original search-space, ρ is the regrouping factor, $\vec{r_3}$ is a uniform random vector, and λ is a percentage to limit $\vec{V}_{max}$.

[g] λ^{iter} is a percentage to limit $\vec{V}_{max}$, and λ^{ini} and λ^{fin} are the percentage to limit $\vec{V}_{max}$ at the start and end of a given run, respectively.

3 Description of the hydroPSO Package

hydroPSO is a multi-platform and model-independent R package that has been designed to allow the user to perform sensitivity analysis, model calibration, and assessment of the results. Sensitivity analysis is performed by using the LH-OAT method (*van Griensven et al.*, 2006). Sensitive parameters are then calibrated by tailoring the calibration engine to specific user needs through the use of the numerous PSO variants and fine-tuning options described in Section 2.3. Finally, advanced plotting functionalities together with detailed information regarding the evolution of the algorithm's performance facilitate the interpretation and assessment of the calibration results.

3.1 Main hydroPSO Functions

The interaction among the main functions comprising the *hydroPSO* package is shown in Figure 3. These functions have been specifically developed for *hydroPSO* and are summarised in Table 3 and further described below.

Table 3: Main functions included in the *hydroPSO* R package.

Function	Short Description
`lhoat()`	Sensitivity analysis using LH-OAT (*van Griensven et al.*, 2006)
`hydromod()`	Control of the model code to be calibrated
`hydroPSO()`	Multi-platform and model-independent enhanced PSO calibration engine
`read_results()`	Reading of results produced by *hydroPSO*. It is a wrapper to:
`read_particles()`	Reading *"Particles.txt"* output file
`read_velocities()`	Reading *"Velocities.txt"* output file
`read_out()`	Reading *"Model_out.txt"* output file
`read_convergence()`	Reading *"ConvergenceMeasures.txt"* output file
`read_GofPerParticle()`	Reading *"Particles_GofPerIter.txt"* output file
`plot_results()`	Plotting of results produced by *hydroPSO*. It is a wrapper to:
`plot_particles()`	Plotting parameters
`plot_velocities()`	Plotting the evolution of particles' velocities
`plot_out()`	Plotting model outputs (simulated equivalents)
`plot_convergence()`	Plotting the evolution of the global optimum and δ^t_{norm}
`plot_GofPerParticle()`	Plotting the evolution of the goodness-of-fit per particle
`verification()`	Run the model code with one or more parameter sets specified by the user
`PEST2hydroPSO`	Import PEST (*Doherty*, 2010) files into input files for *hydroPSO*
`hydroPSO2PEST`	Export *hydroPSO* files into input files for PEST
`test_functions()`	Implementation of a set of n-dimensional test functions for benchmarking

1. The `lhoat()` function implements the Latin Hypercube One-factor-At-a-Time (LH-OAT) sensitivity analysis technique (*van Griensven et al.*, 2006). `lhoat()` produces a ranking with the parameter having the largest effect receiving a rank of 1 and the one(s) with the smallest effect receiving a rank equal to the total number of parameters (D). LH-OAT works by taking M LH sampling points (M strata for each

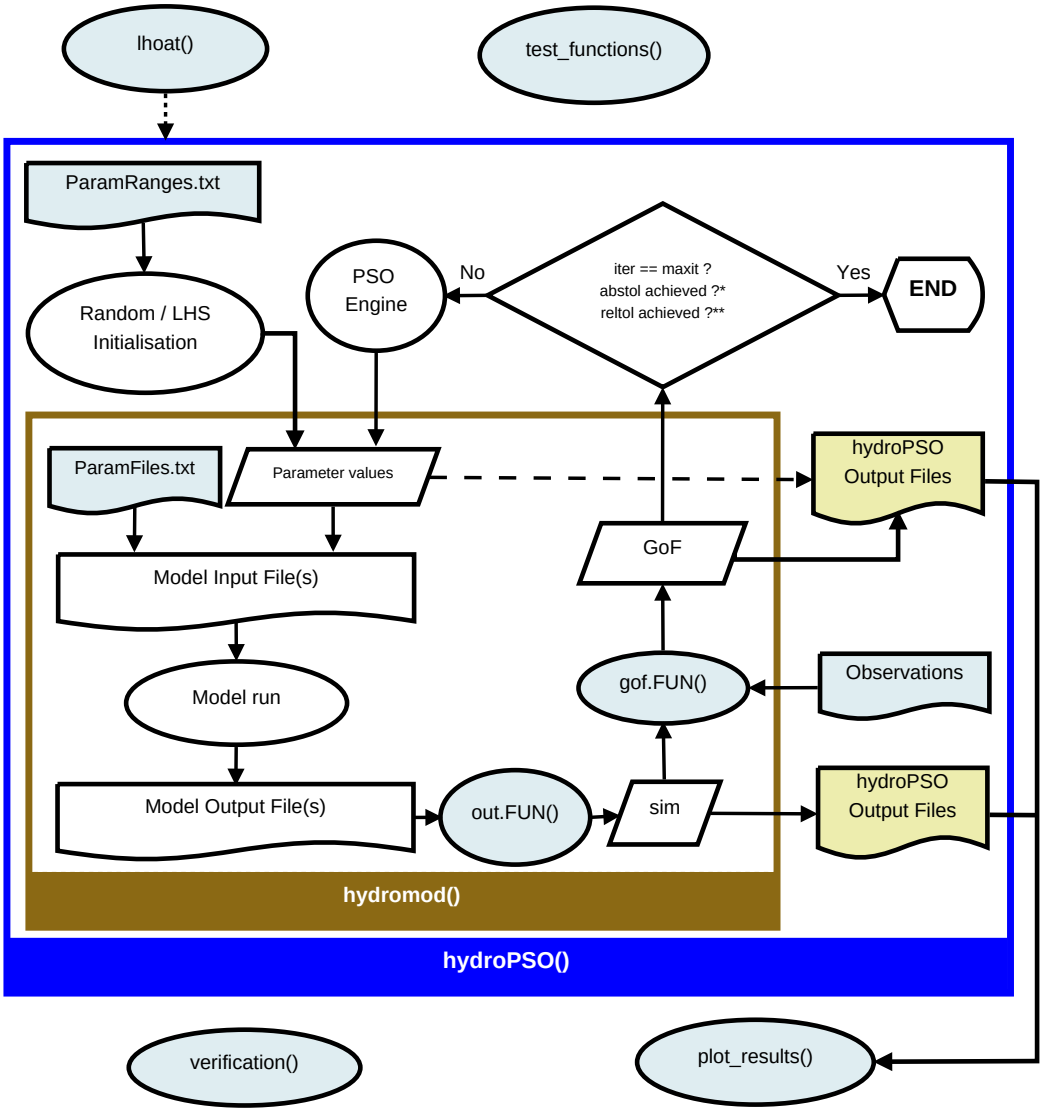

Figure 3: Flowchart describing the interaction of the main hydroPSO functions. User-defined files `ParamRanges.txt` and `ParamFiles.txt` provide information on the parameters to be calibrated, whereas `out.FUN()`, `gof.FUN()`, and observations are used to assess the quality of the particles' positions through a goodness-of-fit measure. Light-blue shaded boxes require user intervention (after *Zambrano-Bigiarini and Rojas*, 2012).

parameter) and then varying, by a fraction s, each LH sampling point D times, where D is the number of parameters. For each LH sampling point a partial impact for each parameter is calculated and then a final impact (on model performance/predictions) is obtained by averaging

these partial impacts. The method is very efficient requiring a total of only $M(D+1)$ runs.

2. The `hydromod()` function is one of the key components of *hydroPSO*. It has a unique role consisting of linking and controlling the execution of the user-defined model code with the optimisation engine. `hydromod()` first reads a set of parameter values, which are then written into the corresponding model input file(s) by using the information provided in the user-defined `ParamFiles.txt` file. `ParamFiles.txt` defines the name of the model parameters, input files, and specific location of the parameters in the input files. Unlike tools such as PEST (*Doherty*, 2010), UCODE-2005 (*Poeter et al.*, 2005), OSTRICH (*Matott*, 2005) or MADS (*Vesselinov and Harp*, 2012), only one file is required to define *where* each parameter is located, with no need for creating numerous *template* files replicating model input files. Then, `hydromod()` executes the model code to obtain the corresponding model outputs, which are read through the `out.FUN()` R function. Finally, simulated equivalents are compared against user-provided observations of system state variables through a `gof.FUN()` R function. `hydromod()` returns the simulated equivalents and the corresponding goodness-of-fit values.

3. The main calibration engine is implemented in the `hydroPSO()` function. This function is another key component of *hydroPSO* that includes state-of-the-art PSO variants and numerous fine-tuning options to customise the calibration engine to specific user needs (see Table 2). At the first iteration, parameter sets are sampled from a feasible range obtained from the user-defined `ParamRanges.txt` file. Then, `hydromod()` is called to obtain a goodness-of-fit measure for each particle in the swarm, i.e. each parameter set. Particle velocity and position evolve according to the user-defined PSO configuration until some termination criteria are met (e.g. maximum number of iterations (`maxiter`), relative changes in the personal best below a user-defined threshold (`reltol`), or global optimum is below a user-defined threshold (`abstol`)). `hydroPSO()` returns the optimum parameter set, all sampled parameters and their corresponding goodness-of-fit, model outputs, particle velocity, and convergence measures.

4. Results from the `hydroPSO()` function are post-processed using the `plot_results()` function, which delivers a series of user-friendly and customised plots to facilitate the assessment of the model calibration. These plots include: parameter scatter plots, histograms and empirical CDFs, (pseudo) 3D scatter plots, parameters and velocities for each

iteration, goodness-of-fit value for each particle per iteration, and the evolution of the convergence measures.

5. The `verification()` function is designed to validate (one or more) user-defined parameter sets. It returns a goodness-of-fit measure for each parameter set defined, the best parameter set, and the goodness-of-fit measure corresponding to the best parameter set.

6. Function `PEST2hydroPSO` translates existing files generated to run a PEST model calibration (i.e. *.tpl, *.pst, and *.ins) into the basic files to set up a *hydroPSO* optimisation (i.e. `ParamFiles.txt`, `ParamRanges.txt` and `hydroPSO-script.R`).

7. Function `hydroPSO2PEST` translates the *hydroPSO* input files into the basic files required for performing a PEST local-calibration. This function is particularly useful for obtaining detailed information on sensitivity and uncertainty of parameters once a global optimisation has been performed using *hydroPSO*. In particular, this function would allow linking *hydroPSO* with tools using PEST-based templates such as OSTRICH, UCODE-2005, and squads/MADS.

8. The *hydroPSO* package includes a series of n-dimensional functions commonly used as benchmarks to assess the performance of optimisation algorithms. These functions are described in the `test_functions()` function.

3.2 `ParamFiles.txt` and `ParamRanges.txt`

For the case when *hydroPSO* is used to optimise any R function, the user only needs to provide suitable values for the `lower` and `upper` arguments, which define the minimum and maximum possible values for each parameter. The length of `lower` and `upper` is used to define the dimension (D) of the optimisation problem, with no need of any additional input file.

For the case when *hydroPSO* is used to calibrate an external model code (i.e. not an R function but an executable file), two input files are required to interface *hydroPSO* with the corresponding model, namely, `ParamFiles.txt` and `ParamRanges.txt`. These two files are simple ASCII files used to define *which* model parameters need to be calibrated and *where* they need to be written, and they should suffice for basic applications. For more advanced applications, *hydroPSO* may require the definition of additional R wrapper functions, mainly for reading model outputs (`out.FUN`), and for computing

model's performance (`gof.FUN`). In sections 5.2 and 6.4 we provide examples of both cases.

`ParamFiles.txt` file defines the exact location of the parameters to be calibrated in the model input files. The format of `ParamFiles.txt` is as follows,

ParamFiles.txt

ParameterNmbr	ParameterName	Filename	Row.Number	Col.Start	Col.End	DecimalPlaces
1	par1	file1	x1	y1	z1	w1
2	par2	file1	x2	y2	z2	w2
3	par3	file2	x3	y3	z3	w3
4	par4	file3	x4	y4	z4	w4
1	par1	file4	x5	y5	z5	w5
1	par1	file5	x5	y5	z5	w5
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
n	parn	filem	xn	yn	zn	wn

where `ParameterNmbr` is a consecutive number assigned to each parameter, `ParameterName` is a user-defined parameter name, `Filename` is the name of the model input file where the corresponding `ParameterName` is located, `Row.Number` is the row number in the `Filename` file where `ParameterName` can be found, `Col.Start` and `Col.End` define the column numbers in the `Filename` file where `ParameterName` have to be written, and `DecimalPlaces` defines the decimal places used for writing the corresponding `ParameterName` value. Please note that the same file (`file1`) can include more than one parameter (`par1` and `par2`) located in different places in the file, and also that the same parameter (`par1`) can appear in different input files (`file1`, `file4`, `file5`). In addition, it is very important to define `Col.Start` and `Col.End` in such a way that the parameter value AND its decimal places fit the width defined as `Col.Start` - `Col.End` + 1. For users familiar with PEST (*Doherty*, 2010), `Col.Start` and `Col.End` are equivalent to the token used to define *where* to write parameters during the optimisation.

`ParamRanges.txt` file defines the feasible range for each parameter to be calibrated. The format of `ParamRanges.txt` is as follows,

ParamRanges-Sens.txt

ParameterNmbr	ParameterName	MinValue	MaxValue
1	par1	min1	max1
2	par2	min2	max2
3	par3	min3	max3
4	par4	min4	max4
5	par5	min5	max5
.	.	.	.
.	.	.	.
.	.	.	.
n	parn	minn	maxn

where `MinValue` and `MaxValue` are the minimum and maximum parameter values, respectively. Note that care must be taken in numbering and naming the parameters to be calibrated as they require to be consistent in both files.

4 Step-by-Step *hydroPSO* Application

4.1 Setting Up the Environment

1. In the R console, use the next commands to retrieve the current working directory, set a new working directory, and list the files included in the new working directory, respectively.

```
> getwd()
> setwd("~")
> list.files(".")
```

2. Installing the latest stable version of *hydroPSO*.

```
> install.packages("hydroPSO")
```

3. Loading the *hydroPSO* library containing functions, help pages, and examples used in this analysis.

```
> library(hydroPSO)
```

4.2 Basic *hydroPSO* Application

This section illustrates the implementation of the *hydroPSO* package to optimise two (highly) multi-modal and multi-dimensional test functions with a considerable number of local sub-optimal solutions. These functions are commonly used as benchmarking for the performance of optimisation algorithms. Otherwise stated, the optimal solution for these functions is zero.

In this section we do not aim at finding an "optimum" configuration for the *hydroPSO* parameters, which in itself can be a massive task. Instead, we illustrate a few controlling options to handle problems such as premature convergence for (highly) multi-modal objective functions. We refer the reader to *Zambrano-Bigiarini and Rojas* (2012) for the results of a validation of *hydroPSO* against the Standard PSO 2011 (SPSO 2011) (*Clerc*, 2012).

4.2.1 Optimisation of Ackley Function

1. The **Ackley** function is a multi-modal and separable function with several local optima located at regular intervals and a global optimum at `o=(0,...,0)`. Complexity of the Ackley function is moderated resembling a noisy pattern in the range [-32;32]. The Ackley function is defined by:

$$f(x) = 20 + \exp(1) - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{2}\sum_{i=1}^{n}\cos(2\pi x_i)\right);$$
$$-32.7 \leq x_i \leq 32.7\ ;\ i = 1, 2, \ldots, n. \tag{11}$$

2. For optimising the Ackley function we need to define the upper and lower limits of the search space [-32;32] and its dimensionality (D=10). *hydroPSO* default values are used for the optimisation (`method="spso2011"`, `npart=40`, `maxit=1000`):

```
> D <- 10
> lower <- rep(-32,D)
> upper <- rep(32,D)
> set.seed(1111)
> hydroPSO(fn=ackley,lower=lower,upper=upper)

$par
       Param1        Param2        Param3        Param4
-2.653285e-06  7.559908e-07 -3.300330e-07 -4.472642e-08
       Param5        Param6        Param7        Param8
-1.556062e-06 -1.537707e-06 -5.336524e-07  7.740605e-07
       Param9       Param10
-6.617440e-07  9.748457e-07

$value
[1] 4.86307e-06

$best.particle
[1] 8

$counts
function.calls     iterations    regroupings
          9200            230              0

$convergence
[1] 1

$message
```

```
[1] "Converged ('reltol' criterion)"
```

Solution for the optimisation is reached at iteration 230 (9200 function calls (40*230)), with an optimum value of `4.863E-06`.

In the previous example, the *hydroPSO* finished before reaching the maximum number of iterations (by default `maxit=1000`) because the relative tolerance (`reltol`) was met. `reltol` is defined (by default) on the basis of the precision of the computational platform employed as `sqrt(.Machine$double.eps)`, which is typically in the order of `1E-8`.

3. Using less particles (i.e. less function evaluations) to get a global optimum similar to the previous one. For this case, 20 particles.

```
> set.seed(1111)
> hydroPSO(fn=ackley,lower=lower,upper=upper,
+          control=list(npart=20))

$par
       Param1        Param2        Param3        Param4
 1.230005e-06  2.246731e-07 -3.501479e-06  2.378093e-06
       Param5        Param6        Param7        Param8
-2.626830e-06 -2.202960e-06  1.924229e-06  1.449411e-06
       Param9       Param10
-1.369645e-06  3.520221e-06

$value
[1] 9.0604e-06

$best.particle
[1] 15

$counts
function.calls     iterations    regroupings
          5040            252              0

$convergence
[1] 1

$message
[1] "Converged ('reltol' criterion)"
```

For this case the solution for the optimisation is reached at iteration 252 with an optimum value of `9.06E-06`, but only with 5040 function calls.

4. Plotting the results:

```
> plot_results(do.png=TRUE, MinMax="min", do.pairs=TRUE)
```

By default `plot_results` plots directly to the R graphical device. Using the option `do.png=TRUE` all the the graphical outputs are re-directed to `png` files stored in `./PSO.out/pngs/`. Setting `do.pairs=TRUE` allows the creation of a matrix plot summarizing the interaction among parameters (cross-correlation, histograms, and statistical significance of the correlation). Results produced are shown in Figures 4-13.

Figure 4 shows the evolution of the Global Optimum and the Normalised Swarm Radius (NSR). The latter indicates the convergence of the swarm to the (optimum) attraction zone. Assessment of both Global Optimum and NSR is particularly useful when the regrouping strategy used to tackle premature convergence is activated (`use.RG=TRUE`).

Figure 5 shows dotty-plots for each of the 10 parameters, where each dot represents an evaluation of the function (model) being optimised. By default, dotty-plots are shown for all parameter sets, however, and as seen from Figure 5, not all of them may show a good performance. At the same time, Figure 6 shows boxplots summarizing each parameter. Using the options `beh.thr` and `MinMax`, a "behavioural" threshold is defined to select a sub-set of the best performing parameters. By defining `beh.thr` we discard parameters showing a poor performance and, as such, this shows some parallelism with the specification of the rejection threshold in Generalized Likelihood Uncertainty Estimation (GLUE) (see *Beven*, 2006) or the definition of the burn-in samples in Markov Chain Monte Carlo (MCMC) simulation (see *Gelman et al.*, 2004).

Interaction between parameters is shown in Figure 7. This figure shows the interaction between the first 5 parameters (selected by default by *hydroPSO*) as a function of the goodness-of-fit. Using `dp3D.names` the user can select specific parameters to plot. If the option `beh.thr` is specified, this figure shows the sub-set representing only the best-performance particles.

Figure 8 shows a (matrix) summary of the interaction among parameters. Here, the statistical significance of the correlation as well as dispersion-like and histogram plot summaries are obtained.

Empirical Cumulative Distribution Functions (ECDFs) for each parameter are shown in Figure 9. ECDFs are created for all parameter sets evaluated unless the `beh.thr` option is specified. In the last case, ECDFs are built from the sub-set representing the best performance.

Figure 10 shows the histograms of parameters. As in Figures 7 and 9, these histograms are calculated for all parameter sets retained unless the `beh.thr` option is specified.

Finally, Figures 11, 12 and 13 provide detailed information about the evolution of the goodness-of-fit function per iteration for each particle defined in the swarm, and particles' positions and velocities versus function (model) evaluation, respectively.

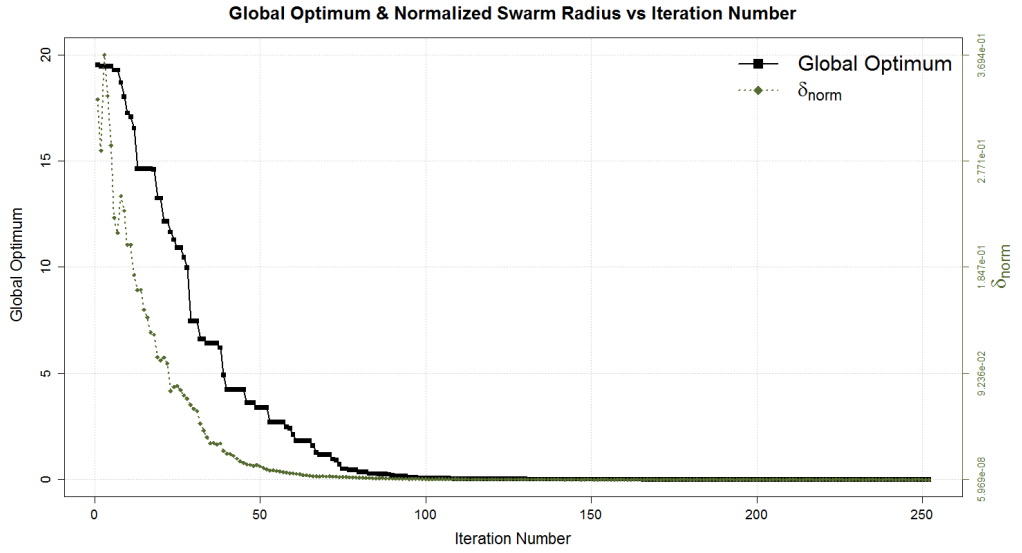

Figure 4: Global Optimum and Normalised Swarm Radius versus iteration number.

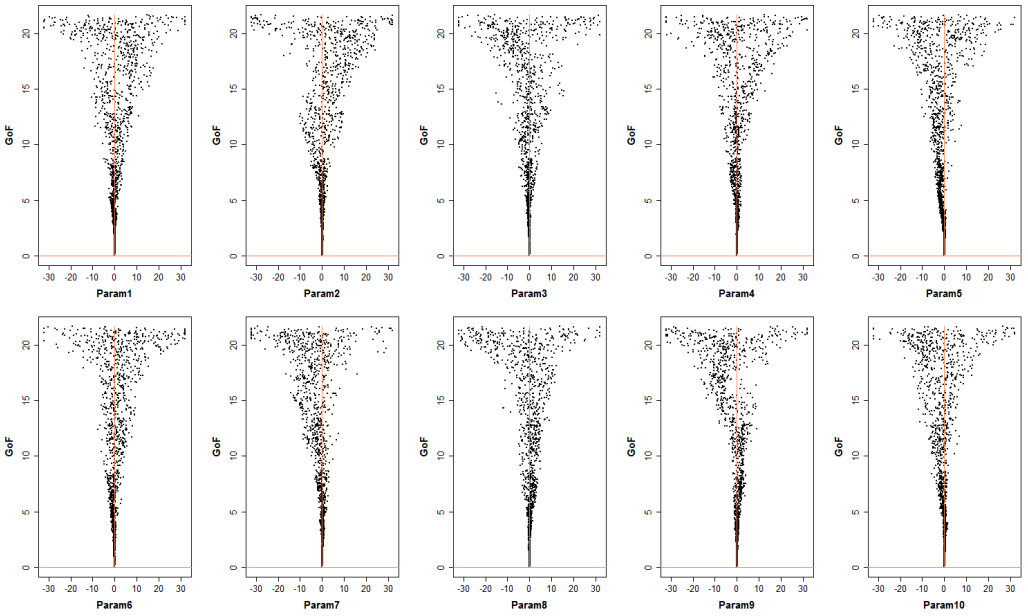

Figure 5: Dotty-plots for each parameter.

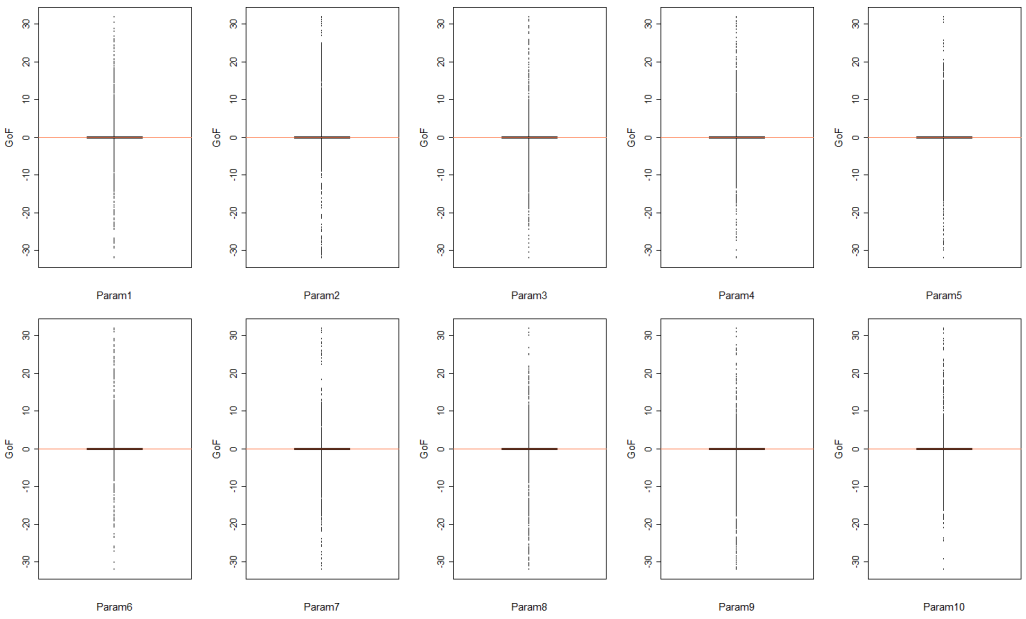

Figure 6: Box-plots for each parameter.

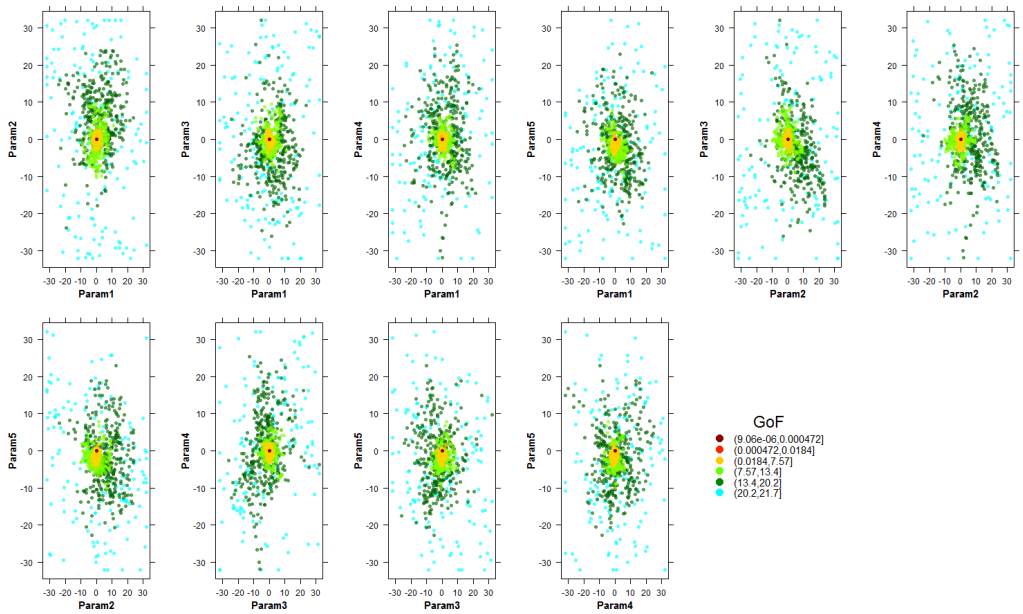

Figure 7: 2-dimensional projected dotty-plots highlighting the interaction between parameters.

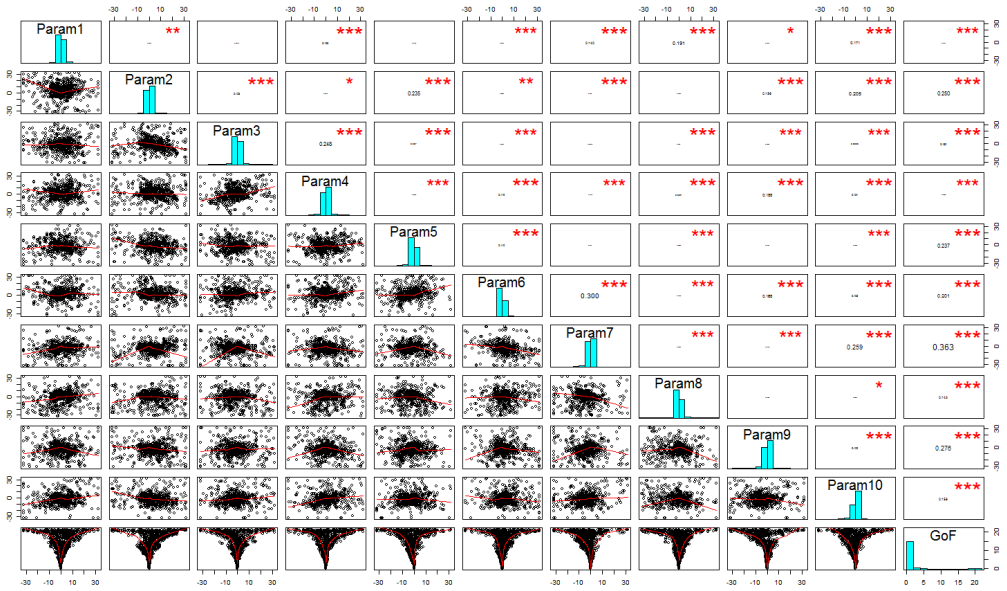

Figure 8: Matrix summarizing the interaction among parameters (e.g. cross correlation, histograms, and statistical significance of the correlation).

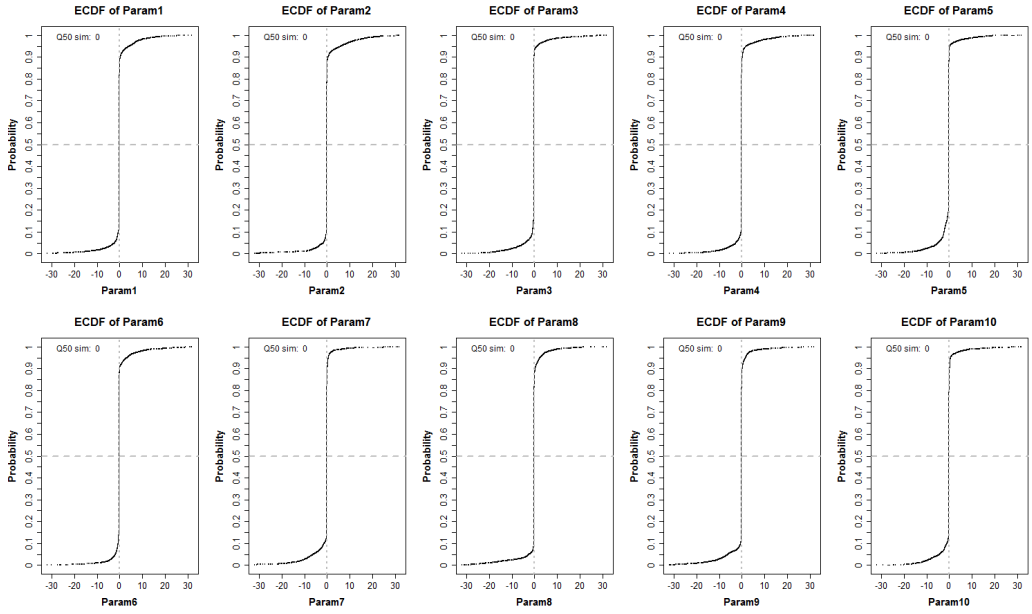

Figure 9: Empirical Cumulative Distribution Functions (ECDFs) for parameters.

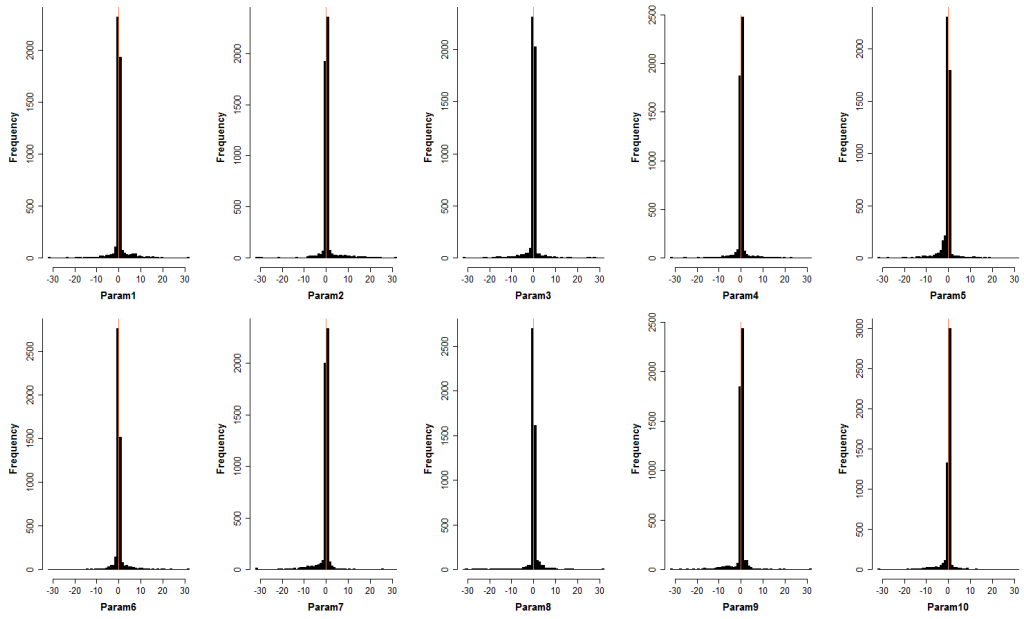

Figure 10: Histograms for parameters.

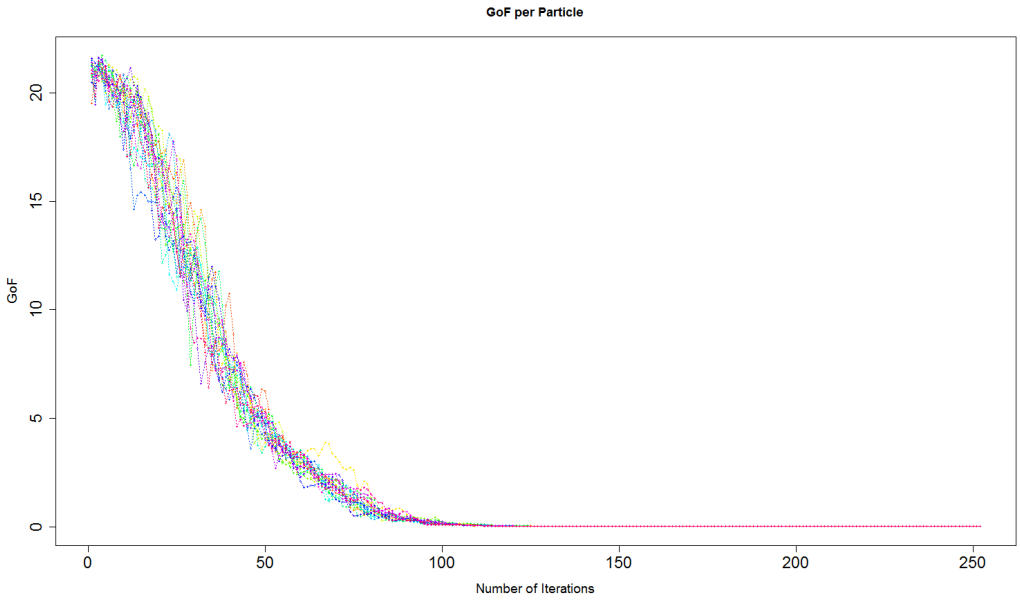

Figure 11: Goodness-of-fit measure for particles per iteration.

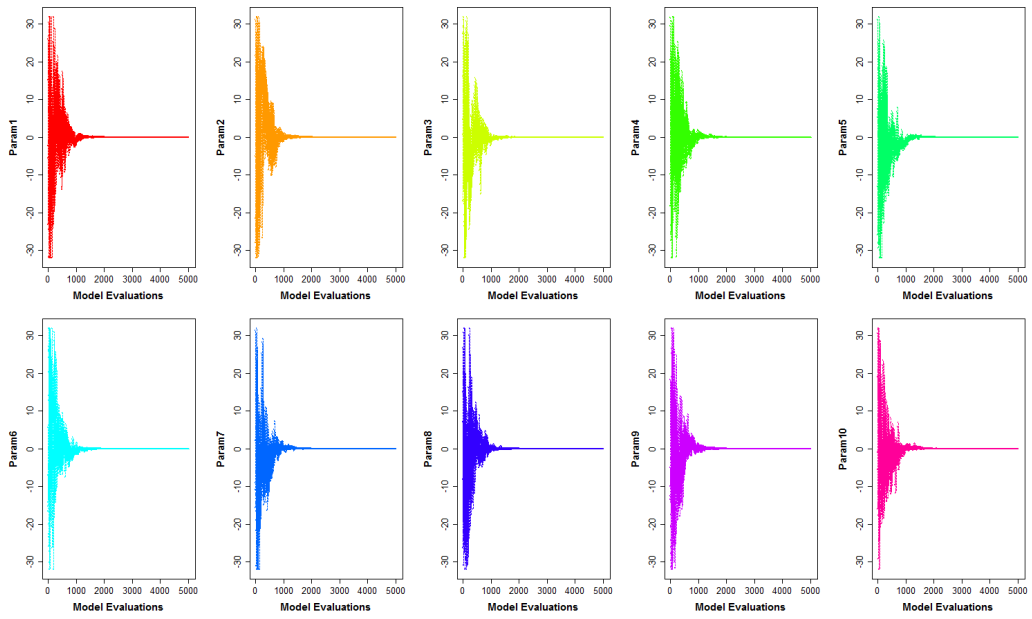

Figure 12: Parameter value per run.

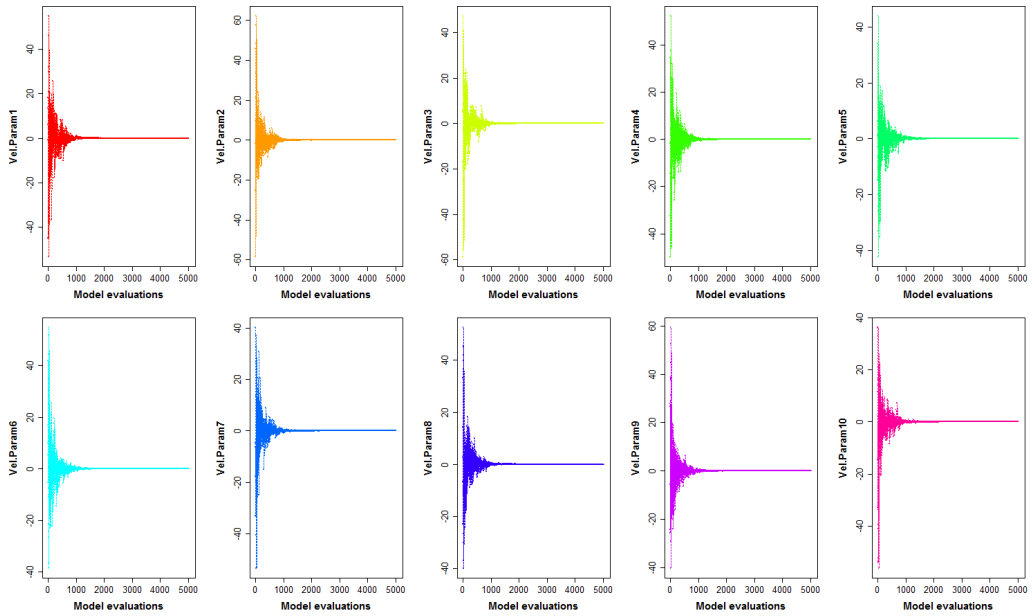

Figure 13: Particle velocities per run.

4.2.2 Optimisation of Rastrigin Function

1. The **Rastrigin** function (Equation 12) is a non-convex, multi-modal and additively separable function. It has several local optima arranged in a regular lattice with a global optimum located at `o=(0,...,0)`. The search range for the Rastrigin function is [-5.12;5.12] in each variable. This function is a fairly difficult problem due to its large search space and its large number of local minima. Gradient-based algorithms will be most likely trapped in local optima.

$$\begin{aligned} f(x) &= 10n + \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i)\right] ; \\ &-5.12 \leq x_i \leq 5.12 \ ; \ i = 1, 2, \ldots, n. \end{aligned} \tag{12}$$

2. For optimising the Rastrigin function we need to define the upper and lower limits of the search space [-5.12;5.12] and its dimensionality (D=5). Default values are used for the *hydroPSO* optimisation engine (`method="spso2011"`, `npart=40`, `maxit=1000`), whereas `write2disk=FALSE` is selected to speed up the optimisation by avoiding writing the results to the disk:

```
> D <- 5
> lower <- rep(-5.12,D)
> upper <- rep(5.12,D)
> set.seed(100)
> hydroPSO(fn=rastrigin, lower=lower, upper=upper,
+          control=list(write2disk=FALSE))

$par
      Param1       Param2       Param3       Param4
4.382399e-05 9.947626e-01 9.950853e-01 9.948407e-01
      Param5
5.619973e-05

$value
[1] 2.984892

$best.particle
[1] 5
```

```
$counts
function.calls     iterations    regroupings
          5920            148              0

$convergence
[1] 1

$message
[1] "Converged ('reltol' criterion)"
```

In the previous example, the algorithm finished before reaching the maximum number of iterations (`maxit=1000`) because the relative tolerance was reached. Therefore, we decrease the relative tolerance to allow all the iterations be executed `reltol=1E-20`.

3. Using the `vonNeumann` topology:

```
> set.seed(100)
> hydroPSO(fn=rastrigin,lower=lower,upper=upper,
+          control=list(topology="vonNeumann", reltol=1E-20,
+                       REPORT=50, write2disk=FALSE))

$par
       Param1        Param2        Param3        Param4
-7.074624e-10  2.367807e-09  1.368931e-09  9.949586e-01
       Param5
-9.949586e-01

$value
[1] 1.989918

$best.particle
[1] 33

$counts
function.calls     iterations    regroupings
         40000            1000              0

$convergence
[1] 3
```

```
$message
[1] "Maximum number of iterations reached"
```

With this configuration, the maximum number of iterations was reached (see the `message` output), with a better global optimum than in the previous case (`1.99`).

4. From the R console output we see premature convergence around iteration 300 for a NSR ca. `7.0E-02`, where the global optimum got stagnated in `1.990E+00`. One option implemented in *hydroPSO* to tackle this problem corresponds to the "regrouping strategy" developed by *Evers and Ghalia* (2009). For this case we activate the regrouping strategy (`use.RG`) when the NSR is smaller than a threshold `RG.thr` equal to `7.0E-02`, whereas the output directory is set to a user-defined value (`drty.out="PSO.out.rastr"`) and we report results to the screen every 50 iterations (`REPORT=50`):

```
> set.seed(100)
> hydroPSO(fn=rastrigin,lower=lower,upper=upper,
+      control=list(topology="vonNeumann", reltol=1E-20,
+                   drty.out="PSO.out.rastr", use.RG=TRUE,
+                   RG.thr=7e-2,RG.r=3, RG.miniter=50,
+                   REPORT=50))

$par
       Param1        Param2        Param3        Param4
-3.249248e-06 -7.620773e-06  1.241578e-05 -2.879237e-06
       Param5
-9.949444e-01

$value
[1] 0.9949591

$best.particle
[1] 38

$counts
function.calls     iterations    regroupings
         40000           1000              3

$convergence
[1] 3
```

```
$message
[1] "Maximum number of iterations reached"
```

From the previous example we see that the regrouping strategy allows particles escaping from stagnation and finding a new optimum (`9.9E-01`), which is better than the optimisation without regrouping (`1.99E+00`) for the same number of iterations (`maxit=1000`).

5. By setting the working directory to `PSO.out` and using the `read_convergence` *hydroPSO* function we can directly assess the results from the optimisation as function of the iterations:

```
> setwd("./PSO.out.rastr")
> conv <- read_convergence(do.png=TRUE,
+                          png.fname="ConvergenceMeasuresRG.png")
```

Figure 14 shows the effect of the regrouping strategy. In this figure we observe the first stagnation occurring at iteration 111, and the corresponding triggering of the regrouping for NSR values smaller than `7.0E-02`. After the first triggering, a new exploration stage is activated until a second stagnation is observed (at iteration 355), triggering a second re-grouping of the swarm. The aforementioned process is repeated a third time at iteration 786 before reaching the maximum number of iterations.

6. Using the `fips` PSO variant with a `gbest` topology:

```
> set.seed(100)
> hydroPSO(fn=rastrigin,lower=lower,upper=upper,
+          method="fips", control=list(topology="gbest",
+          reltol=1E-9, write2disk=FALSE))

$par
       Param1        Param2        Param3        Param4
-2.164684e-06 -9.848671e-07  7.784858e-07  1.505466e-06
       Param5
-5.175599e-07

$value
[1] 1.745086e-09

$best.particle
```

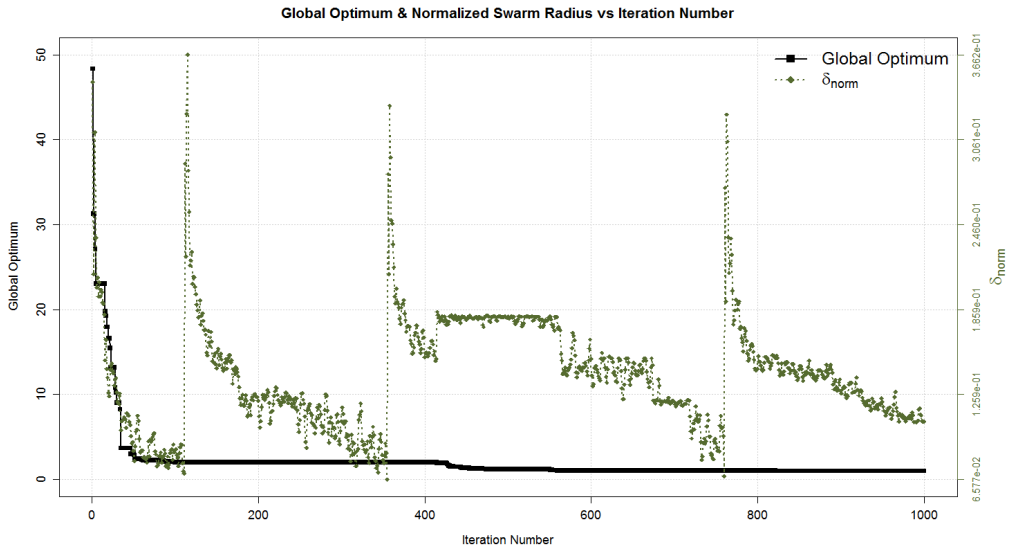

Figure 14: Effect of regrouping strategy on the Global Optimum and the Normalised Swarm Radius (NSR) versus iteration number.

```
[1] 40

$counts
function.calls     iterations    regroupings
          3720             93              0

$convergence
[1] 1

$message
[1] "Converged ('reltol' criterion)"
```

With the `fips` variant the maximum relative tolerance was reached, but with a global optimum much better than in the previous cases equal to `1.745E-09`.

7. Finally, using the `fips` PSO variant with a relative tolerance of `1E-20` and an absolute tolerance equal to `0`:

```
> set.seed(100)
> hydroPSO(fn=rastrigin,lower=lower,upper=upper,
```

```
+           method="fips", control=list(topology="gbest",
+           reltol=1E-20, abstol=0, write2disk=FALSE,
+           REPORT=10))

$par
       Param1        Param2        Param3        Param4
 7.788527e-10  8.870123e-10  8.663895e-10 -2.135894e-09
       Param5
-2.879583e-09

$value
[1] 0

$best.particle
[1] 3

$counts
function.calls     iterations    regroupings
          5320            133              0

$convergence
[1] 0

$message
[1] "Converged ('abstol' criterion)"
```

This time we obtained the true global optimum for the Rastrigin function at 0.

hydroPSO has been validated (in terms of computational implementation) against the Standard PSO 2011 algorithm developed by *Clerc* (2012) employing different test functions commonly used to assess the performance of optimisation algorithms. Validation indicates that both the Standard PSO 2011 and *hydroPSO* produce comparable average results for fixed boundary condition, topology, inertia weight and number of iterations. At the same time, we have validated the performance of *hydroPSO* against standard optimisation (calibration) algorithms such as SCE_UA (*Duan et al.*, 1992, 1994), DREAM (*Vrugt et al.*, 2008), and DEOptim (*Mullen et al.*, 2011). For a detailed validation analysis we refer the reader to *Zambrano-Bigiarini and Rojas* (2012).

Finally, here we have illustrated a few options to increase the performance of the *hydroPSO* package or to adapt the optimisation engine to different

problems (e.g. premature convergence). *hydroPSO* offers numerous fine-tuning options that are worth exploring for specific optimisation problems. We must note, however, that a successful optimisation for a given model code is most likely a proper combination of modeller's expertise and a versatile optimisation engine.

5 Calibration of a Semi-Distributed Hydrological Model Using *hydroPSO*

5.1 Hydrological System and Conceptualization

The Ega River is a tributary of the Ebro River and originates in Álava (Cantabrian mountain range) flowing through the province of Navarra (see Figure 15). It has an area of 1445 km^2 and elevations ranging from 300 to 1400 m above sea level (a.s.l.) (*CHE*, 2000). For the implementation of the *hydroPSO* package we concentrate on the headwater of the Ega catchment. This upper catchment has an area of 808 km^2, mean annual precipitation of ca. 818 mm year^{-1} and mean daily discharge equal to 12.5 m^3 s^{-1} measured in Ega en Estella (Q071 in Figure 15) for the period 1961-1990.

Simulated daily discharges were obtained with the Soil and Water Assessment Tool (SWAT) 2005 (*Arnold et al.*, 1998; *Arnold and Fohrer*, 2005), which is a basin scale, physically-based, continuous-time hydrological model operating on a daily time step. Model components include weather, hydrology, erosion/sedimentation, and diverse components for the plant-soil-nutrients system. For a full overview the reader is referred to *Arnold et al.* (1998) and *Neitsch et al.* (2005).

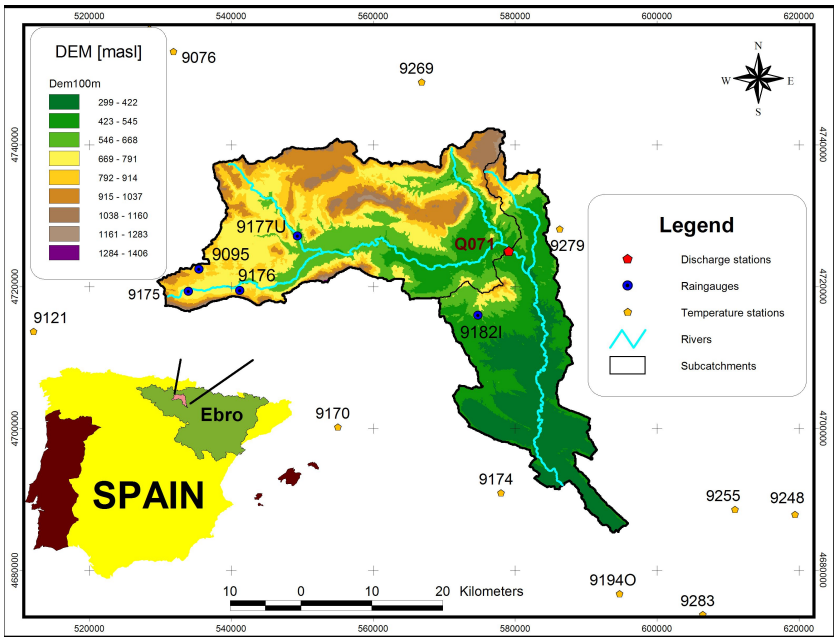

Figure 15: Location of the Ega River catchment and gauging station (Q071) used for calibration.

Table 4: Parameters of the SWAT-2005 model relevant for hydrological simulation. Range and sensitivity ranking constitute the basis for the implementation of the *hydroPSO* package to calibrate SWAT-2005 in the upper Ega catchment.

Parameter		SWAT-2005	Range		Default	Sensitivity
		File	Min	Max	Value	Ranking[1]
Baseflow alpha factor [days]	ALPHA_BF	*.gw	1.00e-01	9.90e-01	4.80e-02	1
Manning's "n" value for the main channel [-]	CH_N2	*.rte	1.60e-02	1.50e-01	1.40e-02	2
Initial SCS CN II value [-]	CN2	*.mgt	4.00e+01	9.50e+01	5.21e+02[2]	3
Saturated hydraulic conductivity [mm/hr]	SOL_K	*.sol	1.00e-03	1.00e+03	4.28e+00[2]	4
Available water capacity [mm H2O/mm soil]	SOL_AWC	*.sol	1.00e-02	3.50e-01	1.20e-01[2]	5
Effective hydraulic conductivity in main channel alluvium [mm/hr]	CH_K2	*.rte	0.00e+00	2.00e+02	0.00e+00	6
Soil evaporation compensation factor [-]	ESCO	*.hru	1.00e-02	1.00e+00	9.50e-01	7
Surface runoff lag time [days]	SURLAG	*.bsn	1.00e+00	1.20e+01	4.00e+00	8
Snowfall temperature [°C]	SFTMP	*.bsn	-5.00e+00	5.00e+00	1.00e+00	9
Snowmelt base temperature [°C]	SMTMP	*.bsn	-5.00e+00	5.00e+00	5.00e-01	10[3]
Minimum melt factor for snow [°C]	SMFMN	*.bsn	1.40e+00	6.90e+00	4.50e+00	11[3]
Snowpack temperature lag factor [-]	TIMP	*.bsn	1.00e-02	1.00e+00	1.00e+00	12[3]
Maximum melt factor for snow [°C]	SMFMX	*.bsn	1.40e+00	6.90e+00	4.50e+00	13[3]
Manning's "n" value for overland flow [-]	OV_N	*.hru	8.00e-03	6.00e-01	1.00e-01	14[3]
Deep aquifer percolation factor [-]	RCHRG_DP	*.gw	0.00e+00	1.00e+00	5.00e-02	15[3]
Threshold water depth in the shallow aquifer for flow [mm]	GWQMN	*.gw	0.00e+00	5.00e+03	0.00e+00	16[3]
Groundwater "*revap*" coefficient [-]	GW_REVAP	*.gw	0.00e+00	2.00e-01	2.00e-02	17[3]
Groundwater delay time [days]	GW_DELAY	*.gw	1.00e+00	1.00e+02	3.10e+01	18[3]
Moist soil albedo	SOL_ALB	*.sol	0.00e+00	1.00e-01	1.00e-02[2]	19[3]
Threshold water depth in the shallow aquifer for "*revap*" [mm]	REVAVMN	*.gw	1.00e+00	5.00e+02	1.00e+00	22[3]
Plant uptake compensation factor [-]	EPCO	*.bsn	1.00e-02	1.00e+00	1.00e+00	22[3]
Maximum canopy storage [mm H_2O]	CANMX	*.hru	0.00e+00	1.00e+01	0.00e+00	22[3]

[1] Sensitivity analysis based on LH-OAT (see Section 5.4.2).
[2] Default values based on the study area information.
[3] Insensitive parameters obtained from the LH-OAT analysis.

We set up SWAT-2005 with the modified SCS curve number for computing surface runoff, the Priestley-Taylor method for computing the evapotranspiration (ETP), and the Muskingum procedure for the channel routing (see *Neitsch et al.*, 2005). The AVSWAT-X GIS interface (see *Di Luzio et al.*, 2004) was used to prepare all the input files required by SWAT-2005. Information on weather, topography, soil properties, and land use in the study area was provided by the Confederación Hidrográfica del Ebro (CHE). Dominant soils in the catchment are marlstones, argillaceous marlstones, and breccia, whereas dominant land uses are forest (57.5%), pasture (39.0%), agriculture (2.95%), rocks (0.4%) and urban areas (0.12%). Precipitation estimates were obtained from interpolation of daily data in four rain gauges (P9175, P9176, P9095, P9177U see Figure 15).

SWAT-2005 contains numerous parameters describing processes where hydrology, water quality, and the soil-plant system interact. Table 4 shows a subset of 22 parameters (potentially) relevant for hydrological simulations only. These parameters are located in different files required for SWAT-2005 (see column Location in Table 4) and, thus, this tutorial illustrates how to interface *hydroPSO* and a model code with multiple controlling files (see Section 5.2). As explained later, only sensitive parameters were selected for calibrating the hydrological model for the Ega headwater catchment.

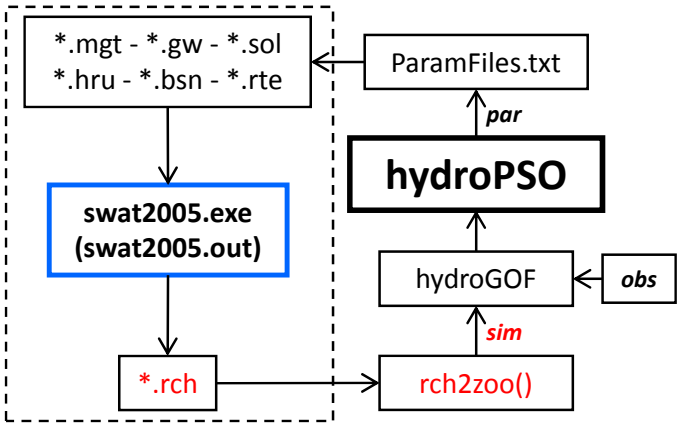

Figure 16: Interaction of *hydroPSO* with SWAT-2005 and main I/O wrapper functions defined.

5.2 Interfacing *hydroPSO* and SWAT-2005

The interaction between SWAT-2005 and *hydroPSO* is shown in Figure 16. Here, SWAT-2005 is executed by `swat2005.exe` (or `swat2005.out` under GNU/Linux), which reads several input files (e.g. *.mgt, *.gw, *.sol, *.hru, *.bsn, and *.rte) containing the parameters to be calibrated. `swat2005.exe` executes the SWAT-2005 model code and produces the *.rch file, which contains simulated discharges for defined river reaches. River discharges simulated by SWAT-2005 are then read by the *hydroPSO* function `rch2zoo`, which transforms the time series of discharge values into a `zoo` R object (***sim***). Subsequently, the *hydroGOF* R package (see `http://cran.r-project.org/web/packages/hydroGOF/`) is used to calculate a Nash-Sutcliffe Efficiency (NSE) (see *Nash and Sutcliffe*, 1970) as goodness-of-fit measure (Note that loading the *hydroGOF* R package gives the user a full suite of goodness-of-fit measures such as: root mean square error (rms), normalised rms (nrms), Pearson correlation coefficient (r), coefficient of determination (R2), modified NSE (mNSE), index of agreement (d), coefficient of persistence (cp), percent bias (pbias), Kling-Gupta Efficiency (KGE) among others). NSE is used to assess the quality of the current parameter set (particles' positions) by *hydroPSO*. Then, *hydroPSO* updates the current particles' positions (parameter set) on the basis of the current NSE and the new updated parameter values (best particles' positions) are written into the corresponding files defined in `ParamFiles.txt`.

5.3 Definition of `ParamFiles.txt` and `ParamRanges.txt` files

The interaction between *hydroPSO* and SWAT-2005 is defined through the specification of, first, the names and location of the parameters to be calibrated and, second, the definition of meaningful parameter ranges. This information is provided by the (problem-specific) ASCII files `ParamFiles.txt` and `ParamRanges.txt`, which must be contained in the `./PSO.in` folder.

In general, if a sensitivity analysis is implemented for identifying (in)sensitive parameters, the *hydroPSO* user will have to define two sets of interfacing files, one for the sensitivity analysis (most likely including all parameters considered) and other for the calibration stage (most likely including only sensitive parameters). In the SWAT-2005 case study, `ParamFiles-Sens.txt` and `ParamRanges-Sens.txt` are defined as the interfacing files used for the sensitivity analysis, which correspond to the full set of hydrology-related parameters listed in Table 4, whereas `ParamFiles.txt` and `ParamRanges.txt` are the interfacing files used during calibration.

ParamFiles-Sens.txt

ParameterNmbr	ParameterName	Filename	Row.Number	Col.Start	Col.End	DecimalPlaces
1	CN2	000010001.mgt	11	4	16	5
1	CN2	000020001.mgt	11	4	16	5
2	ESCO	basins.bsn	13	4	16	3
3	SURLAG	basins.bsn	20	4	16	3
4	RCHRG_DP	000010001.gw	9	1	16	7
4	RCHRG_DP	000020001.gw	9	1	16	7
5	GWQMN	000010001.gw	6	1	16	7
5	GWQMN	000020001.gw	6	1	16	7
6	GW_REVAP	000010001.gw	7	1	16	7
6	GW_REVAP	000020001.gw	7	1	16	7
7	REVAPMN	000010001.gw	8	1	16	7
7	REVAPMN	000020001.gw	8	1	16	7
8	GW_DELAY	000010001.gw	4	1	16	7
8	GW_DELAY	000020001.gw	4	1	16	7
9	ALPHA_BF	000010001.gw	5	1	16	7
9	ALPHA_BF	000020001.gw	5	1	16	7
10	SOL_K	000010001.sol	11	28	39	5
10	SOL_K	000010001.sol	11	40	51	5
10	SOL_K	000010001.sol	11	52	63	5
10	SOL_K	000020001.sol	11	28	39	5
10	SOL_K	000020001.sol	11	40	51	5
10	SOL_K	000020001.sol	11	52	63	5
11	SOL_AWC	000010001.sol	10	28	39	5
11	SOL_AWC	000010001.sol	10	40	51	5
11	SOL_AWC	000010001.sol	10	52	63	5
11	SOL_AWC	000020001.sol	10	28	39	5
11	SOL_AWC	000020001.sol	10	40	51	5
11	SOL_AWC	000020001.sol	10	52	63	5
12	CH_N2	000010000.rte	6	4	16	3
12	CH_N2	000020000.rte	6	4	16	3
13	CH_K2	000010000.rte	7	4	16	3
13	CH_K2	000020000.rte	7	4	16	3
14	OV_N	000010001.hru	5	4	16	3
14	OV_N	000020001.hru	5	4	16	3

```
15              SFTMP           basins.bsn      4          4        16      3
16              SMTMP           basins.bsn      5          4        16      3
17              SMFMX           basins.bsn      6          4        16      3
18              SMFMN           basins.bsn      7          4        16      3
19              TIMP            basins.bsn      8          4        16      3
20              EPCO            basins.bsn     14          4        16      3
21              CANMX           000010001.hru   9          4        16      3
21              CANMX           000020001.hru   9          4        16      3
22              SOL_ALB         000010001.sol  17         28        39      5
22              SOL_ALB         000010001.sol  17         40        51      5
22              SOL_ALB         000010001.sol  17         52        63      5
22              SOL_ALB         000020001.sol  17         28        39      5
22              SOL_ALB         000020001.sol  17         40        51      5
22              SOL_ALB         000020001.sol  17         52        63      5
```

ParamRanges-Sens.txt

```
ParameterNmbr   ParameterName   MinValue      MaxValue
1               CN2             40            95
2               ESCO            0.01          1
3               SURLAG          1             12
4               RCHRG_DP        0             1.0
5               GWQMN           0             5000
6               GW_REVAP        0             0.2
7               REVAPMN         1             500
8               GW_DELAY        1             100
9               ALPHA_BF        0.01          0.99
10              SOL_K           0.001         1000
11              SOL_AWC         0.01          0.35
12              CH_N2           0.016         0.150
13              CH_K2           0             200
14              OV_N            0.008         0.600
15              SFTMP           -5            5
16              SMTMP           -5            5
17              SMFMX           1.4           6.9
18              SMFMN           1.4           6.9
19              TIMP            0.01          1
20              EPCO            0.01          1
21              CANMX           0             10
22              SOL_ALB         0             0.1
```

5.4 Implementation Details and Results of the Calibration

5.4.1 Configuring `hydromod()`

Before implementing an optimisation with *hydroPSO*, the user must configure the `hydromod()` function. As explained before, `hydromod` links the model inputs, outputs, and executables with *hydroPSO*. Two of the most important arguments of `hydromod` are the `out.FUN` and the `gof.FUN` functions. The first one explicitly handles and interprets the outputs from the model code to be calibrated. Here it is important to notice that *hydroPSO* **does not require that the model outputs be stored in plain text files** since it can take advantage of ca. 4000 contributed R packages to read several spatial and spatio-temporal file formats as well as other foreign formats. This

constitutes a real advantage compared to model-independent tools restricted to read PEST-like (input/output) templates stored in plain text files. The second function calculates a (user-defined) measure of model performance by comparing observations against simulated equivalents. `gof.FUN` can be defined on the basis of the *hydroGOF* R package or on the basis of a customised R function.

Once arguments for `hydromod()` have been defined, the user should run the model to be calibrated by simply calling the `hydromod()` function. This will execute the model code only once and it will verify that model inputs, outputs, executables, and assessment of model performance are being calculated correctly.

In the following section, arguments for `hydromod()` are defined using the `rch2zoo` function from the *SWAT2R* R package for `out.FUN`, and the `NSE` (Nash-Sutcliffe Efficiency) goodness-of-fit measure from the *hydroGOF* R package.

5.4.2 Sensitivity Analysis

As first step, a sensitivity analysis is performed on the 22 parameters listed in Table 4. The `lhoat()` *hydroPSO* function allows the user to rank relevant parameters according to their impact on model predictions or performance using the Latin Hypercube One-factor-At-a-Time (LH-OAT) method developed by *van Griensven et al.* (2006). LH-OAT works by taking M LH sampling points (M strata for each parameter) and then varying by a fraction s each LH sampling point D times, where D is the number of parameters (i.e. problem dimensionality). For each LH sampling point a partial effect for each parameter is calculated and a final effect (impact on model predictions) is calculated by averaging these partial effects for each parameter. The method is very efficient requiring a total of $M(D+1)$ runs. Details of the sensitivity analysis implemented for SWAT-2005 are as follows:

1. Nash-Sutcliffe Efficiency (NSE) is used as a goodness-of-fit measure (*hydroGOF* R package).

2. Period for the analysis corresponds to 01-Jan-1962 to 31-Dec-1970 using a daily time step.

3. The number of strata for the LH sampling was defined as $M = 300$, whereas the fraction of variation was 10%, i.e. $s = 0.1$.

4. Observations used to assess the model performance are stored in the auxiliary file `SWAT_obs.txt`, whereas the auxiliary file `LHOAT-SWAT2005.R` implements the sensitivity analysis as follows:

LHOAT-SWAT2005.R

```
###Loading required libraries
library(hydroPSO)
library(hydroGOF)
library(hydroTSM)
library(SWAT2R)   # if not on CRAN, get it from \url{http://www.rforge.net/SWAT2R/}

###Definition of working directory: input, output and model files paths
model.drty <- "~/SWAT2005"
setwd(model.drty)
param.ranges <- paste(model.drty,"/PSO.in/ParamRanges-Sens.txt",sep="")

###Period of analysis (see "file.cio" SWAT file)
Sim.Ini="1962-01-01"
Sim.Fin="1970-12-31"
gof.Ini="1962-01-01"
gof.Fin="1970-12-31"

###Goodness-of-fit function, either customised or pre-defined from hydroGOF
gof.FUN <- "NSE"
gof.FUN.args <- list()

###Getting the OBSERVATIONS
q.obs <- read.zoo("SWAT_obs.txt")

###Arguments for the model to be assessed
model.FUN.args=list(
   model.drty=model.drty,
   param.files=paste(model.drty,"/PSO.in/ParamFiles-Sens.txt",sep=""),
   exe.fname="./swat2005.out", # GNU/Linux
   stderr=FALSE,
   ###Function for reading the simulated equivalents
   out.FUN="rch2zoo",
   out.FUN.args=list(
      file="output.rch",
      col.names="FLOW_OUTcms",
      out.type="Q",
      rchID=1,
      Date.Ini=Sim.Ini,
      Date.Fin=Sim.Fin,
      tstep="daily"), ###END out.FUN.args
   ###Function assessing the simulated equivalents against the observations
   gof.FUN=gof.FUN,
   gof.FUN.args=gof.FUN.args
   gof.Ini=gof.Ini,
   gof.Fin=gof.Fin,
   obs=q.obs,
) ###END model.FUN.args

###Main Latin-Hypercube One-factor-At-a-Time Sensitivity Analysis
lhoat(
   fn="hydromod",
   model.FUN="hydromod",
   model.FUN.args=model.FUN.args,
   control=list(
      N=300,
      f=0.1,
      drty.out="LH_OAT",
      param.ranges=param.ranges,
      gof.name="GoF",
      do.plots=FALSE,
      write2disk=TRUE,
```

```
      verbose= TRUE) ###END control options
) ###END lhoat
```

5. This sensitivity analysis requires a total of 6900 iterations ($M = 300$ and $D = 22$). In `LHOAT-SWAT2005.R` we define the keyword `NSE` from *hydroGOF* as goodness-of-fit measure as well as `SWAT_obs.txt` as the file containing the daily discharge observations. Within the arguments for the model code to be assessed, we must define the location of the `ParamFiles.txt` and the name of the executable file (`swat2005.out`). In addition, we specify the function for reading the simulated discharges (`out.FUN`) as `rch2zoo`, whereas the assessment of each simulation is done through `gof.FUN="NSE"`. For the `lhoat()` function we first define the keyword `hydromod`, which indicates that an external model code (i.e. not a pre-defined test function coded in *hydroPSO*) will be analysed. Arguments `N=300`, `f=0.1`, and `drty.out=LH_OAT` are used to define the number of strata, the fraction of variation for each parameter, and the name of the folder where results will be saved, respectively. `lhoat()` produces the file `LH_OAT-Ranking.txt`, which contains a ranking of parameters according to their relative importance.

LH_OAT-Ranking.txt

RankingNmbr	ParameterName	RelativeImportance
1	ALPHA_BF	5.707630e+02
2	CH_N2	2.238171e+02
3	CN2	1.874606e+02
4	SOL_K	1.433732e+02
5	SOL_AWC	1.298079e+02
6	CH_K2	9.871332e+01
7	ESCO	9.403977e+01
8	SURLAG	6.890757e+01
9	SFTMP	5.144755e+01
10	SMTMP	2.285713e+01
11	SMFMN	1.619080e+01
12	TIMP	6.051893e+00
13	SMFMX	3.735775e+00
14	OV_N	2.607991e+00
15	RCHRG_DP	4.295568e-01
16	GWQMN	3.301034e-01
17	GW_REVAP	1.705301e-01
18	GW_DELAY	6.303209e-02
19	SOL_ALB	1.192648e-02
20	REVAPMN	4.581639e-03
22	EPCO	0.000000e+00
22	CANMX	0.000000e+00

6. The ranking obtained from the sensitivity analysis is included in the last column of Table 4. We see from this table that 9 parameters are identified as sensitive using a NSE as goodness-of-fit measure for daily discharge simulations in the period 01-Jan-1962 to 31-Dec-1970 (several

trials showed that the tenth parameter, SMTMP, is relatively insensitive, and for sake of clarity, we have excluded it from the subsequent analysis). In general, this ranking is in agreement with previous research (see e.g. *Holvoet et al.*, 2005; *Muleta and Nicklow*, 2005; *van Liew et al.*, 2005; *van Griensven et al.*, 2006; *Kannan et al.*, 2007; *van Liew et al.*, 2007), and this subset of parameters constitute the basis for the calibration of the SWAT-2005 model.

The resulting files (`ParamFiles.txt` and `ParamRanges.txt`) used to interface *hydroPSO* and SWAT-2005 in the calibration stage are as follows:

ParamFiles.txt

ParameterNmbr	ParameterName	Filename	Row.Number	Col.Start	Col.End	DecimalPlaces
1	CN2	000010001.mgt	11	4	16	5
1	CN2	000020001.mgt	11	4	16	5
2	ESCO	basins.bsn	13	4	16	3
3	SURLAG	basins.bsn	20	4	16	3
4	ALPHA_BF	000010001.gw	5	1	16	7
4	ALPHA_BF	000020001.gw	5	1	16	7
5	SOL_K	000010001.sol	11	28	39	5
5	SOL_K	000010001.sol	11	40	51	5
5	SOL_K	000010001.sol	11	52	63	5
5	SOL_K	000020001.sol	11	28	39	5
5	SOL_K	000020001.sol	11	40	51	5
5	SOL_K	000020001.sol	11	52	63	5
6	SOL_AWC	000010001.sol	10	28	39	5
6	SOL_AWC	000010001.sol	10	40	51	5
6	SOL_AWC	000010001.sol	10	52	63	5
6	SOL_AWC	000020001.sol	10	28	39	5
6	SOL_AWC	000020001.sol	10	40	51	5
6	SOL_AWC	000020001.sol	10	52	63	5
7	CH_N2	000010000.rte	6	4	16	3
7	CH_N2	000020000.rte	6	4	16	3
8	CH_K2	000010000.rte	7	4	16	3
8	CH_K2	000020000.rte	7	4	16	3
9	SFTMP	basins.bsn	4	4	16	3

ParamRanges.txt

ParameterNmbr	ParameterName	MinValue	MaxValue
1	CN2	40	95
2	ESCO	0.01	1
3	SURLAG	1	12
4	ALPHA_BF	0.01	0.99
5	SOL_K	0.001	1000
6	SOL_AWC	0.01	0.35
7	CH_N2	0.016	0.150
8	CH_K2	0	200
9	SFTMP	-5	5

5.4.3 Calibration

After obtaining a ranking with the most sensitive parameters, the calibration of the river discharges for the EGA headwater catchment proceeded as follows:

1. Files interfacing *hydroPSO* and SWAT-2005 (i.e. `ParamFiles.txt` and `ParamRanges.txt`) are stored in the `./PSO.in` folder, within the directory containing all the required files to run SWAT-2005, which for this tutorial is `./SWAT2005`.

2. Several auxiliary files (described below) are included in `./SWAT2005`:

```
Auxiliary Files in SWAT2005
hydroPSO-SWAT2005.R -> Main R script to run hydroPSO
SWAT_obs.txt -> ASCII file with the discharge observations
```

   As previously explained (see Section 5.4.2), the file `LH_OAT-Ranking.txt` is saved in the folder `./SWAT2005/LH_OAT`.

3. The setup for the calibration of the Ega headwater catchment is defined in the `hydroPSO-SWAT2005.R` script. By default all the results from hydroPSO are saved into the `PSO.out` folder, however, this can be redefined by using the `drty.out` argument.

hydroPSO-SWAT2005.R

```
################################################################################################
## Example to interface SWAT-2005 with hydroPSO. This script allows hydroPSO to    #
## take control of the execution of SWAT-2005                                      #
##                                                                                 #
## Part of the hydroPSO R package                                                  #
## http://www.rforge.net/hydroPSO/ http://cran.r-project.org/web/packages/hydroPSO #
## Copyright 2011-2012 Mauricio Zambrano-Bigiarini & Rodrigo Rojas                 #
## Distributed under GPL 2 or later                                                #
##                                                                                 #
## Created by Mauricio Zambrano-Bigiarini and Rodrigo Rojas. 26-Oct-2011           #
## Last saved: 13-Oct-2012                                                         #
################################################################################################

###Loading required libraries
library(hydroPSO)
library(hydroGOF)
library(hydroTSM)
library(SWAT2R)    # if not on CRAN, get it from \url{http://www.rforge.net/SWAT2R/}

###Definition of working directory: input, output and model files paths
model.drty <- "~/SWAT2005"
setwd(model.drty)

###Getting the OBSERVATIONS
q.obs <- read.zoo("SWAT_obs.txt")

###Period of analysis
Sim.Ini="1962-01-01"
Sim.Fin="1970-12-31"
gof.Ini="1962-01-01"
gof.Fin="1970-12-31"

###Goodness-of-fit function, either customised or pre-defined from hydroGOF
gof.FUN <- "NSE"
gof.FUN.args <- list()
```

```
###MAIN model function
model.FUN.args=list(
   model.drty=model.drty,
   param.files=paste(model.drty,"/PSO.in/ParamFiles.txt",sep=""),
   exe.fname="./swat2005.out", # GNU/Linux
   verbose=FALSE,
   stderr=FALSE,
   ###Function for reading the simulated equivalents
   out.FUN="rch2zoo",   #function from the SWAT2R package
   out.FUN.args=list(
      file="output.rch",
      col.names="FLOW_OUTcms",
      out.type="Q",
      rchID=1,
      Date.Ini=Sim.Ini,
      Date.Fin=Sim.Fin,
      tstep="daily",
      verbose=FALSE), ###END out.FUN.args
   ###Function for assessing the simulated equivalents against the observations
   gof.FUN=gof.FUN,
   gof.FUN.args=gof.FUN.args,
   gof.Ini=gof.Ini,
   gof.Fin=gof.Fin,
   obs=q.obs,
   ###OPTIONAL. For plotting
   do.png=FALSE
) ###END model.FUN.args

###MAIN PSO ALGORITHM
###For hydroPSO fine-tuning parameters, see Zambrano-Bigiarini and Rojas,2012
set.seed(100)
hydroPSO(
   fn="hydromod",
   method="spso2011",
   model.FUN="hydromod",
   model.FUN.args=model.FUN.args,
   control=list(
      param.ranges="ParamRanges-09params-sub090.txt",
      normalise=TRUE,
      MinMax="max",
      npart=40,
      maxit=1000,
      reltol=1E-30,
      Xini.type="lhs",
      Vini.type="lhs2011",
      c1=2.05,
      c2=2.05,
      use.IW=FALSE,
      use.CF=TRUE,
      use.TVlambda=TRUE,TVlambda.type="linear",TVlambda.rng=c(1.0,0.5),TVlambda.exp=
      1,
      topology="random", K=11,
      boundary.wall="absorbing2011",
      write2disk=TRUE,
      REPORT=5,
      verbose=TRUE
   ) ###END control options
) ###END MAIN hydroPSO ALGORITHM
```

4. In the `hydroPSO-SWAT2005.R` script we first load all the required libraries, setup the working directory, and define the simulation period as well as the period for calculating the goodness-of-fit measure (these two could be different depending on the nature of the application). To assess the performance of each particle in *hydroPSO* we use the Nash-Sutcliffe Efficiency (NSE) implemented in the *hydroGOF* package as goodness-of-fit measure. Discharge observations used to assess the corresponding simulated equivalents are read as a `zoo` object in R (to take advantage of time attributes) from the `SWAT_obs.txt` ASCII file. Then, we define all arguments related to the main model code to be calibrated. Note that these arguments are identical as for the sensitivity analysis (`LHOAT-SWAT2005.R`) and, so, they will not be repeated here. For the main *hydroPSO* algorithm, the keyword `hydromod` **must** be used for the argument `fn` when a function different from a predefined test function is calibrated. This will indicate *hydroPSO* to expect for external files containing the model executable(s), input(s), an output(s) files. For this example, we use the `spso2011` (default) algorithm to maximise (`MinMax="max"`) the NSE calculated by *hydroGOF* (`gof.FUN="NSE"`). Parameters values are normalised to the [0,1] range during the optimisation (`normalise=TRUE`) as recommended by *Clerc* (2012) when the search space is not a hypercube. A swarm of 40 particles (`npart=40`) considering 1000 iterations (`maxit=1000`) for the algorithm is defined with a relative tolerance (`reltol`) equal to 1E-30. Particle positions are initialized following a LHS approach, whereas velocities are initialized following the SPSO2011. A constant value for the cognitive (`c1`) and social (`c2`) coefficients equal to 2.05 is used. In addition, we improve on the definition of the factor clamping the velocities (`lambda`) by using a linear variation between [1.0,0.5]. Particles interact following the `random` topology with 11 informants. Finally, we employed the absorbing boundary condition. Results are saved by default in the folder `./PSO.out`.

5. Plotting the results (Figures 17 to 28). Using the `plot_results()` function results are saved directly to the folder `./PSO.out/pngs`:

   ```
   > plot_results(do.png=TRUE)
   ```

   Although it is not the aim of this tutorial to provide an extensive analysis of the hydrological calibration for the Ega headwater catchment, we briefly discuss the results obtained from *hydroPSO*. Figure 17 shows the evolution of NSE as a function of the iteration number. We see an initial exploration phase (up to ca. iter=100) that stabilises after

iteration 200 around a value of NSE=0.777. At the same time, the Normalised Swarm Radius indicates a clear convergence to the attraction zone around this optimum. Although not shown here, several trials (including different number of iterations and particles for the swarm) indicate that this solution is most likely the global optimum for the calibration.

A dotty-plot for the different (sensitive) parameters listed in Table 4 is shown in Figure 18. From this figure we see a good identification for all parameters. Figure 19 complements the previous one by showing a 2-Dimensional projection of the goodness-of-fit surface response (NSE) for different parameter pairs. Figure 20, in turn, summarises the interaction among all (sensitive) parameters listed in Table 4. Here a clear correlation between the NSE and CH_N2, CH_K2 and CN_2 is observed, whereas a statistically significant cross-correlation is observed between CH_N2, SFTMP, CH_K2 and SOL_K. Empirical Cumulative Distribution Functions (ECDFs) (Figure 21) and histograms (Figure 22) for the parameters show a good specification with reduced uncertainty around the median and some for the extreme quantiles. Figures 23, 24, and 25 provide, in turn, useful information to assess the performance and convergence of the NSE, parameters, and velocities per iteration number. By analysing these figures we conclude that the *hydroPSO* package converged to a suitable attraction zone.

Figure 26 provides the daily time-series summarizing the calibration together with a series of performance indicators. In general, we see an underestimation of recession and peak-flows (PBIAS=-2.6%) with a good agreement of average flows (MAE=4.67) and a final optimised Nash-Sutcliffe Efficiency of 0.78. At the same time, Figure 27 shows a scatter plot of simulated versus observed daily discharges. Finally, Figure 28 shows different user-defined quantiles (5, 50 and 95) including an estimation of the percentage bias for the specified quantiles. In this figure we clearly see the relevant underestimation of low-flows obtained from this exercise (Bias=-57.3% for the Q5).

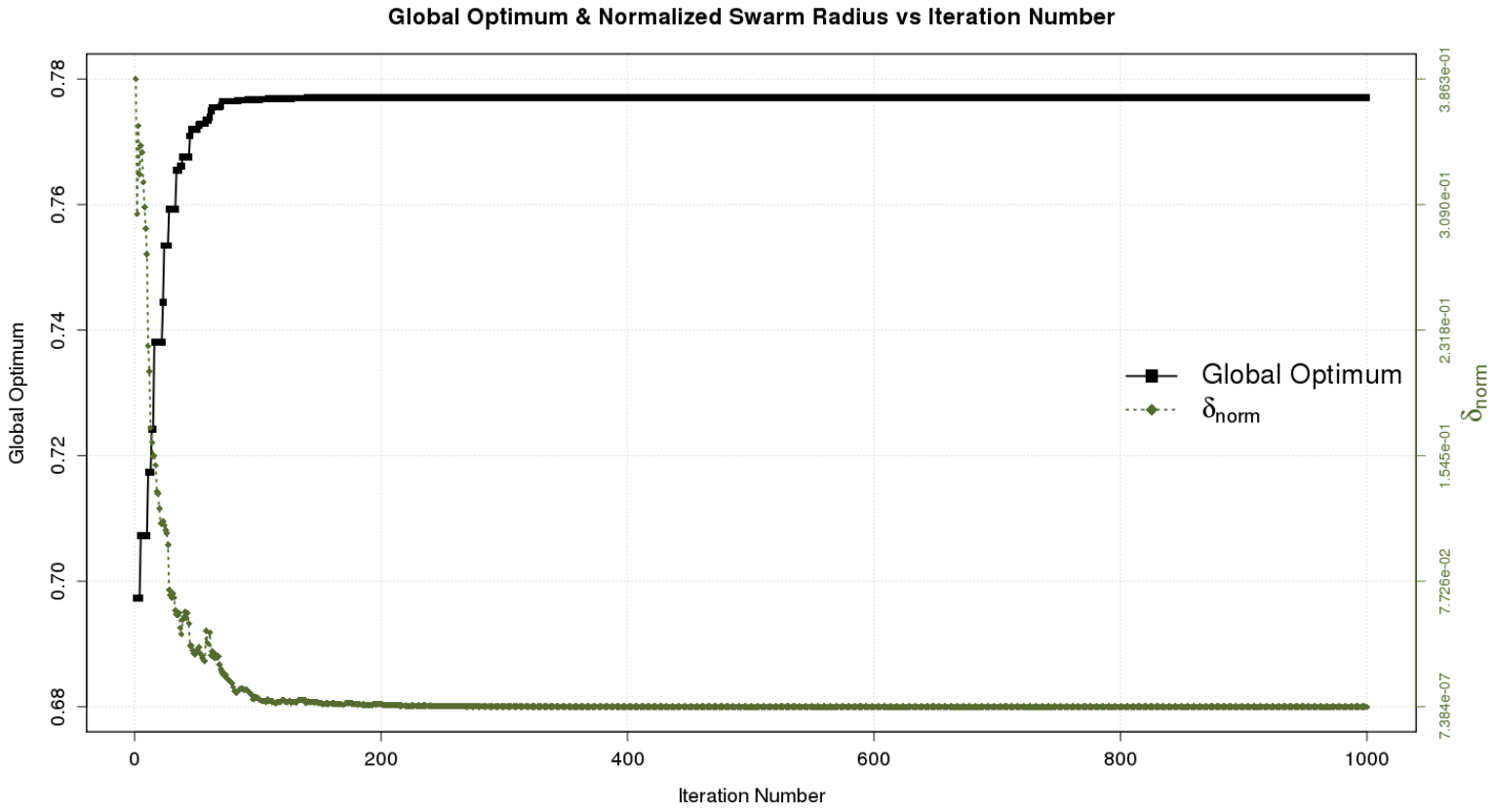

Figure 17: Evolution of the Global Optimum and the Normalised Swarm Radius (NSR) versus iteration number for the calibration of the Ega headwater catchment.

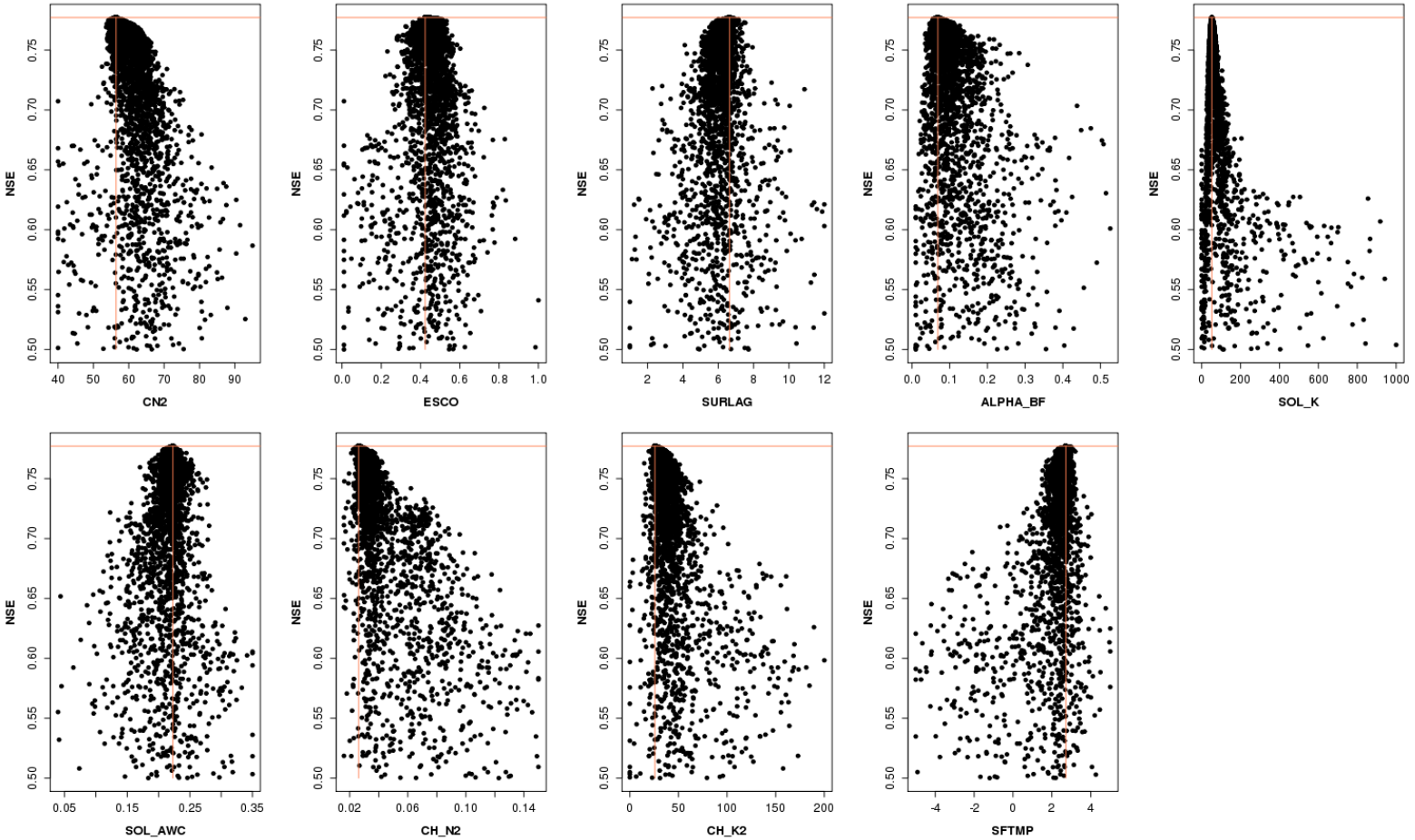

Figure 18: Dotty-plots for the (sensitive) parameters listed in Table 4. Vertical and horizontal coloured lines show the location of the highest goodness-of-fit value (Nash-Sutcliffe Efficiency).

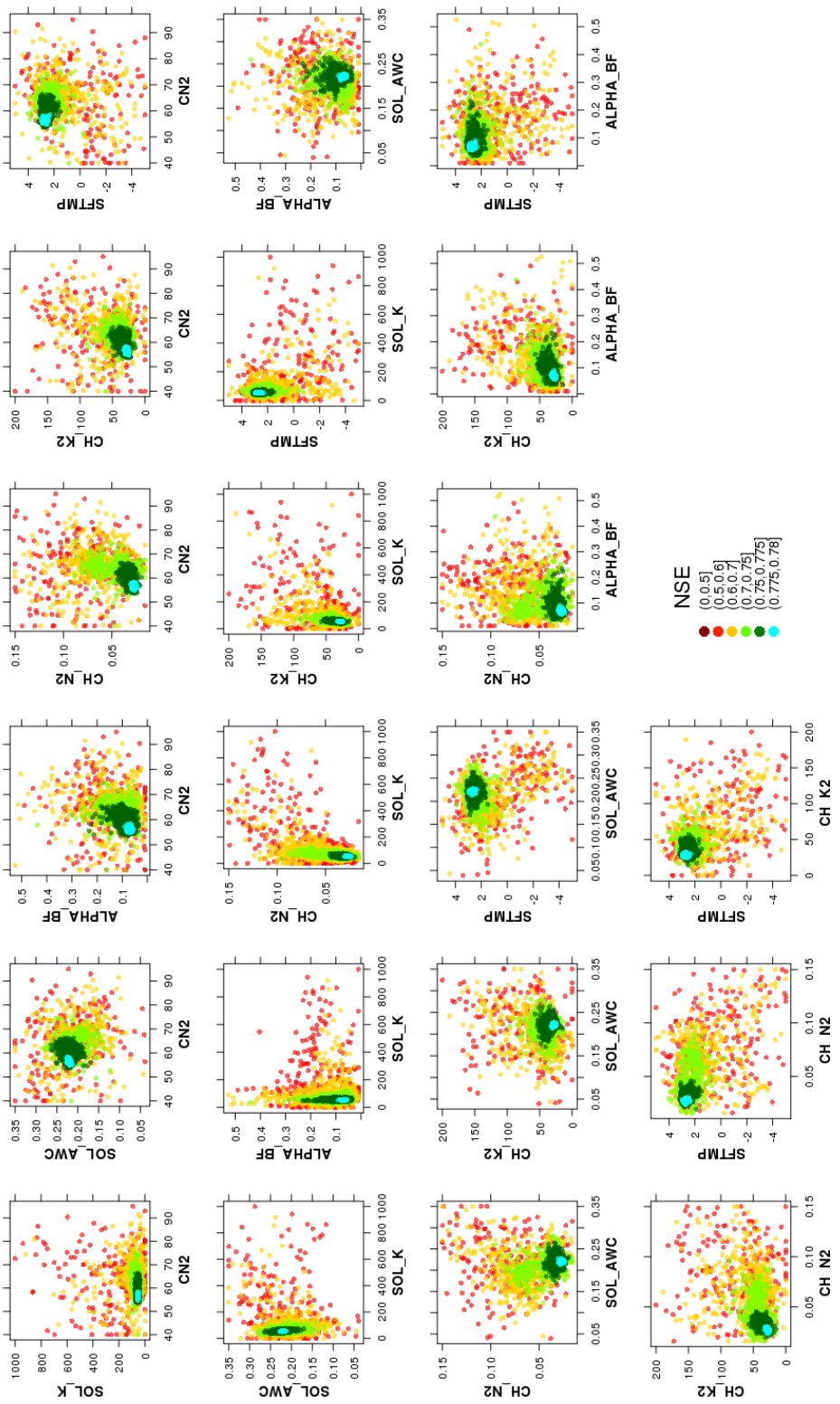

Figure 19: 2-dimensional projected dotty-plots highlighting the interaction between the first seven (sensitive) parameters listed in Table 4.

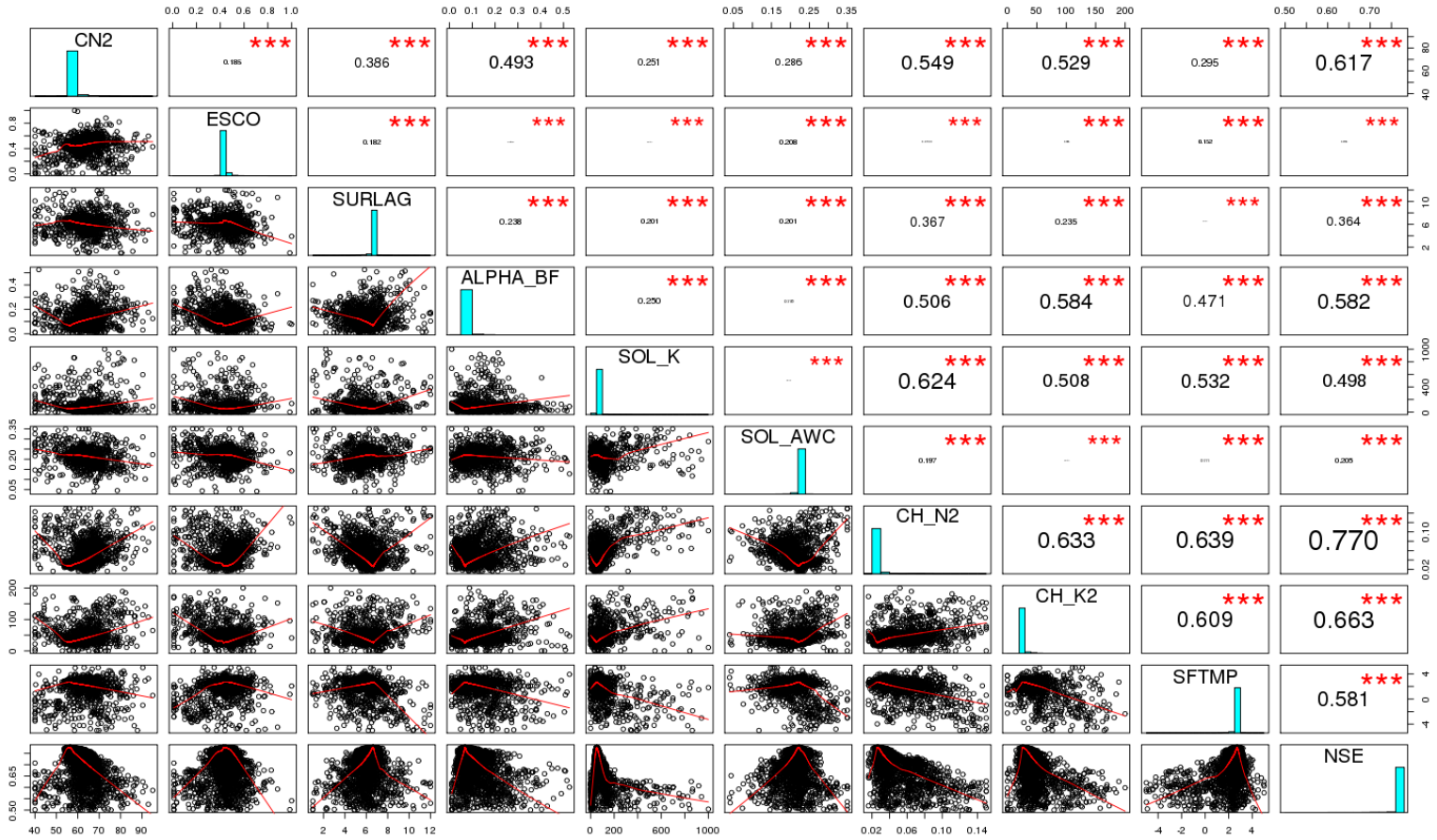

Figure 20: Matrix summarizing the cross-correlation, histograms and statistical significance of the correlation for the (sensitive) parameters listed in Table 4.

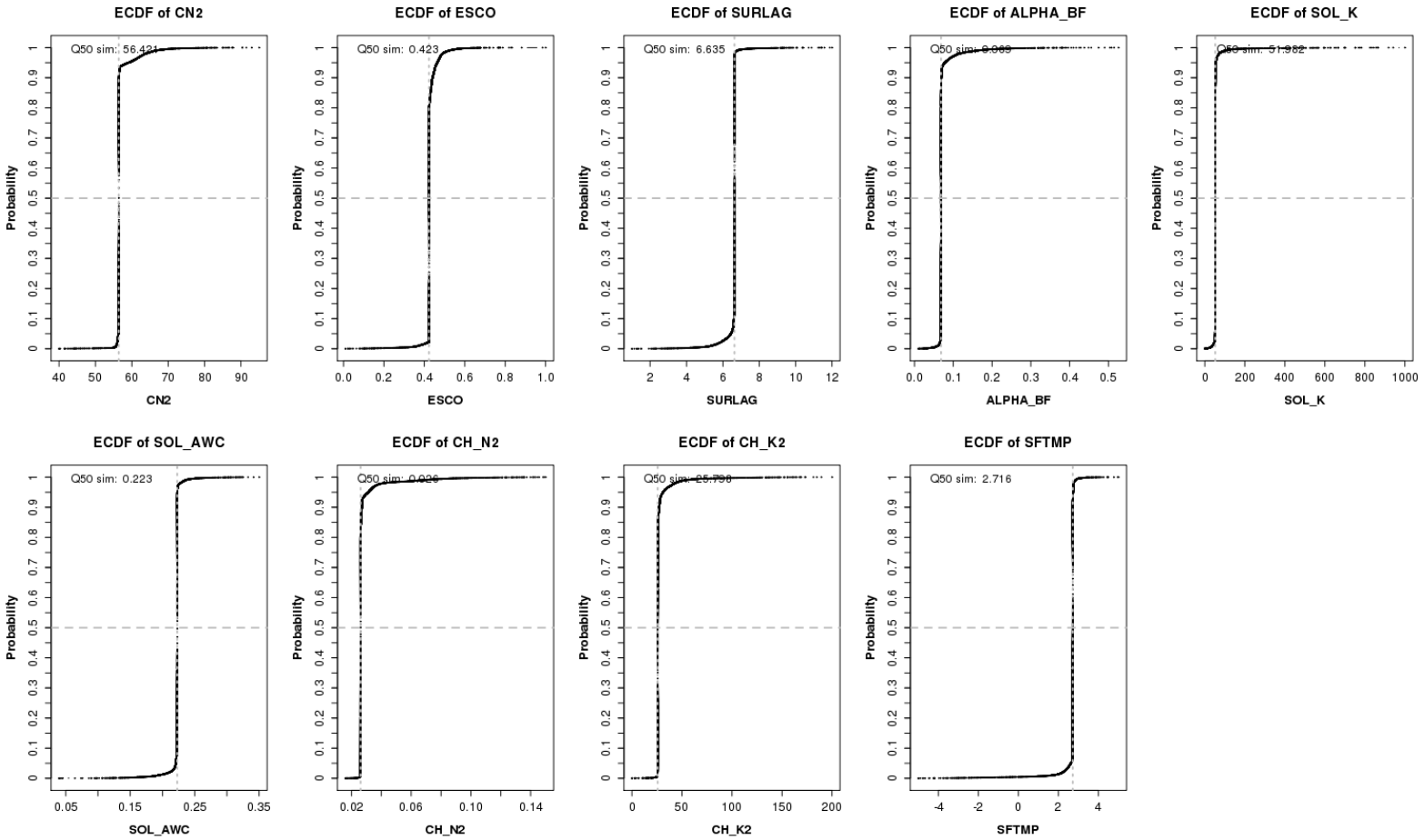

Figure 21: Empirical Cumulative Distribution Functions (ECDFs) for the (sensitive) parameters listed in Table 4. Vertical and horizontal grey dashed-line indicate the location of a user-defined percentile.

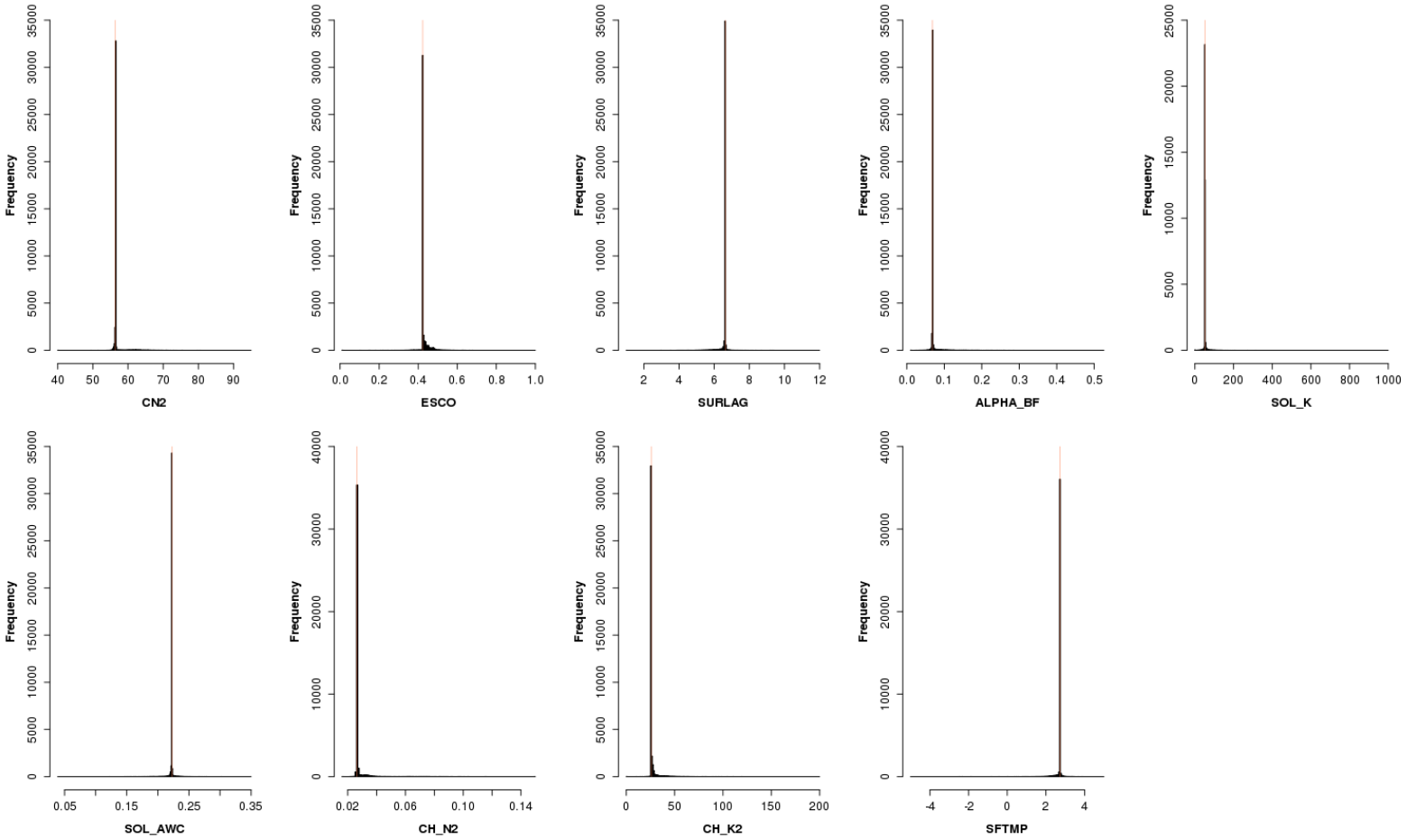

Figure 22: Histograms for the (sensitive) parameters listed in Table 4. Vertical red lines show the location of the highest goodness-of-fit value (Nash-Sutcliffe Efficiency).

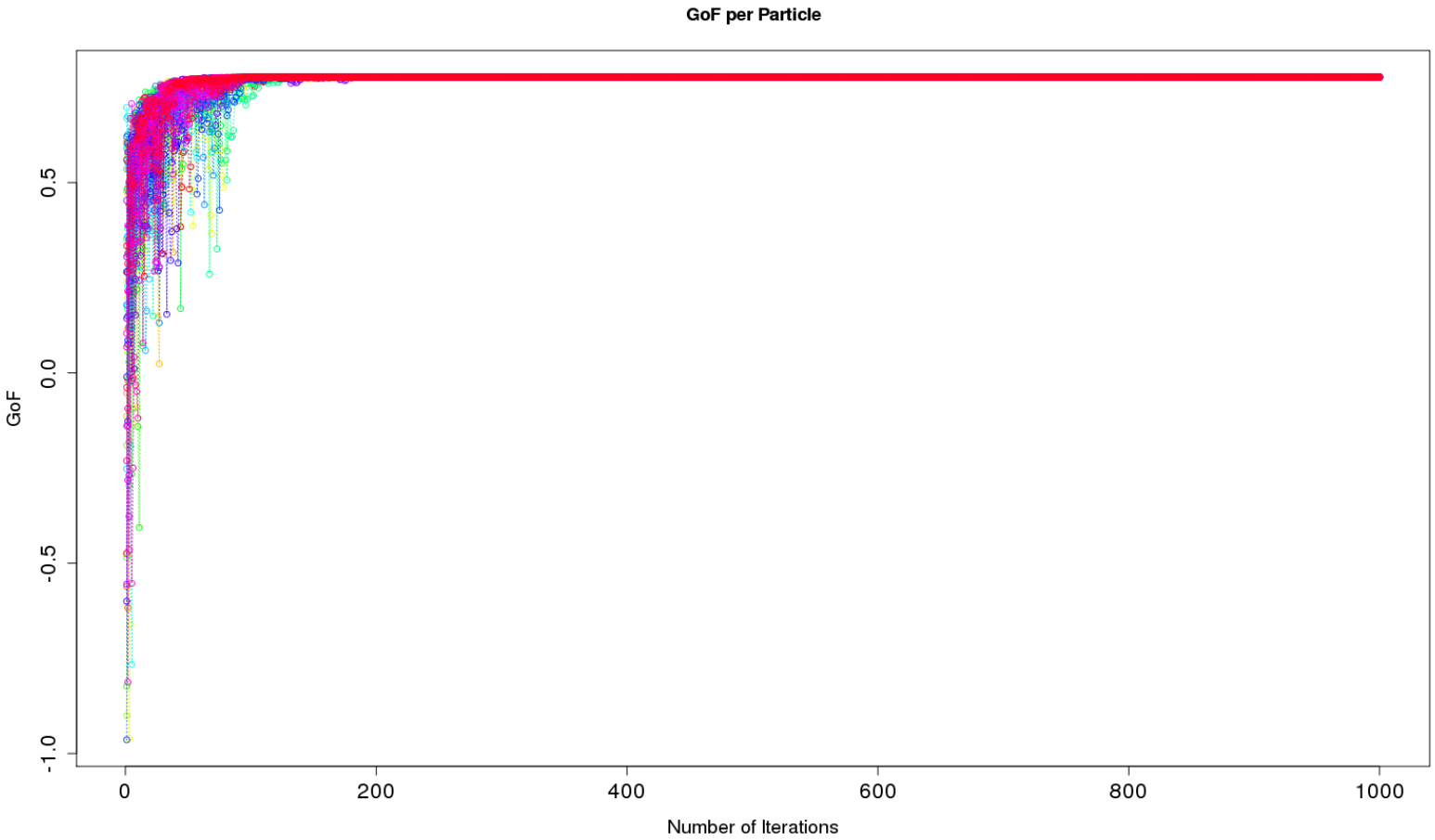

Figure 23: Evolution of the Nash-Sutcliffe Efficiency for all 20 particles for 2000 iterations.

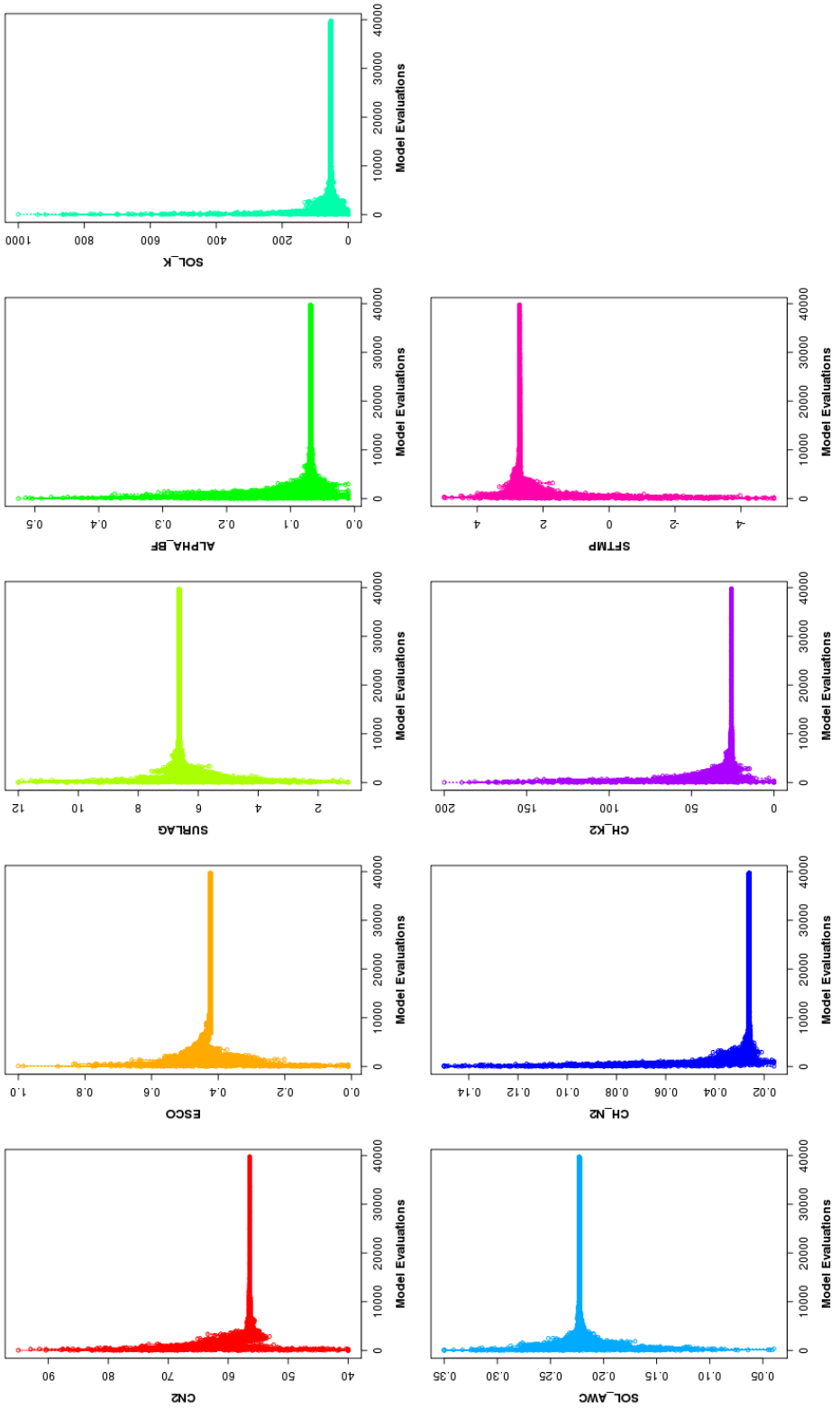

Figure 24: Convergence of the different (sensitive) parameters as a function of the model evaluation.

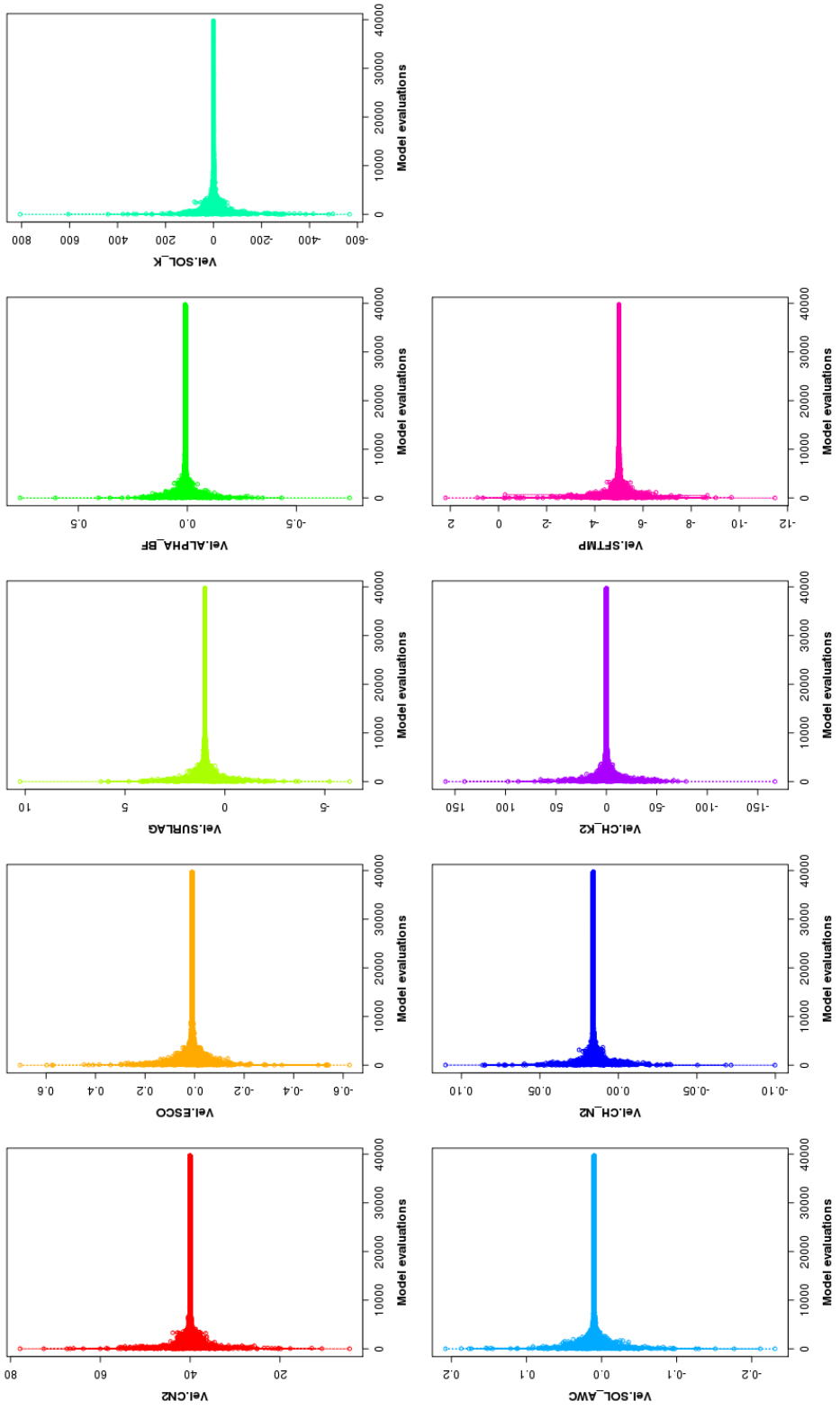

Figure 25: Convergence of the velocities for each (sensitive) parameter as a function of the model evaluation.

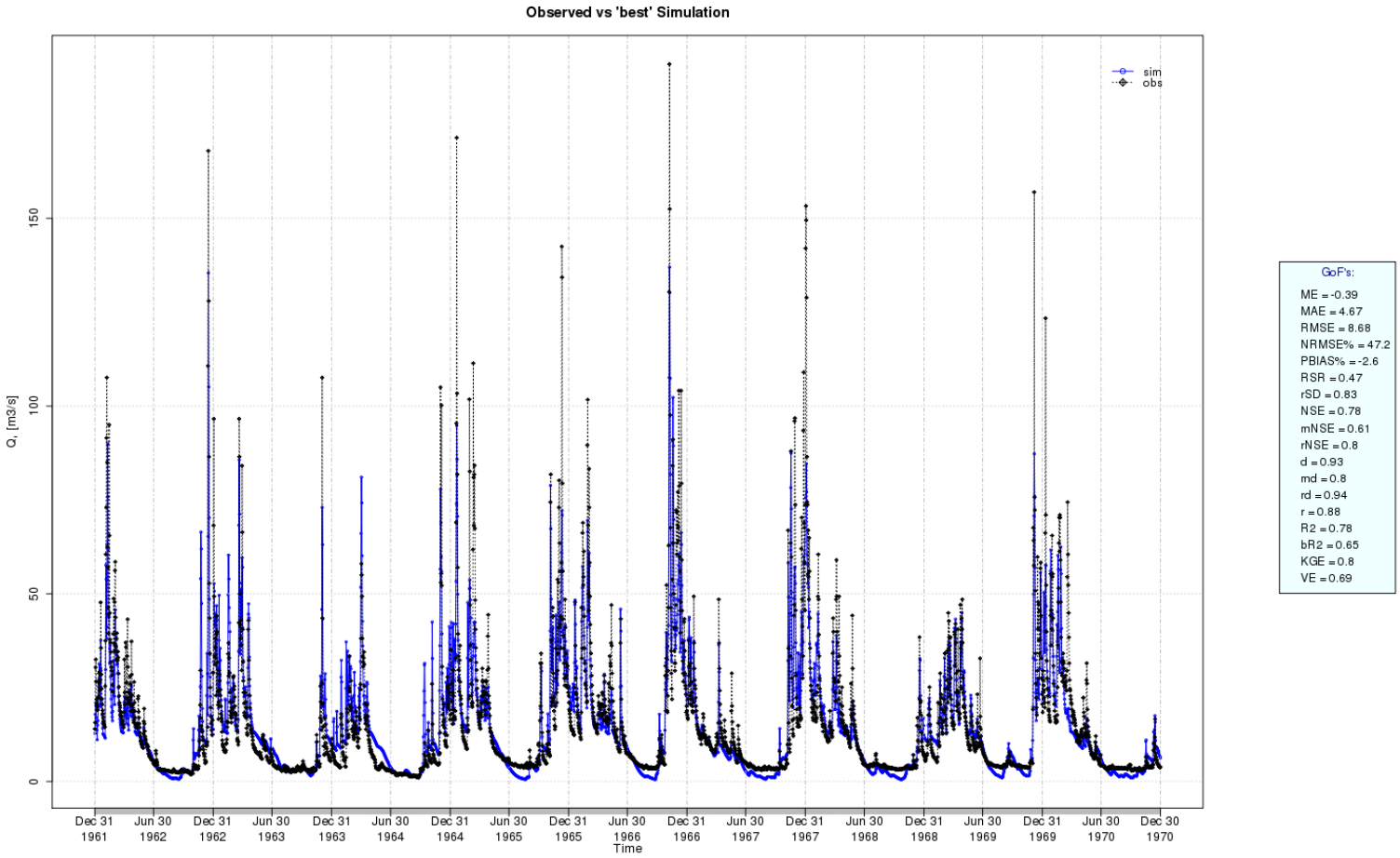

Figure 26: Simulated versus Observed discharges at the station Q071 (see Figure 15). Upper panel shows daily calibrated time series and summary box. Lower panel shows monthly (aggregated) time series and summary box.

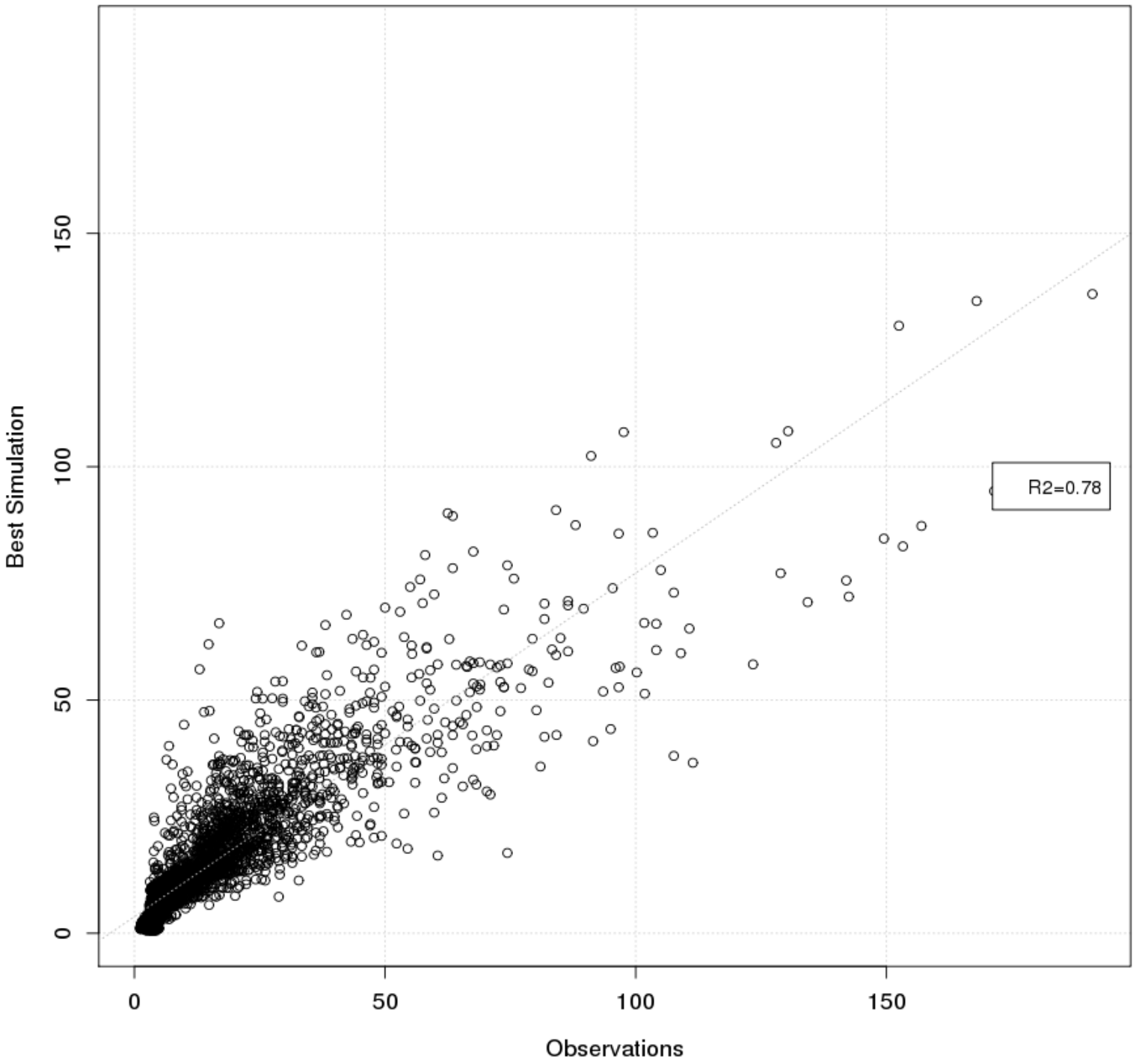

Figure 27: Scatter plot of best simulated versus observed daily discharges at the station O071 (see Figure 15)

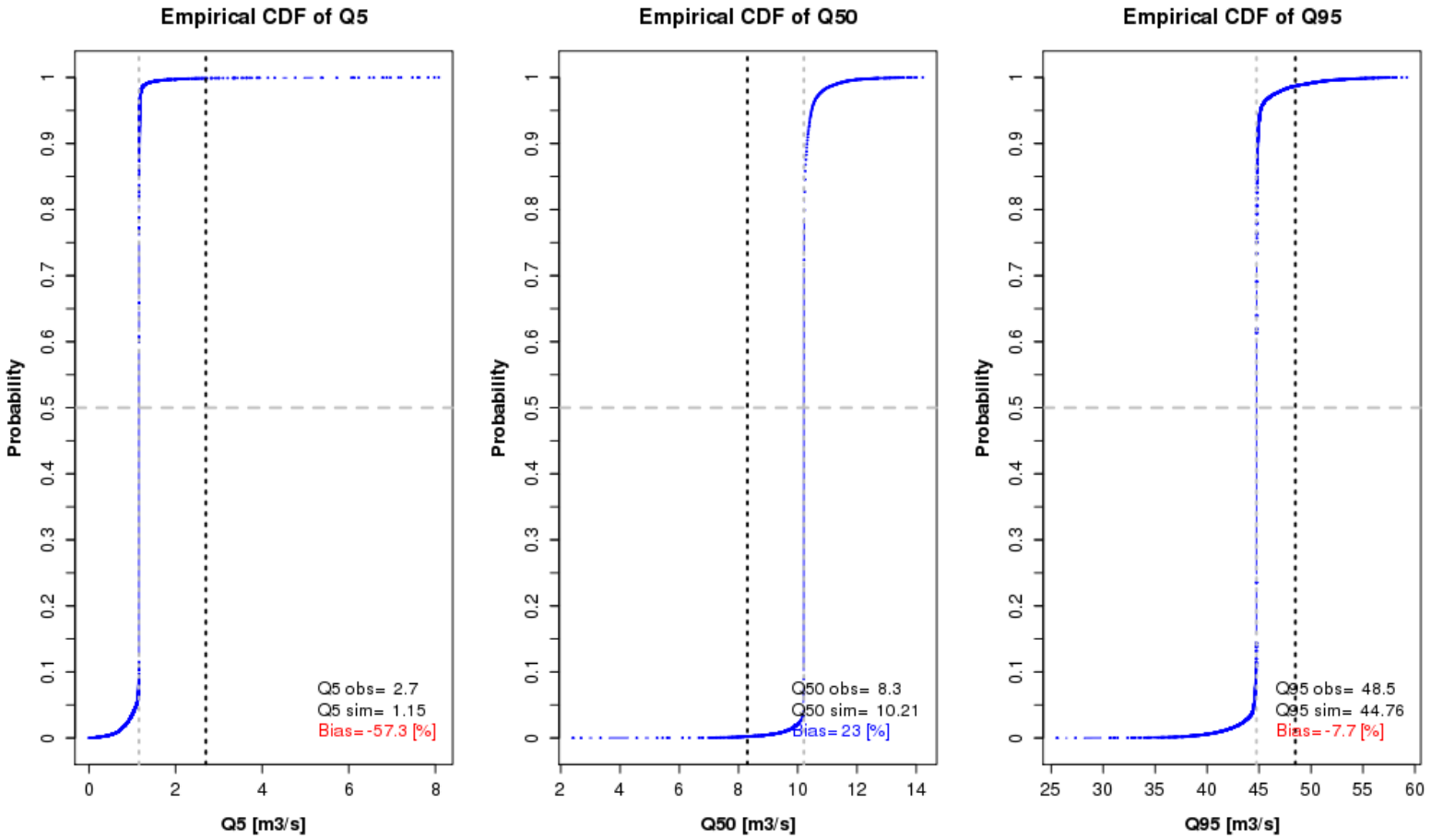

Figure 28: ECDFs for user-defined quantiles (5, 95 and 95). Vertical dashed-line represents the observed (black) and simulated (gray) quantiles.

6 Calibration of a Groundwater Flow Model Using *hydroPSO*

6.1 Groundwater System and Conceptualization

The model to be calibrated is termed "M2" and is similar to the model implemented in *Rojas et al.* (2010). The groundwater system is described by an unconfined aquifer which receives lateral groundwater recharges at the eastern boundary and over all domain through deep fissures in basement rocks. The system exchanges groundwater with contiguous aquifers at the western and southern boundaries, whereas at the northern boundary it shows a groundwater divide. Main outputs of groundwater from the system corresponds to high transpiration rates from forested areas and high evaporation rates from "playas". The groundwater system is assumed to be under steady-state conditions. An schematic representation of the system is shown in Figure 29.

This groundwater system is conceptualised as a one-layer aquifer, with the northern and southern interaction with external forcings expressed as constant-head boundaries using the BAS6 package of MF2005. Recharge mechanisms are expressed by lateral constant flux and spatially-distributed recharge rates using the WEL and RCH packages of MF2005, respectively. We consider 22 hydraulic conductivity zones obtained from *Rojas and Dassargues* (2007) implemented in the LPF package, one transpiration zone defined as negative rates using the RCH package and one evaporation zone implemented in the EVT package. In total, 30 parameters are considered for the calibration of model M2. The definition of parameters as well as their feasible ranges to implement *hydroPSO* are shown in Table 5. In addition, we consider 42 observation wells (evenly) distributed in the modelled domain for calibration purposes.

Table 5: Parameters used to calibrate model M2 with the *hydroPSO* package.

Parameter		Range	
		Min	Max
Recharge (Lateral fluxes) [m^3 d^{-1}]	RECH	0	345600
Recharge from basement rocks [m^3 d^{-1}]	RECH_BAS	0	172800
Transpiration forested areas [m^3 d^{-1}]	TRANSP	0	172800
Discharge to eastern aquifers [m^3 d^{-1}]	NORIA	0	86400
Evaporation rate [m d^{-1}]	EVTR	0	0.01
Extinction depth [m]	EXTD	0	20
Elevation constant head north [m]	CH_N	1075	1120
Elevation constant head south [m]	CH_S	875	920
Hydraulic conductivity for 22 zones [m d^{-1}]	K	0	100

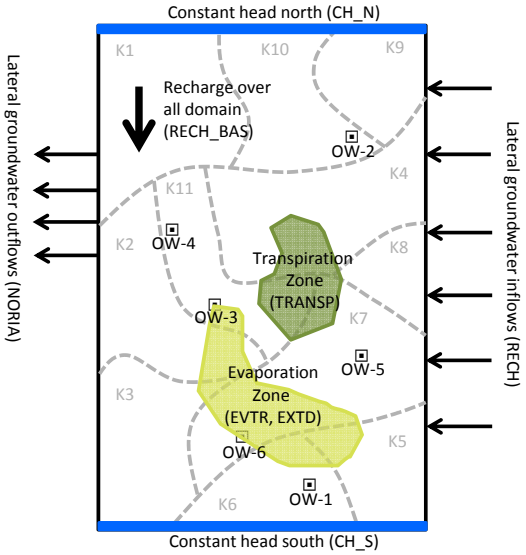

Figure 29: Groundwater system conceptualization for model "M2". Note that M2 is set up with 22 hydraulic conductivity zones and 42 observation wells.

6.2 Interfacing *hydroPSO* and MF2005 and ZB

The interaction between *hydroPSO* and M2 (MF2005/ZB) is depicted in Figure 30. *run_me.bat* is a simple batch script executing programmes `preproc`, `mf2005` and `zonebud_hydroPSO` sequentially. `preproc` is a simple read/write FORTRAN code reading the *pre_param.txt* file for M2, which is an ASCII file containing updated parameter values produced by *hydroPSO* after each iteration. `preproc` is problem-specific and copies the (updated) parameter values in the files M2.BA6, M2.EVT, M2.LPF, M2.RCH and M2.WEL, which are the files containing the parameters for M2. Therefore, for a different groundwater model possibly having different conductivity zones, geometry, boundary conditions, etc., it will be necessary to adapt `preproc` to that particular problem calibration. Special care must be taken in reading parameters as reported in the *pre_param.txt* file, whose order is defined by the user through the file `ParamRanges.txt`. Subsequently, `mf2005` executes MF2005 for the "Name File", M2.NAM. MF2005 produces a summary and the main results of the groundwater flow model (heads and global flow components) in files M2.LST, M2.BUD and M2.HED. Finally, `zonebud_hydroPSO` calculates the balance for each specific "budget" zone defined in the M2.ZON file and saves the results into the M2.BAL file. `zonebud_hydroPSO` is a modified version of the original ZB code where keyboard input is skipped by using pre-defined (hard-coded) default options.

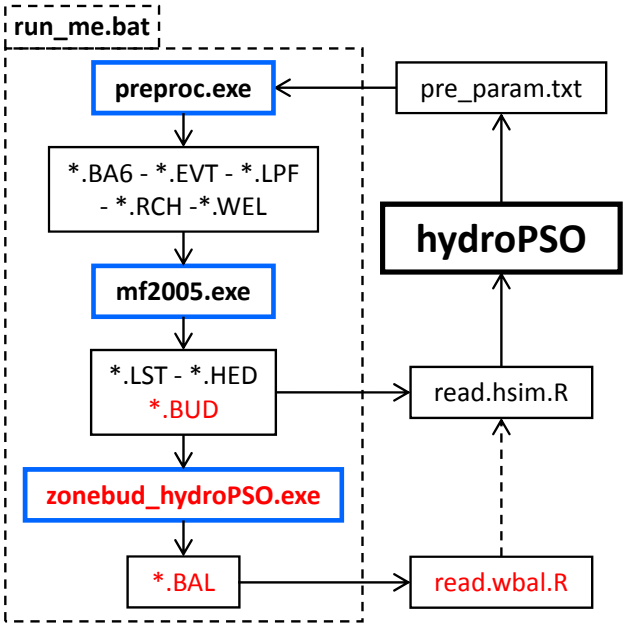

Figure 30: Interaction of *hydroPSO* with MF2005/ZB (M2) and main I/O wrapper functions defined.

6.3 Definition of `ParamFiles.txt` and `ParamRanges.txt` files

Two basic pieces of information are required to interface M2 and *hydroPSO*. First, the names and location of the (multiple) parameters to be calibrated, together with specific location and decimal positions. Second, feasible and physically meaningful parameter ranges. This information is entered in two text files, `ParamFiles.txt` and `ParamRanges.txt`, which must be stored in the subdirectory `./PSO.in`. Note that care must be taken in numbering and naming the parameters as they require to be consistent in both files.

ParamFiles.txt

ParameterNmbr	ParameterName	Filename	Row.Number	Col.Start	Col.End	DecimalPlaces
1	HK_1	pre_param.txt	2	1	20	3
2	HK_2	pre_param.txt	3	1	20	3
3	HK_3	pre_param.txt	4	1	20	3
4	HK_4	pre_param.txt	5	1	20	3
5	HK_5	pre_param.txt	6	1	20	3
6	HK_6	pre_param.txt	7	1	20	3
7	HK_7	pre_param.txt	8	1	20	3
8	HK_8	pre_param.txt	9	1	20	3
9	HK_9	pre_param.txt	10	1	20	3
10	HK_10	pre_param.txt	11	1	20	3
11	HK_11	pre_param.txt	12	1	20	3
12	HK_12	pre_param.txt	13	1	20	3
13	HK_13	pre_param.txt	14	1	20	3
14	HK_14	pre_param.txt	15	1	20	3

```
15          HK_15       pre_param.txt 16      1       20      3
16          HK_16       pre_param.txt 17      1       20      3
17          HK_17       pre_param.txt 18      1       20      3
18          HK_18       pre_param.txt 19      1       20      3
19          HK_19       pre_param.txt 20      1       20      3
20          HK_20       pre_param.txt 21      1       20      3
21          HK_21       pre_param.txt 22      1       20      3
22          HK_22       pre_param.txt 23      1       20      3
23          R_TRANSP    pre_param.txt 24      1       20      4
24          R_BSMNT     pre_param.txt 25      1       20      4
25          R_CHN       pre_param.txt 26      1       20      4
26          R_CHS       pre_param.txt 27      1       20      4
27          R_EVTR      pre_param.txt 28      1       20      8
28          R_EXTD      pre_param.txt 29      1       20      4
29          R_RECH      pre_param.txt 30      1       20      4
30          R_NOR       pre_param.txt 31      1       20      4
```

ParamRanges.txt

```
ParameterNmbr ParameterName MinValue MaxValue
1             HK_1          0        100
2             HK_2          0        100
3             HK_3          0        100
4             HK_4          0        100
5             HK_5          0        100
6             HK_6          0        100
7             HK_7          0        100
8             HK_8          0        100
9             HK_9          0        100
10            HK_10         0        100
11            HK_11         0        100
12            HK_12         0        100
13            HK_13         0        100
14            HK_14         0        100
15            HK_15         0        100
16            HK_16         0        100
17            HK_17         0        100
18            HK_18         0        100
19            HK_19         0        100
20            HK_20         0        100
21            HK_21         0        100
22            HK_22         0        100
23            R_TRANSP      0        172800
24            R_BSMNT       0        172800
25            R_CHN         1075     1120
26            R_CHS         875      920
27            R_EVTR        0        0.01
28            R_EXTD        0        20
29            R_RECH        0        345600
30            R_NOR         0        86400
```

6.4 Basic I/O Wrapper Functions

For a successful calibration a number of observations are required for comparison against the simulated equivalents and, as such, they constitute the basis for the calculation of the objective function. For MF2005, we can define observations for a series of flow processes, e.g. specified-head flows, drains, rivers, groundwater heads, streams, etc., see *Hill et al.* (2000). For model

M2, however, we employ 42 groundwater head observations contained in the M2.HOB file, which are read through a simple R function (*read.hobs.R*). As explained later, *read.hobs.R* is called only once and is not necessarily required. We include it to illustrate how additional observations for other flow components can be read, stored in an R object and used for calculating the objective function.

read.hobs.R

```
read.hobs <- function(fname="M2.HOB",ncol=9,skip=2) {
  x <- read.table(file=fname,skip=skip,header=FALSE)
  x <- x[,ncol]
  return(x) }
```

The main I/O function to interface *hydroPSO* and M2 (MF2005/ZB) is *read.hsim.R*. This function reads simulated groundwater heads, global groundwater balance components and residuals, and calculates a customised goodness-of-fit measure as objective function. For this case, we read both simulated heads and calculated residuals from the M2.LST file. Alternatively, we could read simulated groundwater heads from the M2.HED file (for specific well locations), however, this does not allow us to read in one-step the global groundwater balance components available in the M2.LST file. By directly reading the residuals as reported in the M2.LST, we avoid having to read the groundwater head observations contained in the M2.HOB file.

At the same time, *read.hsim.R* calculates a Gaussian likelihood measure (L) using,

$$L = (2\pi)^{-N/2} |C|^{-1/2} \exp\left(-\frac{1}{2} (h_{sim} - h_{obs})^T C^{-1} (h_{sim} - h_{obs})\right) \qquad (13)$$

where C is the covariance matrix of the observed system variables, N is the number of observations, and the likelihood for the corresponding *hydroPSO* iteration (i.e. parameter set) is obtained using a product inference function (see, e.g., *Rojas et al.*, 2008). Note that alternative formulations can be implemented as objective functions, however, the current version of *hydroPSO* works only with single-objective functions.

Finally, *read.hsim.R* writes the calculated likelihood, which is processed by *hydroPSO* to assess the quality of the particles' positions, and the global groundwater balance components to simple ASCII files, `lik_gauss.txt` and `WBAL.txt`, respectively.

read.hsim.R

```
read.hsim <- function(fname="M2.LST",nobs=42) {
sim <- rep(NA,nobs)
```

```
lik <- NA
out <- rep(NA,7)
L <- 0
x <- readLines(fname)
stg <- " HEAD AND DRAWDOWN OBSERVATIONS"
n <- which(x==stg)
L <- length(n)

if (L > 0) {
  suppressWarnings(tmp <- read.table(file=fname,skip=n+3,header=FALSE,nrows=nobs,colClasses
  =c("NULL","NULL","numeric","numeric"),fill=TRUE,na.strings="OMITTED"))
  sim <- as.numeric(tmp[,1])
  res1 <- as.numeric(tmp[,2])
  na.index <- which(is.na(sim))

  if ((length(sim)==nobs) & (length(na.index)==0)) {
    # Gaussian likelihood (stdev = 10 same for MCMC analysis Rojas et al. 2010)
    gauss1 <- 2*pi
    gauss2 <- 10*sqrt(gauss1)
    gauss3 <- 1/gauss2
    gauss4 <- 2*(10^2)
    res2 <- res1^2
    lik1 <- gauss3*exp(-(res2/gauss4))
    lik2 <- prod(lik1)
    lik <- lik2^(1/nobs) #likelihood using "product" inference function (Rojas et al. 2010)

          system2("zonbud_hydroPSO.exe")
  stg <- "  VOLUMETRIC BUDGET FOR ENTIRE MODEL AT END OF TIME STEP  1 IN STRESS PERIOD   1"
    n <- which(x==stg)
    L <- length(n)
    if (L > 0) {
      suppressWarnings(tmp <- read.table(file=fname,skip=n+11,header=FALSE,nrows=9,fill=TRU
      E,stringsAsFactors=FALSE))
      rech <-as.numeric(tmp[1,3])
      evap <-as.numeric(tmp[8,3])
      transp <-as.numeric(tmp[9,3])
      out1 <- c(rech,evap,transp )
      names(out1) <- c("rech","evap","transp")
      out2 <- read.wbal("M2.BAL")
      out <- c(out1, out2)
    }
  } else {
      sim <- rep(NA,nobs)
      lik <- NA
      out <- rep(NA,7)
    }
}
# Adding the results of the water balance to "WBAL.txt"
wb.Text.file <- file("WBAL.txt","a")
writeLines(as.character(out),wb.Text.file,sep=" ")
writeLines("",wb.Text.file)
close(wb.Text.file)
write(lik,"lik_gauss.txt")
return(sim) }
```

In principle, using *read.hsim.R* and correctly defining the `ParamFiles.txt` file, should suffice for interfacing *hydroPSO* and M2 (MF2005/ZB). However, to consider alternative "goodness-of-fit" measures and/or observations might require using *read.hobs.R*.

In addition, for this tutorial we develop an R script to read the results from the ZB program. *read.wbal.R* reads the results for 4 "budget" zones defined in the M2.ZON file, and saved in the M2.BAL file by `zonebud_hydroPSO`. These flows are recharge due to deep fissures in basement rocks (`rechdeep`), outflows at the aquifer's southern cross-section (`cgordo`), (point) southernmost incoming lateral flux (`chaca`), and outflows to western aquifers (`noria`). These results are stored in an R object (`out`), which is saved in the WBAL.txt file by the *read.hsim.R* script.

read.wbal.R

```
read.wbal <- function(fname="M2.BAL") {
if (length(readLines(fname)) > 34 ) {
  suppressWarnings(rechdeep <- as.numeric(read.table(file=fname,skip=34,header=FALSE,nrows=
  1,colClasses=c("NULL","NULL","numeric"),fill=TRUE)))
  suppressWarnings(cgordo <- as.numeric(read.table(file=fname,skip=42,header=FALSE,nrows=1,
  colClasses=c("NULL","NULL","NULL","NULL","NULL","numeric"),fill=TRUE)))
  noria <- NA
  chaca <- NA

  x <- readLines(fname)
  stg <- "     Flow Budget for Zone  5 at Time Step  1 of Stress Period  1"
  n <- which(x==stg)
  L <- length(n)
  if (L > 0) noria <- read.table(file=fname,skip=n+18,header=FALSE,nrows=1,colClasses=c("NU
  LL","NULL","numeric"),fill=TRUE)

  stg <- "     Flow Budget for Zone 16 at Time Step  1 of Stress Period  1"
  n <- which(x==stg)
  L <- length(n)
  if (L > 0) chaca <- read.table(file=fname,skip=n+8,header=FALSE,nrows=1,colClasses=c("NUL
  L","NULL","numeric"),fill=TRUE)
  out <- c(rechdeep,noria,cgordo,chaca)
} else out <- rep(NA,4)
names(out) <- c("rechdeep","noria","cgordo","chaca")
return(out) }
```

6.5 Implementation Details and Results of the Calibration

1. Once `ParamFiles.txt` and `ParamRanges.txt` files have been created, they need to be stored in the `./PSO.in` directory within the folder containing the main MF2005 model files, which for this tutorial corresponds to `./MF2005`.

2. The arguments for the `hydromod()` function have been defined using `read.hsim` script for the function `out.FUN`, whereas `read.lik` has been used as argument for the `gof.FUN` function.

3. Several auxiliary files (described below) are included in `./MF2005`:

Auxiliary Files in MF2005

```
hydroPSO-MF2005.R -> Main R script to run hydroPSO
lik_gauss.txt -> ASCII file containing the likelihood value
pre_param.txt -> ASCII file containing the updated parameters
TOY_LPF_M2.txt -> Template for the LPF package used by preproc
TOY_RCH_M2.txt -> Template for the RCH package used by preproc
```

4. The setup for the problem as well as all the options implemented to calibrate the groundwater flow model are defined in the `hydroPSO-MF2005.R` script. By default all results from *hydroPSO* are saved into the `PSO.out` folder, however, this can be redefined by using the `drty.out` argument.

hydroPSO-MF2005.R

```
################################################################################################
## Example to interface MODFLOW-2005 and ZONE BUDGET with hydroPSO. This script    #
## allows hydroPSO to take control of the execution of MODFLOW-2005 and ZONEBUDGET #
## through the definition of a batch file (run_me.bat) and a series of simple I/O  #
## R scripts                                                                       #
##                                                                                 #
## Part of the hydroPSO R package                                                  #
## http://www.rforge.net/hydroPSO/ http://cran.r-project.org/web/packages/hydroPSO #
## Copyright 2011-2012 Mauricio Zambrano-Bigiarini & Rodrigo Rojas                 #
## Distributed under GPL 2 or later                                                #
##                                                                                 #
## Created by Mauricio Zambrano-Bigiarini and Rodrigo Rojas. 26-Oct-2011           #
## Last saved: 25-Nov-2012                                                         #
################################################################################################

###Loading required libraries
library(hydroPSO)
library(hydroTSM)
library(hydroGOF)

###Definition of working directory: input, output and model files paths
model.drty <- "~/MF2005"
setwd(model.drty )

###Customised I/O functions (R scripts) to interface MF2005 with hydroPSO
source("read.hobs.R")
source("read.hsim.R")
source("read.wbal.R")
source("read.lik.R")

###Goodness-of-fit, either customised or pre-defined from hydroGOF
gof.FUN <- "read.lik"
gof.FUN.args <- list()

###Getting the OBSERVATIONS (not strictly necessary for this example)
obs.fname <- "M2.HOB"
obs.fname <- paste(file.path(model.drty),"/",obs.fname,sep="")
obs <- read.hobs(fname=obs.fname)

###MAIN model function
model.FUN="hydromod",
model.FUN.args=list(
   model.drty=model.drty,
   param.files=paste(model.drty,"/PSO.in/ParamFiles.txt",sep=""),
```

```
   exe.fname="run_me.bat",
   ###Function for reading the simulated equivalents
   out.FUN="read.hsim",
   out.FUN.args=list(
      fname="M2.LST",
      nobs=42),
  ###Function assessing the simulated equivalents against the observations
  gof.FUN=gof.FUN,
  gof.FUN.args=gof.FUN.args,
  obs=obs
) ###END model.FUN.args

###MAIN PSO ALGORITHM
###For hydroPSO fine-tuning parameters, see Zambrano-Bigiarini and Rojas,2012
#set.seed(1111)
hydroPSO(
   fn="hydromod",
   model.FUN.args=model.FUN.args,
   method="spso2011",
   control=list(
      MinMax="max",
      npart=70,
      maxit=3000,
      reltol=1e-30,
      c1=2.05,
      c2=2.05,
      use.IW=FALSE,
      use.CF=TRUE,
      use.TVlambda=TRUE,TVlambda.type="linear",TVlambda.rng=c(1.0,0.5),TVlambda.exp=1,
      topology="random", k=11,
      drty.out="SPSO2011_i31.out",
      boundary.wall="reflecting",
      normalise=TRUE,
      REPORT=50
   ) ###END control options
) ###END MAIN hydroPSO ALGORITHM
```

In the `hydroPSO-MF2005.R` script we use `read.hobs.R` to read groundwater head observations from M2.HOB, `read.hsim.R` to read simulated equivalents and calculate a likelihood measure, `read.wbal.R` to read groundwater balance components defined in the `M2.ZON` file, and `read.lik.R` to read the one-line ASCII file containing the likelihood measure. When a model code other than pre-defined test functions coded in *hydroPSO* is used, the `fn` argument **must** take the value `hydromod`. The latter will indicate *hydroPSO* to expect for external files containing the model executable(s), input(s) and output files. We use the Standard PSO 2011 algorithm (`method="spso2011"`) to maximise (`MinMax="max"`) the likelihood calculated by `read.hsim.R`. For that purpose, we employ a swarm of 70 particles (`npart=70`) and a maximum number of iterations equals to 3000 (`maxit=3000`). A constant cognition and social coefficients (2.05) and a linear (`TVlambda.type="linear"`) time-variant lambda factor (velocity clamping factor) (`use.TVlambda=TRUE`) between [1.0,0.5] are used for fine-tuning the algorithm. A random

topology with 11 informants and a reflecting boundary condition are implemented. All parameter values are normalise in the range [0;1] before optimisation. As model arguments we define `run_me.bat` as the main code running the (sequence of) model(s) together with the main model outputs (`out.FUN="read.hsim"`) and the likelihood of that particular parameter set (`gof.FUN="read.lik"`). Finally, we save results in folder `SPSO2011_i31.out`.

5. Here, we illustrate the use of individual *hydroPSO* "reading" and "plotting" functions to obtain customised graphs. First, we set up the corresponding directory with the results of the calibration:

```
> setwd("~/MF2005/SPSO2011_i31.out")
```

6. Plotting the evolution of the global optimum:

```
> read_convergence(do.png=TRUE)
```

   Figure 31 shows the evolution of the Gaussian likelihood (customised Goodness-of-Fit measure) as a function of the number of iterations. In general, we see that there is an initial exploratory phase of ca. 500 iterations where significant improvements in the global optimum are found. After iteration nr. 1000, the global optimum stabilises and the NSR seems to converge towards an attraction zone.

7. Plotting the evolution of the 30 parameters. First, we read the file `Particles.txt` and then we call function `plot_ParamsPerIter` for plotting:

```
> parts <- read_params(file="Particles.txt",param.cols=
+                         4:33,plot=FALSE)
> plot_ParamsPerIter(parts[["params"]])
```

   Figure 32 shows the result from the previous command lines. In this figure we see the evolution of the parameters of M2 as a function of the number of model evaluations (i.e. 70 particles $\times$ 3000 iterations). We see that several parameters show a weak convergence to stable values (e.g. HK_1, HK_5, HK_10-15, R_CHN, R_TRANSP, R_EVTR, R_EXTD), indicating a rather insensitive behaviour. On the contrary, other parameters show a clear zone of attraction (e.g. HK_20, R_CHS, R_NOR). On the basis of previous runs, we have verified that the use of the `normalise=TRUE` option for the parameters helps improving the convergence of the parameter values as well as the velocities of the

particles. However, caution must be considered in the case a sensitivity analysis is not performed, as the use of `normalise` could give the impression of stable convergence for "insensitive" parameters.

8. From the previous item it is clear that not all parameters are sensitive. So, here we plot a sub-set of parameters using the options of the `plot_results()` *hydroPSO* function:

```
> plot_results(drty.out=getwd(),MinMax="max",do.png=TRUE,
+              param.names=c("HK_2","R_CHS","R_NOR","HK_19",
+              "HK_16","R_RECH","R_BSMNT","R_CHN","R_TRANSP",
+              "R_EVTR","R_EXTD","HK_1"))
```

   This command line will produce several figures summarizing the results of the calibration for M2 (see, e.g., Section 5.4.3). Figure 33 shows dotty-plots for the subset of parameters identified in Figure 32. Here, we see clear zones of attraction for some parameters (e.g. HK_2, R_CHS, and R_RECH), whereas others show slightly flat (and non-symmetric) likelihood response surfaces. Figure 34, in turn, shows (projected) 2D dotty plots among parameters highlighting non-linear interactions.

   Figure 35 shows the ECDs for the sub-set of parameters of M2, where the most likely parameter value is highlighted. Finally, Figure 36 shows a good correspondence between the best simulated and the observed groundwater heads.

9. At the same time, it is possible to analyse the results defining a "behavioural" threshold for the full set of simulated parameters. Here, we select all simulations with a Gaussian Likelihood greater than 3.8×10^{-2}:

```
> plot_results(drty.out=getwd(),MinMax="max",do.png=TRUE,
+              param.names=c("HK_2","R_CHS","R_NOR","HK_19",
+              "HK_16","R_RECH","R_BSMNT","R_CHN","R_TRANSP",
+              "R_EVTR","R_EXTD","HK_1"),beh.thr=3.8e-2)
```

   Figures 37 and 38 show the full ECDFs for the simulated groundwater heads at 42 observation wells, highlighting the groundwater head observations and the percentage bias for the quantile 50. Figure 39 show the resulting plots for the (projected) 2D interaction among parameters, where a clearer picture of the complex interactions for the zone of attraction can be seen for several parameters (e.g. R_RECH

vs. R_NOR and R_RECH vs. HK_2). At the same time, Figure 40 shows the resulting ECDFs for a behavioural threshold of 3.8×10^{-2}.

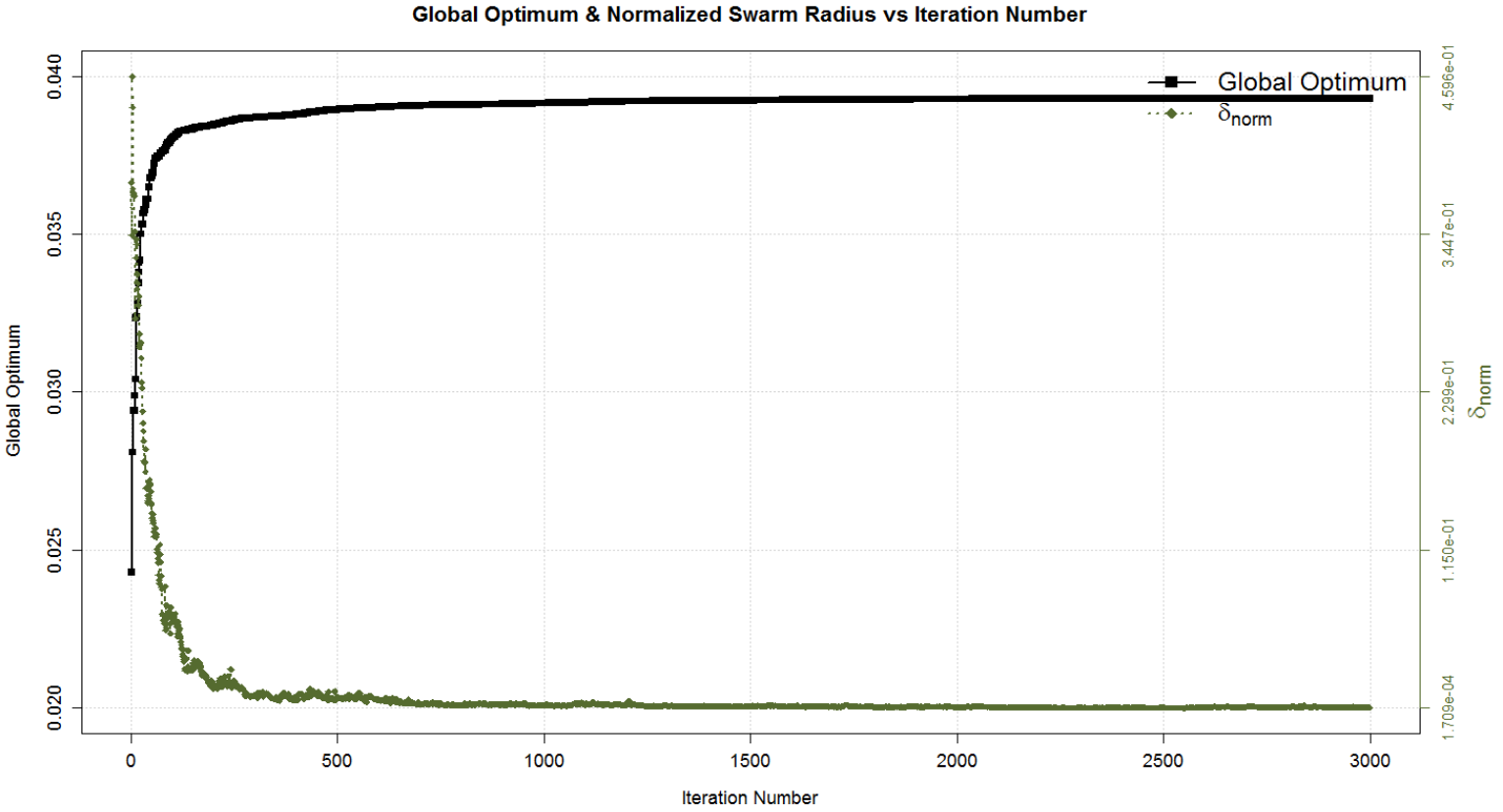

Figure 31: Evolution of the Global Optimum and the Normalised Swarm Radius (NSR) versus iteration number for the calibration of the Ega headwater catchment.

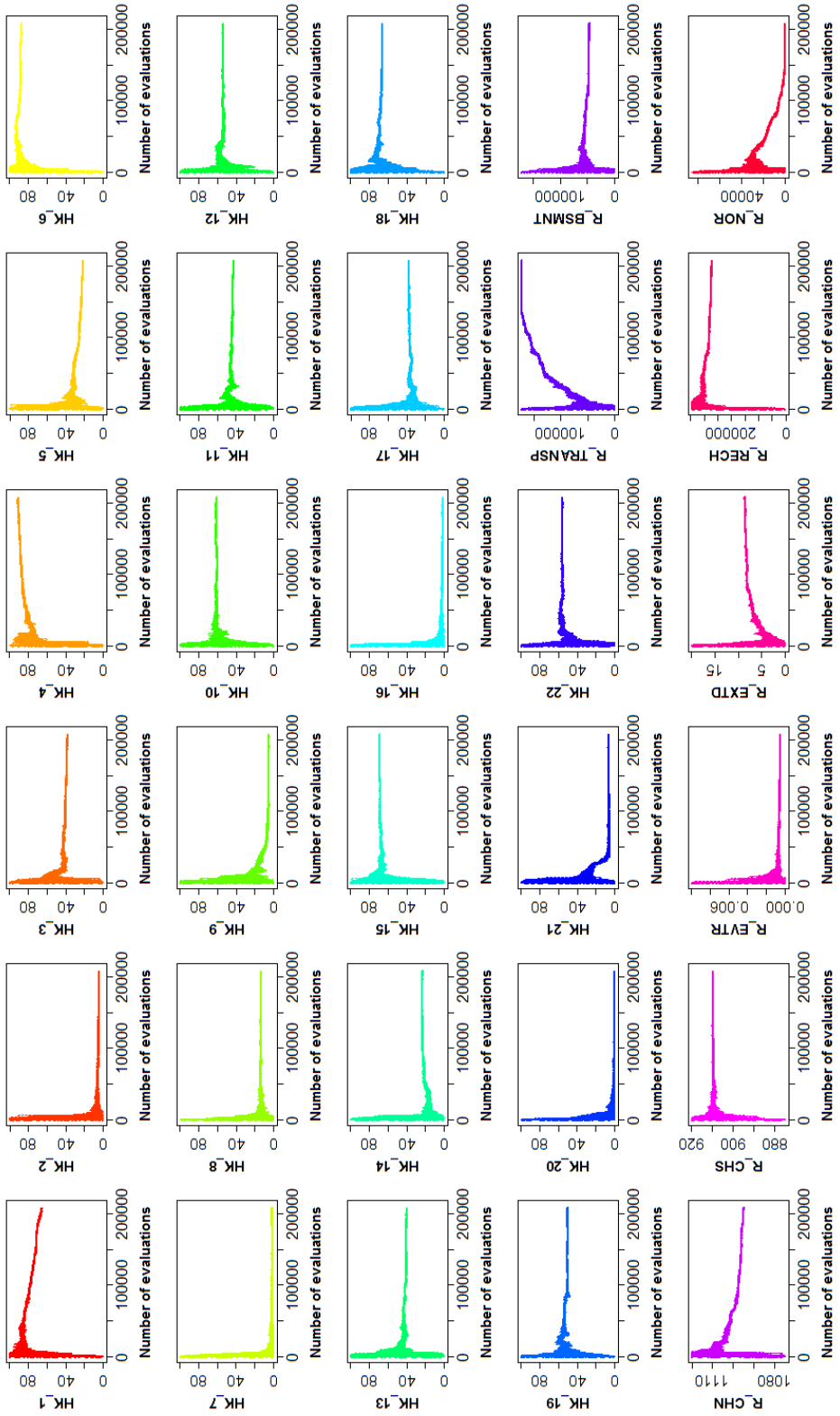

Figure 32: Convergence of the different MF2005 parameters as a function of the iteration number.

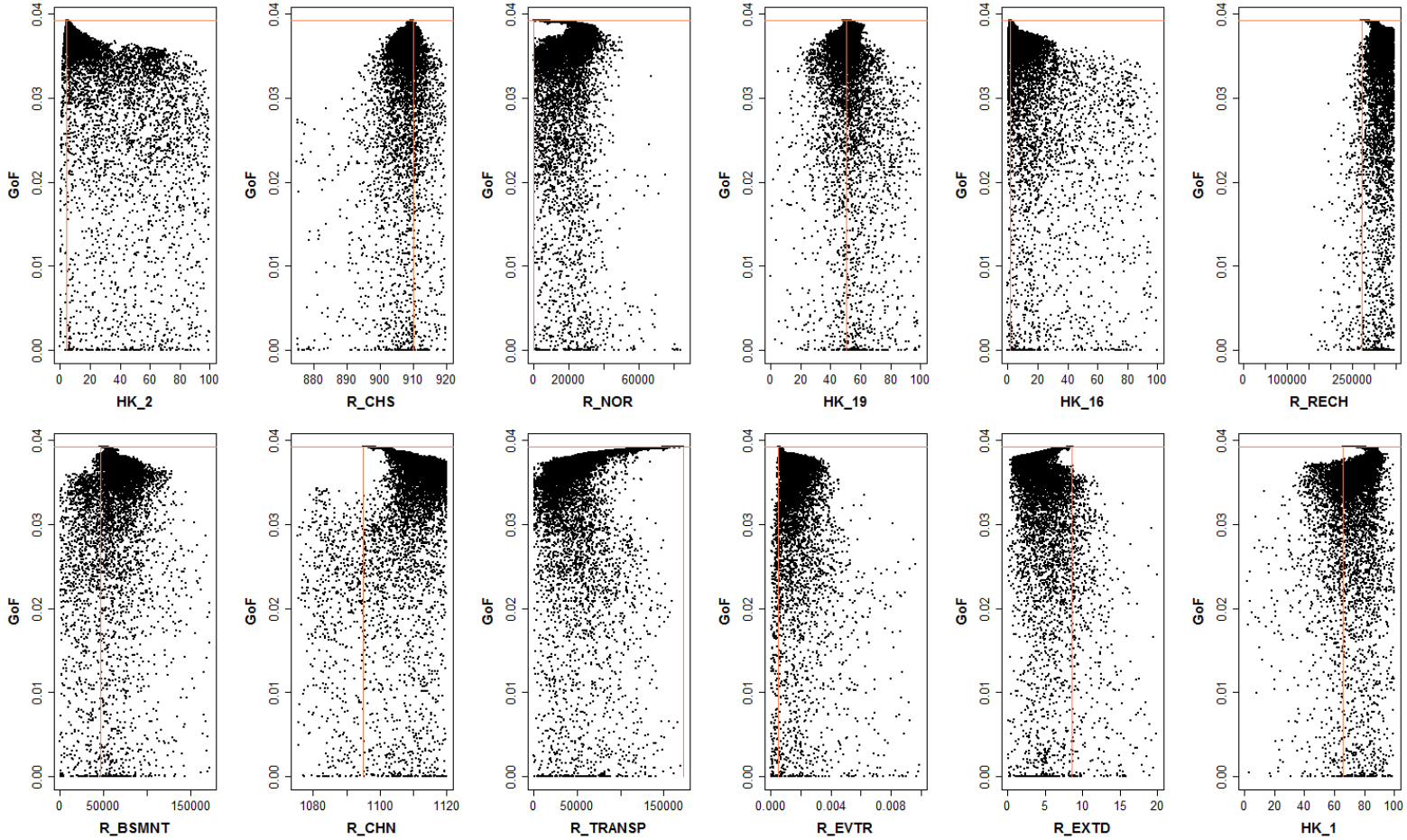

Figure 33: Dotty-plots for the sensitive parameters obtained from Figure 32 for M2. Vertical and horizontal coloured lines show the location of the highest goodness-of-fit value (Gaussian Likelihood).

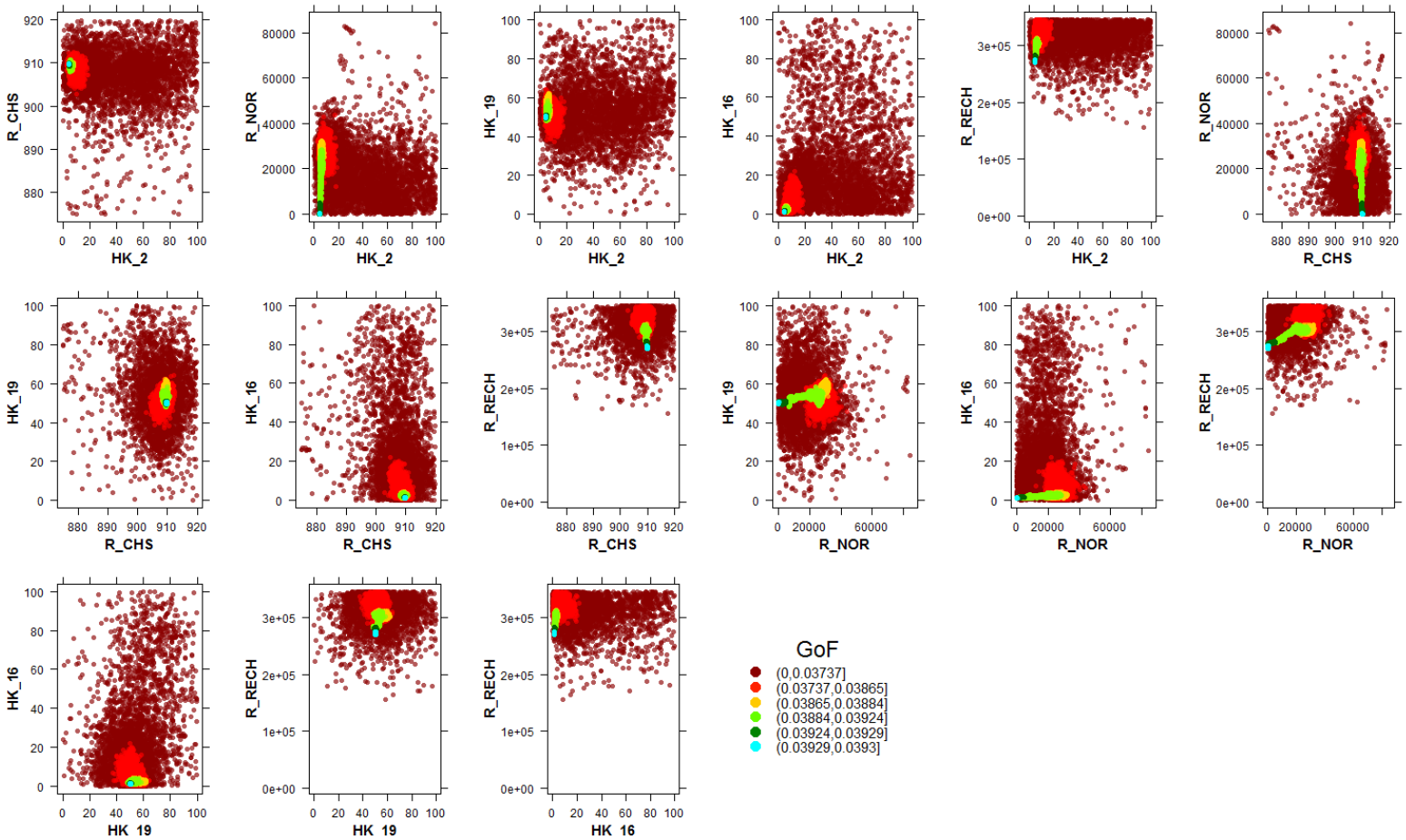

Figure 34: 2-dimensional projected dotty-plots highlighting the interaction among the first seven sensitive parameters obtained from Figure 32 for M2.

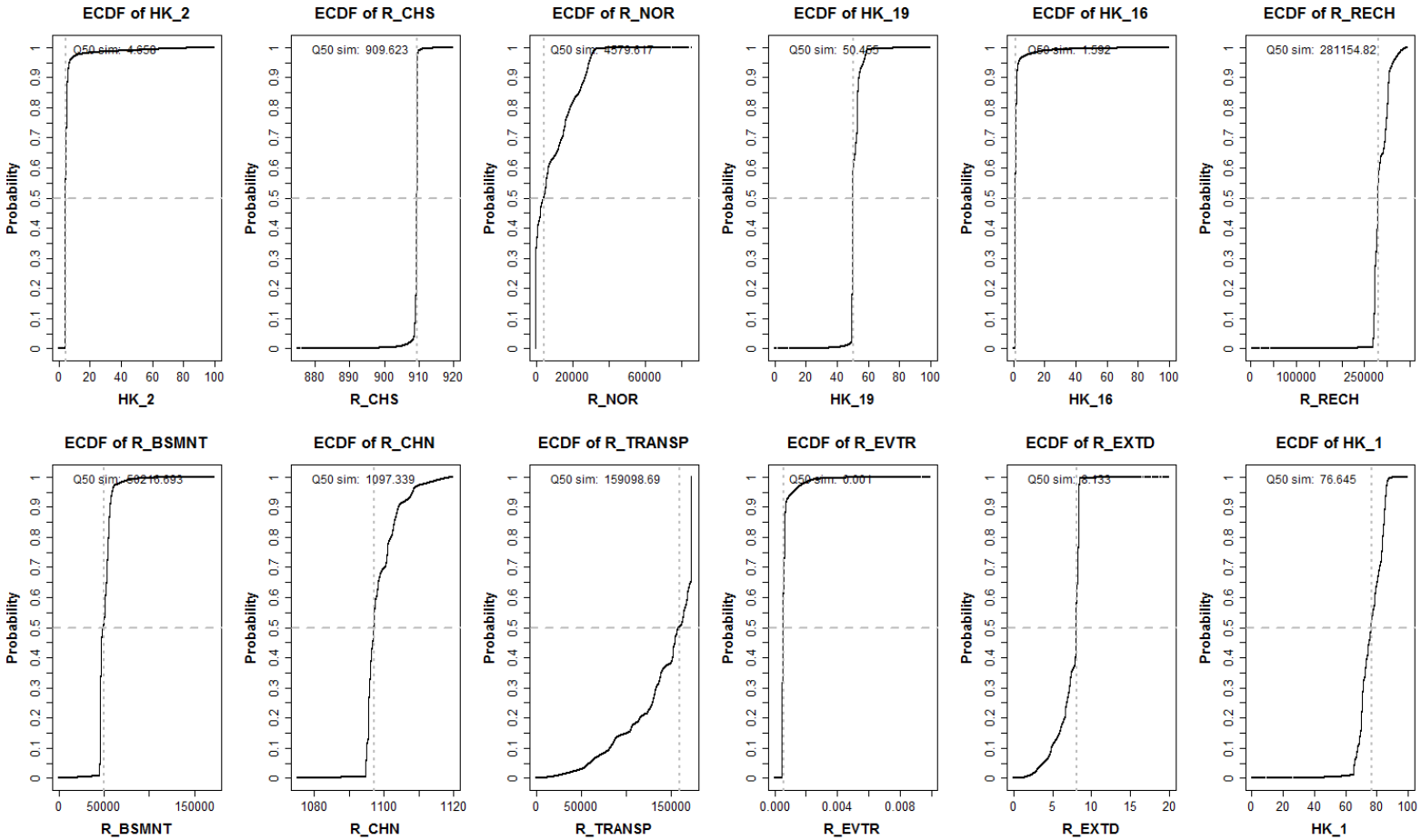

Figure 35: Empirical Cumulative Distribution Functions (ECDFs) for the sensitive parameters obtained from Figure 32 for M2. Vertical and horizontal grey dashed-line indicate the location of a user-defined percentile.

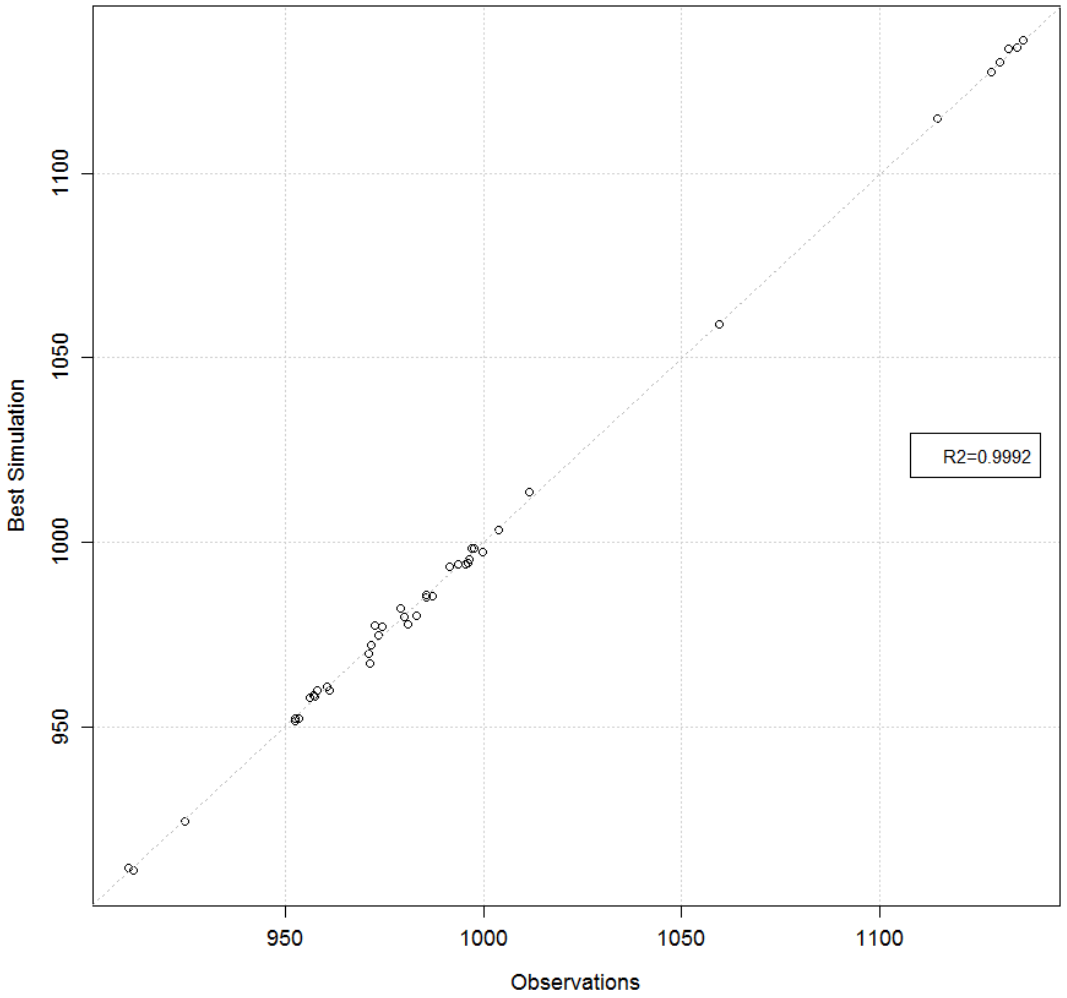

Figure 36: Scatter plot of best simulated versus observed groundwater heads at 42 observation wells for M2.

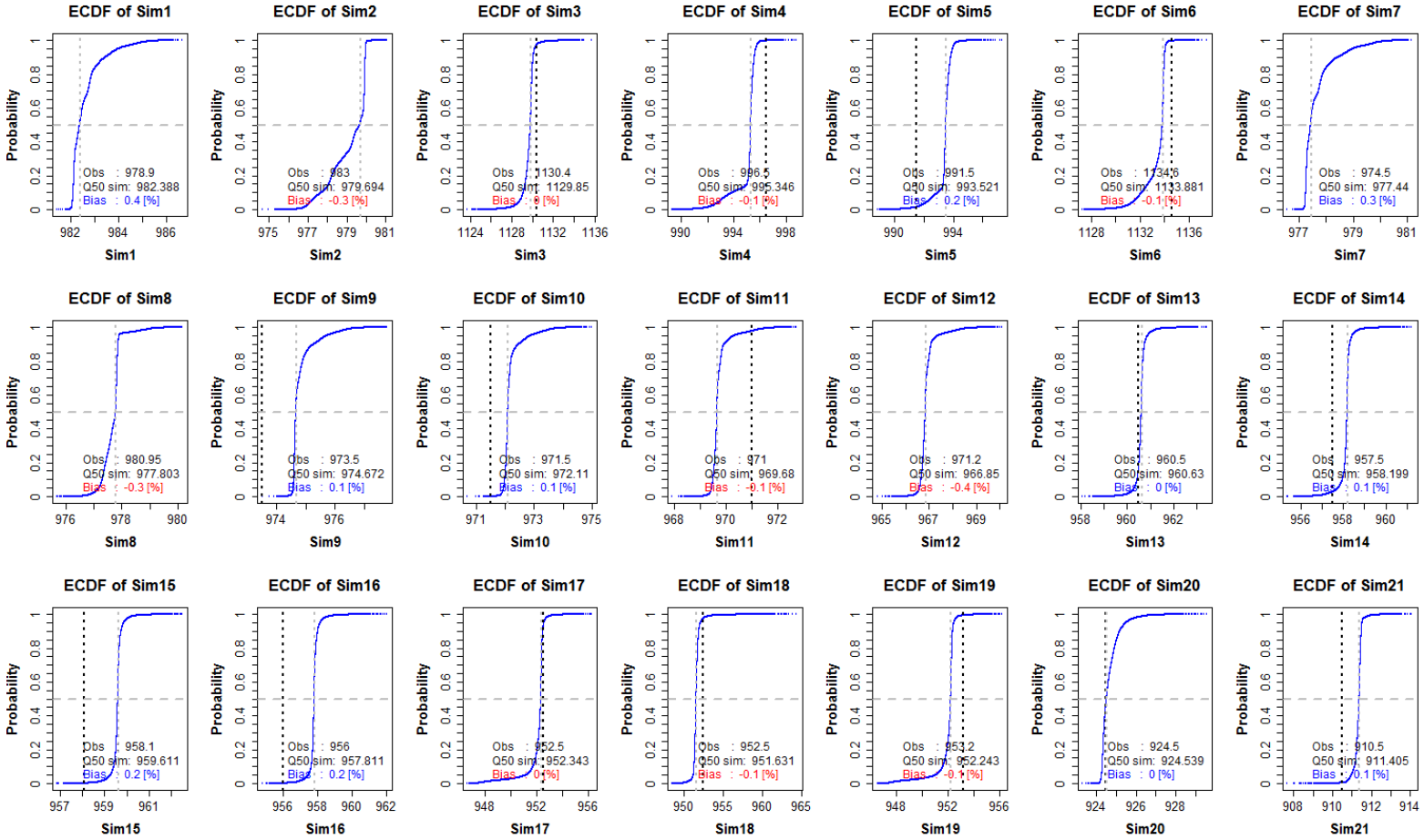

Figure 37: ECDFs for groundwater heads at 42 observation wells for M2. Vertical dashed-line represents the observed value.

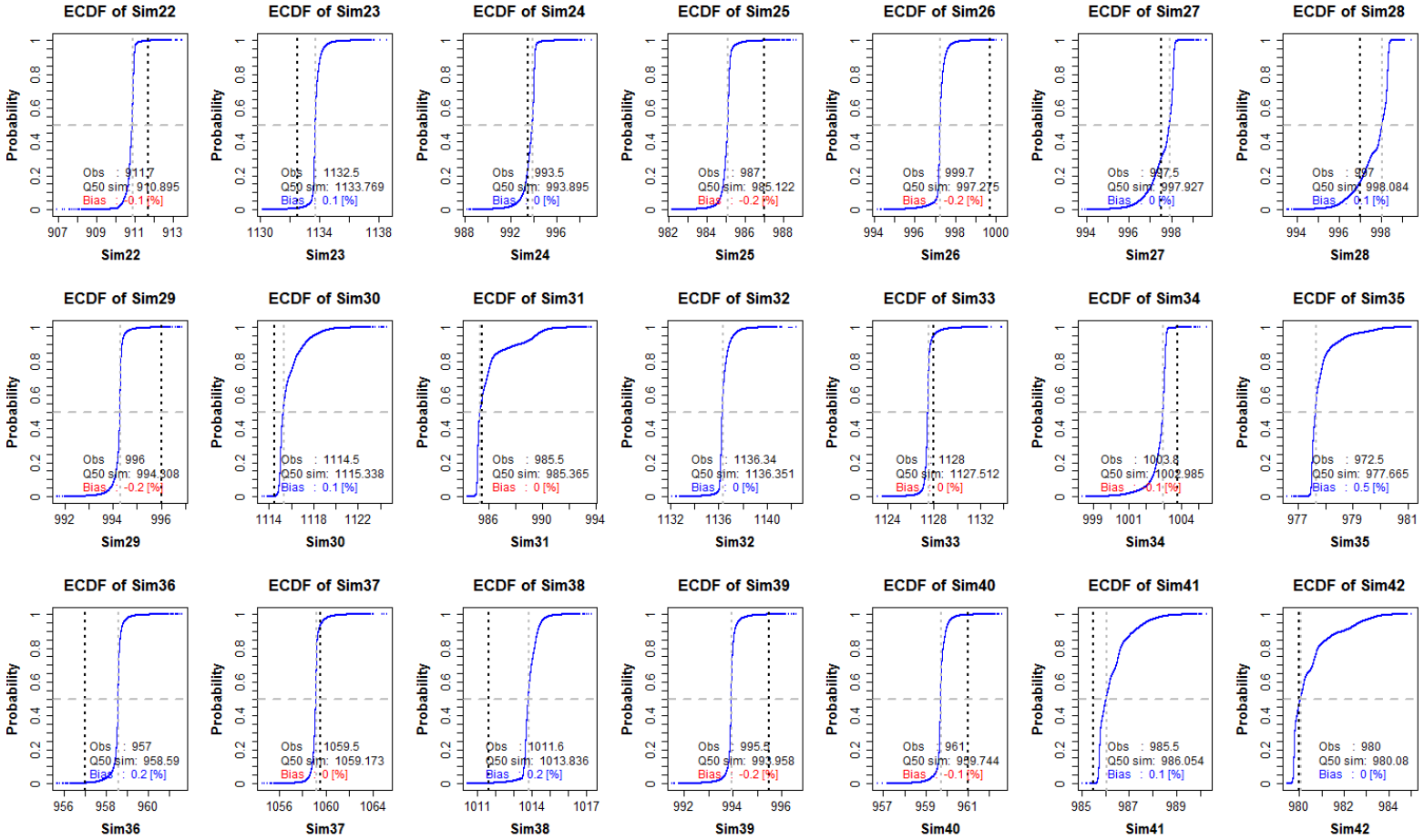

Figure 38: ECDFs for groundwater heads at 42 observation wells for M2. Vertical dashed-line represents the observed value.

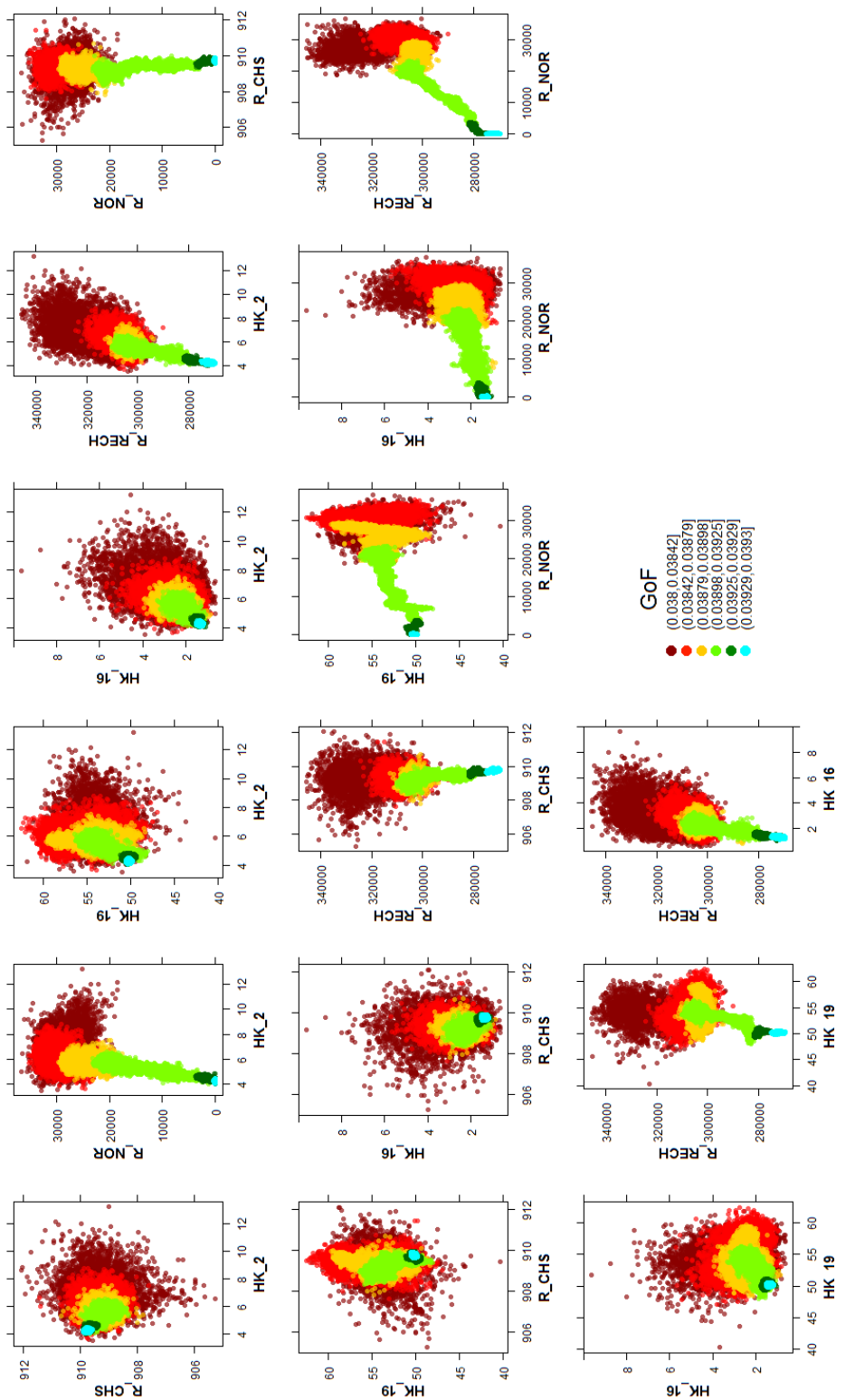

Figure 39: Idem as Figure 34 with a "behavioural" threshold of 3.8×10^{-2}.

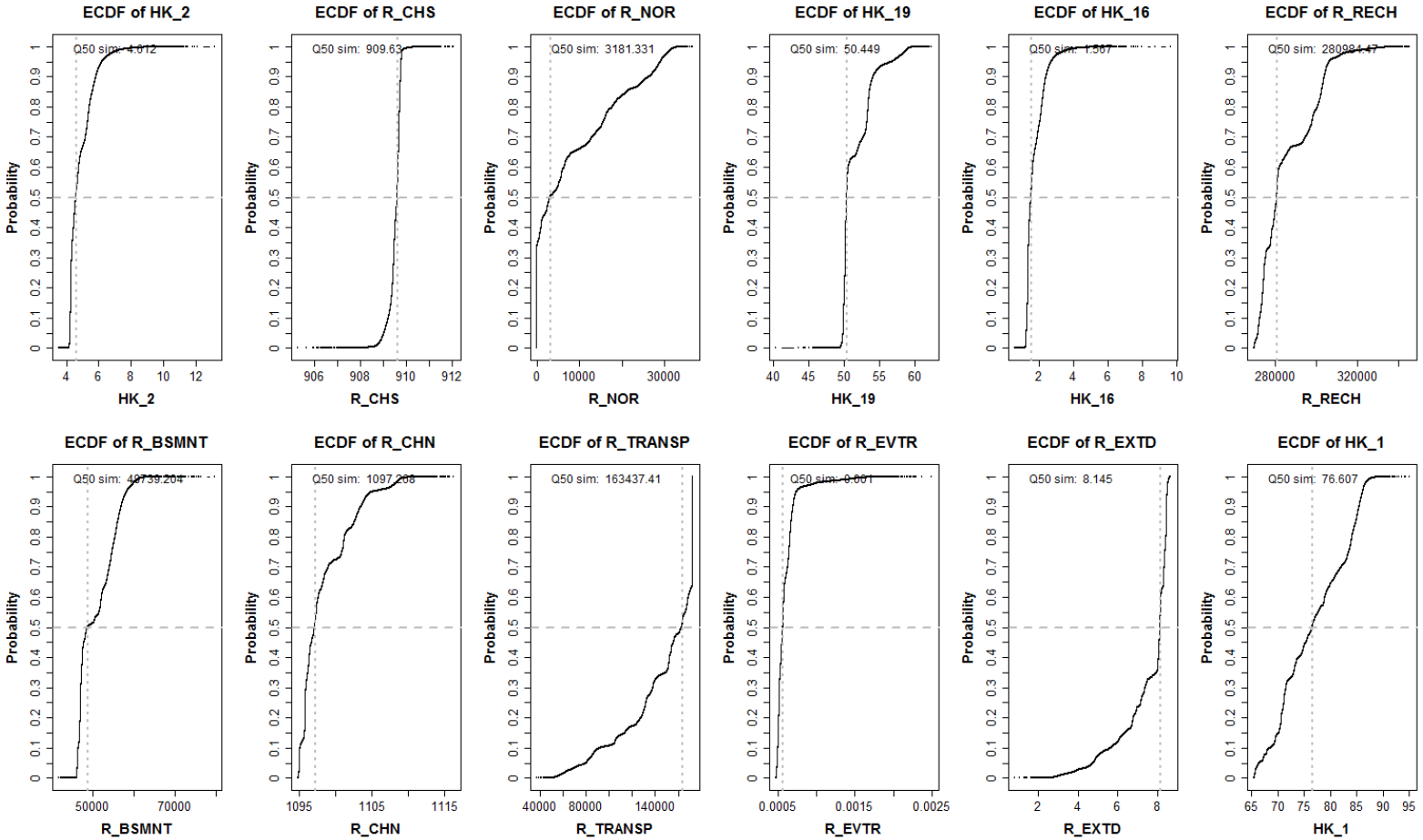

Figure 40: Idem as Figure 35 with a "behavioural" threshold of 3.8×10^{-2}.

7 Interfacing hydroPSO and PEST: `hydroPSO2pest` and `pest2hydroPSO` Functions

Parameter ESTimation (PEST) (*Doherty*, 2010) is probably amongst one of the most used (gradient-based) calibration software packages nowadays. PEST is a calibration tool applicable to any model code reading and writing ASCII files, which can be applied in estimation/prediction/regularization modes, can be used to perform sensitivity and uncertainty analyses, and provides the user with numerous statistics for the assessment of results. Its use is particularly relevant in groundwater-related sciences, with an increasing user community outside this domain (see, e.g., http://www.pesthomepage.org/).

hydroPSO includes two functions allowing to interface *hydroPSO* with PEST, namely, `hydroPSO2pest` and `pest2hydroPSO`.

1. Function `hydroPSO2pest` will convert *hydroPSO* files (`ParamRanges.txt`, `ParamFiles.txt`, and `hydroPSO-script.R`) into the basic files used to perform a PEST calibration (i.e. *.pst, *.ins, and *.tpl). Here, we illustrate how `hydroPSO2pest` is called for the SWAT-2005 case study assuming that a successful optimisation with *hydroPSO* has been achieved. We define as working directory the folder where all the files for applying SWAT-2005 and *hydroPSO* are located:

   ```
   > getwd()

   > setwd(dirname(getwd()))
   > setwd("SWAT2005")
   ```

2. Obtaining information about the arguments required by the `hydroPSO2pest` function:

   ```
   > ?hydroPSO2pest
   ```

3. If located in the correct working directory, the user only needs to specify the name(s) of the executable file(s) used in the optimisation through the `exe.fname` argument, which in our case is `swat2005.exe` (or `swat2005.out` in GNU/LINUX). `hydroPSO2pest` will search the remaining arguments in the default directories automatically written by *hydroPSO*, i.e. `/PSO.in` and `/PSO.out`.

   ```
   > hydroPSO2pest(exe.fname="swat2005.exe")
   ```

4. `hydroPSO2pest` will create the template files (*.tpl), a basic instruction file (*.ins) to be completed by the user, and the controlling file (*.pst). We must notice that these templates must be verified/modified by the user for a successful PEST application, given the fact that PEST uses "token-based" instructions for reading model inputs and outputs.

5. Function `pest2hydroPSO` will convert PEST files (*.pst, *.ins, and *.tpl) into the basic files used to perform a *hydroPSO* optimisation (i.e. `ParamRanges.txt`, `ParamFiles.txt`, and `hydroPSO-Rscript.R`). Here, we illustrate how this function is called for the SWAT-2005 case study assuming that a successful calibration with PEST has been achieved. We keep the working directory where all the files for applying SWAT-2005 and PEST are located. To obtain information about the arguments required by `pest2hydroPSO` type:

```
> ?pest2hydroPSO
```

6. If located in the right directory, the user only needs to define the name of the controlling file for the PEST calibration (in our example this is called `SWAT2005_EGA.pst`). Then, `pest2hydroPSO` will create two folders, namely, `/PSO.in` and `/Rscripts`, where the files required by *hydroPSO* will be saved (to avoid overwriting of the original *hydroPSO* files we define a new output directory `drty.out="pest2hydroPSO.in"`):

```
> pest2hydroPSO(pst.fname="SWAT2005_EGA.pst",
+               drty.out="pest2hydroPSO.in")
```

7. `ParamRanges.txt` and `ParamFiles.txt` files will be saved in `drty.out`, whereas `hydroPSO-Rscript.R` will be saved in `/Rscripts`. `hydroPSO-Rscript.R` is a basic R script that must be modified by the user to define a Goodness-of-Fit measure (`gof.FUN`), period of analysis (`Sim.Ini`, `Sim.Fin`, `Gof.Ini`, `Gof.Fin`), and a function to read the model outputs (`out.FUN`).

8 Parallel Configuration

From version 0.3-0 *hydroPSO* can be executed taking advantage of multi-core machines or clusters by defining only three arguments: `parallel`, `par.nnodes`, and `par.pkgs`.

`parallel` indicates how to parallelise *hydroPSO*, strictly speaking only the evaluation of the objective function `fn` is parallelised. Available options are:

1. `multicore`: parallel computations for machines with multiple cores or CPUs. The evaluation of the objective function `fn` is done with the `mclapply` function of the *multicore* R package. It requires POSIX-compliant OS (essentially anything but Windows).

2. `parallel`: parallel computations for network clusters or machines with multiple cores or CPUs. A 'FORK' cluster is created with the `makeForkCluster` function. The evaluation of the objective function `fn` is done by using with the `clusterApply` function of the *parallel* R package.

3. `parallelWin`: parallel computations for network clusters or machines with multiple cores or CPUs (this is the only available option for Windows machines). A 'PSOCK' cluster is created with the `makeCluster` function. The evaluation of the objective function `fn` is done with the `clusterApply` function of the*parallel* R package.

`par.nnodes` indicates the number of cores/CPUs to be used in the local multi-core machine, or the number of nodes to be used in the network cluster. By default `par.nnodes` is set to the number of cores detected by the function `detectCores()` obtained from the *multicore* or *parallel* R packages.

Finally, `par.pkgs` (used only when `parallel=='parallelWin'`) is a list of package names that need to be loaded on each node in order to be used by the objective function `fn`.

9 Closing Remarks

1. This tutorial aimed at providing a general overview of the capabilities of the *hydroPSO* R package. In particular, we illustrated how to optimise commonly used test functions and how to calibrate two model codes commonly used in hydrology-related applications, namely, SWAT-2005 and MODFLOW-2005. Given the flexibility of *hydroPSO*, however, we believe this calibration/optimisation engine can be applied to a wider range of environmental models requiring some form of parameter estimation.

2. The version of *hydroPSO* used in this tutorial takes advantage of multi-core machines or network clusters to alleviate the computational burden.

3. In *hydroPSO* we consider only single-objective functions for optimisation. We hope in the next future to include multi-objective functionalities to the main algorithm.

4. *hydroPSO* has been validated against standard global optimisation software such as SCE_UA (*Duan et al.*, 1992), DEoptim (*Mullen et al.*, 2011), and the SPSO-2011 (*Clerc*, 2012). Results indicate that *hydroPSO* is competitive in terms of computational efficiency and effectiveness to find the global optimum for a series of benchmarking functions.

5. Finally, investigation on improvements to the canonical PSO algorithm is one of the most dynamic and active research areas in Particle Swarm literature. Therefore, we are particularly interested in constantly updating the *hydroPSO* package according to the most relevant innovations in this research domain. Authors would greatly appreciate any feedback from the users, which can be provided through the *hydroPSO* rforge webpage (`http://www.rforge.net/hydroPSO/`).

This tutorial was built under:

```
[1] "hydroPSO 0.2-2"
```

```
[1] "R version 2.15.1 (2012-06-22)"
```

```
[1] "i386-pc-mingw32/i386 (32-bit)"
```

References

Andrews, F., B. Croke, and A. Jakeman (2011), An open software environment for hydrological model assessment and development, *Environmental Modelling & Software*, *26*(10), 1171–1185, doi: 10.1016/j.envsoft.2011.04.006.

Arnold, J., and N. Fohrer (2005), SWAT2000: current capabilities and research opportunities in applied watershed modelling, *Hydrological Processes*, *19*(3), 563–572, doi: 10.1002/hyp.5611.

Arnold, J., R. Srinivasan, R. Muttiah, and J. Williams (1998), Large area hydrologic modeling and assessment part I: Model development, *Journal of the American Water Resources Association*, *34*(1), 73–89.

Arumugam, M., and M. Rao (2008), On the improved performances of the particle swarm optimization algorithms with adaptive parameters, crossover operators and root mean square (RMS) variants for computing optimal control of a class of hybrid systems, *Applied Soft Computing*, *8*(1), 324–336, doi: 10.1016/j.asoc.2007.01.010.

Beven, K. (2006), A manifesto for the equifinality thesis, *Journal of Hydrology*, *320*(1–2), 18–36, doi: 10.1016/j.jhydrol.2005.07.007.

Chatterjee, A., and P. Siarry (2006), Nonlinear inertia weight variation for dynamic adpatation in particle swarm optimization, *Computers & Operations Research*, *33*(3), 859–871, doi: 10.1016/j.cor.2004.08.012.

CHE (2000), Los aprovechamientos en la cuenca del ebro: Afección en el régimen hydrológico fluvial -2000-ph-24i [the exploitation in the ebro basin: Alterations to the hydrological regime of the river - 2000-ph-24i], *Tech. rep.*, Confederación Hidrográfica del Ebro, available on http://oph.chebro.es/DOCUMENTACION/Hidrologicos/RegimenHidrologico.htm. [last accessed Oct-2011].

Chen, T., and T. Chi (2010), On the improvements of the particle swarm optimization algorithm, *Advances in Engineering Software*, *41*(2), 229–239, doi: 10.1016/j.advengsoft.2009.08.003.

Clerc, M. (2006), *Particle Swarm Optimization*, ISTE (International Scientific and Technical Encyclopedia).

Clerc, M. (2007), Back to random topology, *Tech. rep.*, Particle Swarm Central.

Clerc, M. (2009), A method to improve Standard PSO, *Tech. Rep. MC2009-03-13*, Particle Swarm Central.

Clerc, M. (2012), Standard Particle Swarm Optimisation, *Tech. rep.*, Particle Swarm Central, [Online; last accessed 24-Sep-2012].

Clerc, M., and J. Kennedy (2002), The particle swarm - explosion, stability, and convergence in a multidimensional complex space, *IEEE Transactions on Evolutionary Computation*, *6*, 58–73, doi: 10.1109/4235.985692.

Di Luzio, M., R. Srinivasan, and J. Arnold (2004), A GIS-coupled hydrological model system for the watershed assessment of agricultural nonpoint and point sources of pollution, *Transactions in GIS*, *8*(1), 113–136, doi: 10.1111/j.1467-9671.2004.00170.x.

Doherty, J. (2010), *PEST: Model-independent parameter estimation. User manual: 5th edition*, Watermark Numerical Computing, fifth ed.

Duan, Q., S. Sorooshian, and H. Gupta (1992), Effective and efficient global optimization for conceptual ranfall–runoff models, *Water Resources Research*, *28*(4), 1015–1031, doi: 10.1029/91WR02985.

Duan, Q., S. Sorooshian, and V. Gupta (1994), Optimal use of the SCE-UA global optimization method for calibrating watershed models, *Journal of Hydrology*, *158*, 265–284, doi: 10.1016/0022-1694(94)90057-4, gives recommended values for the use of the SCE-UA algorithm:.

Eberhart, R., and J. Kennedy (1995), A new optimizer using particle swarm theory, in *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, pp. 39–43, doi: 10.1109/MHS.1995.494215.

Eberhart, R., and Y. Shi (1998), Comparison between genetic algorithms and particle swarm optimization, in *Evolutionary Programming VII*, vol. 1447, edited by V. Porto, N. Saravanan, D. Waagen, and A. Eiben, pp. 611–616, Springer Berlin / Heidelberg, doi: 10.1007/BFb0040812.

Eberhart, R., and Y. Shi (2000), Comparing inertia weights and constriction factors in particle swarm optimization, in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1, pp. 84–88, doi: 10.1109/CEC.2000.870279.

Eberhart, R., and Y. Shi (2001), Particle swarm optimization: developments, applications and resources, in *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 1, pp. 81–86, doi: 10.1109/CEC.2001.934374.

Evers, G., and M. Ghalia (2009), Regrouping particle swarm optimization: A new global optimization algorithm with improved performance consistency across benchmarks, in *IEEE International Conference on Systems, Man and Cybernetics, 2009. SMC 2009*, pp. 3901–3908, doi: 10.1109/ICSMC.2009.5346625.

Gelman, A., J. Carlin, H. Stern, and D. Rubin (2004), *Bayesian data analysis*, second ed., 696 pp., Chapman & Hall/CRC, New York.

Harbaugh, A. (1990), A computer program for calculating subregional water budgets using results from the u.s. geological survey modular three-dimensional ground-water flow model, *Open file report 90-392*, United States Geological Survey, Reston, Virginia, USA.

Harbaugh, A. (2005), MODFLOW–2005, the U.S. Geological Survey modular ground-water model–the Ground–Water Flow Process, *Techniques and methods 6–a16*, United States Geological Survey, Reston, Virginia, USA.

Hill, M., E. Banta, A. Harbaugh, and E. Anderman (2000), MODFLOW–2000. the U.S. Geological Survey modular ground-water model–User guide to the observation, sensitivity, and parameter–estimation processes and three post processing programs, *Open-file report 00-184*, United States Geological Survey, Denver, Colorado, USA.

Holvoet, K., A. van Griensven, P. Seuntjens, and P. Vanrolleghem (2005), Sensitivity analysis for hydrology and pesticide supply towards the river in swat, *Physics and Chemistry of the Earth, Parts A/B/C*, *30*(8–10), 518–526, doi: 10.1016/j.pce.2005.07.006.

Huang, T., and A. Mohan (2005), A hybrid boundary condition for robust particle swarm optimization, *Antennas and Wireless Propagation Letters*, *4*(1), 112–117, doi: 10.1109/LAWP.2005.846166.

Kannan, N., S. White, F. Worrall, and M. Whelan (2007), Sensitivity analysis and identification of the best evapotranspiration and runoff options for hydrological modelling in SWAT-2000, *Journal of Hydrology*, *332*(3-4), 456–466, doi: 10.1016/j.jhydrol.2006.08.001.

Kennedy, J. (1999), Small worlds and mega-minds: effects of neighborhood topology onparticle swarm performance, in *Proceedings of the 1999 Congress on Evolutionary Computation, 1999*, vol. 3, pp. 3 vol. (xxxvii+2348), doi: 10.1109/CEC.1999.785509.

Kennedy, J. (2006), Swarm intelligence, in *Handbook of Nature-Inspired and Innovative Computing*, edited by A. Zomaya, pp. 187–219, Springer US, doi: 10.1007/0-387-27705-6_6.

Kennedy, J., and R. Eberhart (1995), Particle swarm optimization, in *Proceedings IEEE International Conference on Neural Networks, 1995*, vol. 4, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

Kennedy, J., and R. Mendes (2002), Population structure and particle swarm performance, in *Proceedings of the 2002 Congress on Evolutionary Computation, CEC '02*, pp. 1671–1676, doi: 10.1109/CEC.2002.1004493.

Kennedy, J., and R. Mendes (2003), Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms, in *Proceedings of the 2003 IEEE International Workshop on Soft Computing in Industrial Applications, 2003. SMCia/03.*, pp. 45–50, doi: 10.1109/SMCIA.2003.1231342.

Liu, B., L. Wang, Y. Jin, F. Tang, and D. Huang (2005), Improved particle swarm optimization combined with chaos, *Chaos, Solitons & Fractals*, *25*(5), 1261, doi: 10.1016/j.chaos.2004.11.095.

Matott, L. (2005), *Ostrich: An optimization software tool, Documentation and User's Guide, Version 1.6*, Department of Civil, Structural and Environmental Engineering, University at Buffalo, Buffalo, NY.

McKay, D., R. Beckman, and W. Conover (1979), A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, *Technometrics*, *21*(2), 239–245.

Mendes, R. (2004), Population topologies and their influence in particle swarm performance, Ph.D. thesis, Departamento de Informática. Escola de Engenharia. Universidade do Minho, Minho, Portugal.

Mendes, R., J. Kennedy, and J. Neves (2004), The Fully Informed Particle swarm: Simpler, maybe better, *IEEE Transactions on Evolutionary Computation*, *8*(3), 204–210, doi: 10.1109/TEVC.2004.826074.

Muleta, M., and J. Nicklow (2005), Sensitivity and uncertainty analysis coupled with automatic calibration for a distributed watershed model, *Journal of Hydrology*, *306*(1-4), 127–145, doi: 10.1016/j.jhydrol.2004.09.005.

Mullen, K. M., D. Ardia, D. L. Gil, D. Windover, and J. Cline (2011), Deoptim: An r package for global optimization by differential evolution, *Journal of Statistical Software*, *40*(6), 1–26, doi: http://www.jstatsoft.org/v40/i06.

Mussi, L., S. Cagnoni, and F. Daolio (2009), Empirical assessment of the effects of update synchronization in particle swarm optimization, in *Proceedings of the 2009 AIIA Workshop on Complexity, Evolution and Emergent Intelligence*, vol. 1, pp. 1–10, published on CD, isbn 978-88-903581-1-1.

Nash, J., and J. Sutcliffe (1970), River flow forecasting through conceptual models. Part I–A discussion of principles, *Journal of Hydrology*, *10*(3), 282–290, doi: 10.1016/0022-1694(70)90255-6.

Neitsch, S., J. Arnold, J. Kiniry, R. Srinivasan, and J. Williams (2005), *Soil and Water Assessment Tool Theoretical Documentation Version 2005*, Grassland, Soil and Water Research Laboratory; Agricultural Research Service 808 East Blackland Road; Temple, Texas 76502; Blackland Research & Extension Center; Texas Agricultural Experiment Station 720 East Blackland Road Temple, Texas 76502, USA.

Poeter, E., M. Hill, E. Banta, S. Mehl, and S. Christensen (2005), UCODE–2005 and six other computer codes for universal sensitivity analysis, calibration, and uncertainty evaluation, *Technical methods 6–a11*, United States Geological Survey, Reston, Virginia, USA.

Poli, R. (2008), Analysis of the publications on the applications of particle swarm optimisation, *Journal of Artificial Evolution and Applications*, *2008*(685175), 1–10, doi: 10.1155/2008/685175.

Poli, R., J. Kennedy, and T. Blackwell (2007), Particle swarm optimization, *Swarm Intelligence*, *1*(1), 33–57, doi: 10.1007/s11721-007-0002-0.

R Development Core Team (2011), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0.

Ratnaweera, A., S. Halgamuge, and H. Watson (2004), Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, *IEEE Transactions on Evolutionary Computation*, *8*(3), 240–255, doi: 10.1109/TEVC.2004.826071.

Robinson, J., and Y. Rahmat-Samii (2004), Particle swarm optimization in electromagnetics, *IEEE Transactions on Antennas and Propagation*, *52*(2), 397–407, doi: 10.1109/TAP.2004.823969.

Rojas, R., and A. Dassargues (2007), Groundwater flow modelling of the regional aquifer of the Pampa del Tamarugal, northern Chile, *Hydrogeology Journal*, *15*(3), 537–551, doi: 10.1007/s10040-006-0084-6.

Rojas, R., L. Feyen, and A. Dassargues (2008), Conceptual model uncertainty in groundwater modeling: Combining generalized likelihood uncertainty estimation and Bayesian model averaging, *Water Resources Research*, *44*(12), W12,418, doi: 10.1029/2008WR006908.

Rojas, R., O. Batelaan, L. Feyen, and A. Dassargues (2010), Assessment of conceptual model uncertainty for the regional aquifer Pampa del Tamarugal–North Chile, *Hydrology and Earth System Sciences*, *14*(2), 171–192, doi: 10.5194/hess-14-171-2010.

Shi, Y., and R. Eberhart (1998), A modified particle swarm optimizer, in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pp. 69–73, doi: 10.1109/ICEC.1998.699146.

Shi, Y., and R. Eberhart (1999), Empirical study of particle swarm optimization, in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, p. 1945, doi: 10.1109/CEC.1999.785511.

van den Bergh, F., and A. Engelbrecht (2006), A study of particle swarm optimization particle trajectories, *Information Sciences*, *176*(8), 937–971, doi: 10.1016/j.ins.2005.02.003.

van Griensven, A., T. Meixner, S. Grunwald, T. Bishop, M. Diluzio, and R. Srinivasan (2006), A global sensitivity analysis tool for the parameters of multi-variable catchment models, *Journal of Hydrology*, *324*(1–4), 10–23, doi: 10.1016/j.jhydrol.2005.09.008.

van Liew, M., J. Arnold, and D. Bosch (2005), Problems and potential of autocalibrating a hydrologic model, *Transactions of the ASAE*, *48*(3), 1025–1040.

van Liew, M. W., T. L. Veith, D. D. Bosch, and J. G. Arnold (2007), Suitability of SWAT for the Conservation Effects Assessment Project: Comparison on USDA Agricultural Research Service watersheds, *Journal of Hydrologic Engineering*, *12*(2), 173–189, doi: 10.1061/(ASCE)1084-0699(2007)12:2(173).

Vesselinov, V., and D. Harp (2012), Model analysis and decision support (MADS) for comple physics models, in *XIX International Conference on Water Resources - CMWR 2012*, University of Illinois Urbana-Champaign, Illinois, USA.

Vrugt, J., C. ter Braak, M. Clark, J. Hyman, and B. Robinson (2008), Treatment of input uncertainty in hydrologic modeling: Doing hydrology backward with Markov Chain Monte Carlo simulation, *Water Resources Research*, *44*, W00B09, doi: 10.1029/2007WR006720.

Wu, Y., and S. Liu (2012), Automating calibration, sensitivity and uncertainty analysis of complex models using the r package flexible modeling environment (fme): Swat as an example, *Environmental Modelling & Software*, *31*(0), 99–109, doi: 10.1016/j.envsoft.2011.11.013.

Zambrano-Bigiarini, M., and R. Rojas (2012), hydroPSO: A model-independent particle swarm optimization software for calibration of environmental models, *Environmental Modelling & Software*, *submitted*.

Zhao, B. (2006), An improved particle swarm optimization algorithm for global numerical optimization, in *Computational Science - ICCS 2006*, *Lecture Notes in Computer Science*, vol. 3991, edited by V. Alexandrov, G. van Albada, P. Sloot, and J. Dongarra, pp. 657–664, Springer-Verlag Berlin Heidelberg.