Computers & Geosciences 32 (2006) 1139–1155

COMPUTERS & GEOSCIENCES

www.elsevier.com/locate/cageo

Application of parallel computing to stochastic parameter estimation in environmental models☆

Jasper A. Vrugt[a,*], Breanndán Ó Nualláin[b], Bruce A. Robinson[a], Willem Bouten[c], Stefan C. Dekker[d], Peter M.A. Sloot[b]

[a]*Earth and Environmental Sciences Division, Los Alamos National Laboratory, Mail Stop T003, Los Alamos, NM 87545, USA*
[b]*Faculty of Sciences, Section Computational Science, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands*
[c]*Faculty of Sciences, Computational Bio- and Physical Geography, University of Amsterdam, Nieuwe Achtergracht 166, 1018 WV Amsterdam, The Netherlands*
[d]*Department of Environmental Sciences, Faculty of Geosciences, Utrecht University, P.O. Box 80.115, 3508 TC Utrecht, The Netherlands*

Received 2 May 2005; received in revised form 25 October 2005; accepted 25 October 2005

Abstract

Parameter estimation or model calibration is a common problem in many areas of process modeling, both in on-line applications such as real-time flood forecasting, and in off-line applications such as the modeling of reaction kinetics and phase equilibrium. The goal is to determine values of model parameters that provide the best fit to measured data, generally based on some type of least-squares or maximum likelihood criterion. Usually, this requires the solution of a non-linear and frequently non-convex optimization problem. In this paper we describe a user-friendly, computationally efficient parallel implementation of the Shuffled Complex Evolution Metropolis (SCEM-UA) global optimization algorithm for stochastic estimation of parameters in environmental models. Our parallel implementation takes better advantage of the computational power of a distributed computer system. Three case studies of increasing complexity demonstrate that parallel parameter estimation results in a considerable time savings when compared with traditional sequential optimization runs. The proposed method therefore provides an ideal means to solve complex optimization problems.

Keywords: Optimization; Model; Hydrology; Bird migration; Octave; Message passing interface

1. Introduction and scope

The field of earth sciences is experiencing rapid changes as a result of the growing understanding of environmental physics, along with recent advances in measurement technologies, and dramatic increases in computing power. More complex, spatially explicit computer models are now possible, allowing for a more realistic representation of systems of interest. The increasing complexity of these models has, however, resulted in a larger number of parameters that must be estimated. While the values of some of these parameters might

☆Code available from vrugt@lanl.gov

*Corresponding author. Tel.: +1 505 667 0404; fax: +1 505 665 8737.

E-mail address: vrugt@lanl.gov (J.A. Vrugt).

0098-3004/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cageo.2005.10.015

be estimated directly from knowledge of the underlying system, most represent effective properties that cannot, in practice, be measured via direction observation. Therefore, it is common practice to estimate values for model parameters by calibrating the model against a historical record of input–output data. The successful application of these models depends critically on how well the model is calibrated.

Because of the time-consuming nature of manual trial-and-error model calibration, there has been a great deal of research into the development of automatic methods for parameter estimation (Levenberg, 1944; Marquardt, 1963; Nelder and Mead, 1965; Kirkpatrick et al., 1983; Glover, 1986; Goldberg, 1989; Duan et al., 1993; Bahren et al., 1997; Zitzler and Thiele, 1999; among many others). Automatic parameter estimation methods seek to take advantage of the speed and power of digital computers while being objective and relatively easy to implement. Over the years, many studies have demonstrated that population-based global-search approaches have desirable properties that allow them to overcome many of the difficulties related to the shape of the response surface (the objective function mapped out in the parameter space). These methods have therefore become standard for solving complex non-convex optimization problems. However, application of global optimization methods to high-dimensional parameter estimation problems requires the solution of a large number of deterministic model runs. The computational burden of these models often hampers the use of advanced global optimization algorithms for calibrating parameters in complex environmental models.

Fortunately, during the past decade there has been considerable progress in the development of distributed computer systems using the power of multiple processors to efficiently solve complex, high-dimensional computational problems (Byrd et al., 1993; Coleman et al., 1993a,b; Moré and Wu, 1995). Parallel computing offers the possibility of solving computationally challenging optimization problems in less time than is possible using ordinary serial computing (Abramson, 1991; Goldberg et al., 1995; Alba and Troya, 1999; Herrera et al., 1998; Alba et al., 2004; Eklund, 2004; de Toro Negro et al., 2004; amongst various others). Despite these prospects, parallel computing has not entered into widespread use in the field due to difficulties with implementation and barriers posed by technical jargon. This is unfortunate, as many optimization problems in earth science are "embarrassingly parallel" and thus are ideally suited for solution on distributed computer systems.

In this paper we describe a parallel computing implementation of the Shuffled Complex Evolution Metropolis (SCEM-UA) global optimization algorithm for computationally efficient stochastic estimation of parameters in environmental models. Our implementation uses the recently developed MPITB toolbox for GNU Octave (Eaton, 1998[1]; Fernández Baldomero, 2004[2]; Fernández Baldomero et al., 2004[3]), which is designed to take advantage of the computational power of a distributed computer system. The implementation of parallelization in the SCEM-UA algorithm is done in a user-friendly way, such that the software can be easily adapted without in-depth knowledge of parallel computing. The features and capability of the parallel SCEM-UA implementation are illustrated using a diverse set of modeling case studies of increasing complexity: (1) a synthetic 20-dimensional benchmark problem; (2) the calibration of the conceptual Sacramento Soil Moisture Accounting (SAC-SMA) model, and (3) the prediction of flight routes of migratory birds.

The remainder of this paper is organized as follows. Section 2 presents a short introduction on parameter estimation in environmental models. In Sections 3 and 4 we describe the SCEM-UA algorithm and its parallel implementation on a distributed computer system, briefly describe the MPITB toolbox for GNU Octave for parallelization, and discuss the MPI (Message Passing Interface, 1997) implementation used to distribute tasks between different computers. In Section 5, we illustrate the power and applicability of parallel SCEM-UA parameter optimization using three case studies of increasing complexity. There we focus on the relationship between the computational time needed for parameter estimation and number of nodes used. Finally, in Section 6 we summarize the results and conclusions.

[1]Eaton, J.W., 1998. Web: http://www.octave.org/, University of Wisconsin, Department of Chemical Engineering, Madison, WI 53719.

[2]Fernández Baldomero, J., 2004. LAM/MPI parallel computing under GNU Octave, http://atc.ugr.es/javier-bin/mpitb

[3]Fernández Baldomero, J., Anguita, M., Mota, S., Cañaz, Ortigosa, E., Rojas, F.J., 2004. MPI toolbox for Octave, VecPar'04, Valencia, Spain, http://atc.ugr.es/~javier/investigacion/papers/VecPar04.pdf

2. Parameter estimation

Consider an environmental system Φ for which a model f is to be calibrated. Assume that the mathematical structure of the model is essentially predetermined and fixed, and that realistic upper and lower bounds on each of the p model parameters can be specified a priori. Let $\tilde{Y} = \{\tilde{y}_1, \ldots, \tilde{y}_t\}$ denote the vector of measurement data available at time steps 1,…,t and let $Y(\theta) = \{y_1(\theta), \ldots, y_t(\theta)\}$ represent the corresponding vector of model output predictions using the model f with parameter values θ. The difference between the model-simulated output and measured data can be represented by the residual vector, E:

$$E(\theta) = G[Y(\theta)] - G[\tilde{Y}] = \{e_1(\theta), \ldots, e_t(\theta)\}, \tag{1}$$

where the function $G(\cdot)$ allows for various user-selected linear or non-linear transformations. The aim of model calibration is to determine a set of model parameters θ such that the measure E is in some sense forced to be as close to zero as possible. The formulation of a criterion that mathematically measures the "size" of $E(\theta)$ is typically based on assumptions regarding the distribution of the measurement errors presented in the data.

The classical approach to estimating the parameters in Eq. (1) is to ignore input data uncertainty and to assume that the predictive model f is a correct representation of the underlying physical data-generating system (Φ). In line with classical statistical estimation theory, the residuals in Eq. (1) are then assumed to be mutually independent (uncorrelated) and Gaussian-distributed with a constant variance. Under these circumstances, the traditional "best" parameter set in Eq. (1) can be found by minimizing the following additive simple least-squares (SLS) objective function with respect to θ:

$$F_{SLS}(\theta) = \sum_{i=1}^{t} e_i(\theta)^2. \tag{2}$$

For cases where the residuals in Eq. (1) are correlated a covariance structure of the residuals (or measurement noise) can be included in the definition of Eq. (2) so that the error terms become uncorrelated. Many algorithms have been developed to solve the non-linear SLS optimization problem stated in Eq. (2). These algorithms include local search methodologies, which seek to improve the objective function using an iterative search starting from a single arbitrary initial point in parameter space; and global search methods, in which multiple, concurrent searches are conducted from different starting points within parameter space.

3. The SCEM-UA algorithm

The SCEM-UA algorithm is a general-purpose, global optimization algorithm that provides an efficient estimate of the most likely parameter set and its underlying posterior probability distribution within a single optimization run (see Vrugt et al., 2003b). A condensed description of the method is given below and is illustrated in Fig. 1.

1. **Generate sample**: Sample s parameter combinations $\{\theta_1, \ldots, \theta_s\}$ randomly from the prior distribution and compute the posterior density of each of these points using a slightly modified implementation of Eq. (2) as presented in Box and Tiao (1973).
2. **Rank points**: Sort the s points in order of decreasing posterior density and store them in array $D[1{:}s,1{:}n+1]$, where n denotes the number of parameters.
3. **Initialize Markov Chains**: Initialize the starting locations of k sequences using the first k elements of D; $S^k = D[k, 1 : n+1]$.
4. **Partition into complexes**: Partition the s points of D into k complexes $\{C^1, \ldots, C^k\}$, each containing m points. The first complex contains every $k(j-1)+1$ point of D, the second complex every $k(j-1)+2$ of D, and so on, where $j = 1, \ldots, m$.
5. **Evolve each sequence/complex**: Evolve each sequence and complex using the Sequence Evolution Metropolis algorithm, described in detail in Vrugt et al. (2003b).
6. **Shuffle complexes**: Unpack all complexes C back into D, and rank the points in order of decreasing posterior density.
7. **Check convergence**: If convergence criteria are satisfied, stop; otherwise return to step 4.

The algorithm is an approximate Markov Chain Monte Carlo (MCMC) sampler, which generates k sequences of parameter sets $\{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(N)}\}$ that converges to the stationary posterior distribution for a large enough number of simulations N.

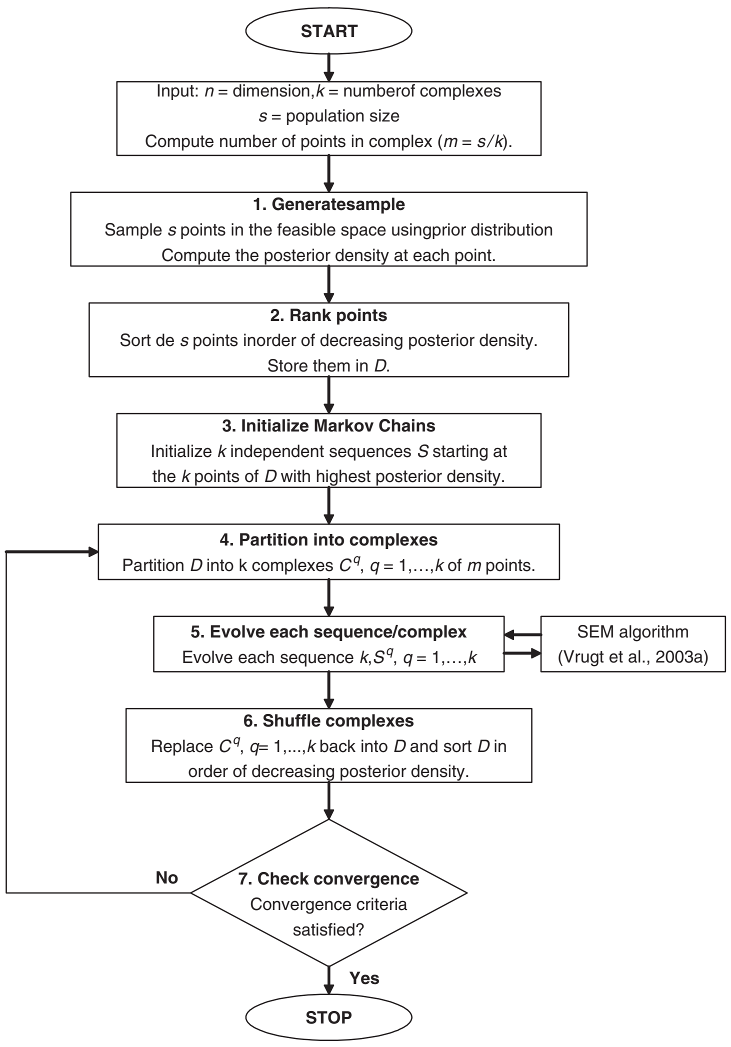

START
Input: n = dimension, k = numberof complexes
s = population size
Compute number of points in complex ($m = s/k$).
1. Generatesample
Sample s points in the feasible space usingprior distribution
Compute the posterior density at each point.
2. Rank points
Sort de s points inorder of decreasing posterior density.
Store them in D.
3. Initialize Markov Chains
Initialize k independent sequences S starting at
the k points of D with highest posterior density.
4. Partition into complexes
Partition D into k complexes C^q, $q = 1,\ldots,k$ of m points.
5. Evolve each sequence/complex
Evolve each sequence k, S^q, $q = 1,\ldots,k$
SEM algorithm
(Vrugt et al., 2003a)
6. Shuffle complexes
Replace C^q, $q= 1,\ldots,k$ back into D and sort D in
order of decreasing posterior density.
No
7. Check convergence
Convergence criteria
satisfied?
Yes
STOP

Fig. 1. Flowchart of a typical sequential implementation of a parameter optimization algorithm: SCEM-UA algorithm.

The SCEM-UA algorithm is related to the successful SCE-UA global optimization method (Duan et al., 1993), but uses the Metropolis–Hastings (MH) search strategy (Metropolis et al., 1953; Hastings, 1970) instead of the Downhill Simplex method for population evolution. This approach enables the SCEM-UA algorithm to simultaneously infer the most likely parameter set and its underlying posterior probability distribution within a single optimization run. Although the former is not inferred directly, it can be inferred from the sample set of points generated with the algorithm. A detailed description and explanation of the method appears in Vrugt et al. (2003b), and so will not be repeated here.

3.1. *Sequential implementation*

The traditional implementation and application of many local and global optimization methods involves sequential execution of the algorithm using the computational power of a single Central Processing Unit (CPU). For example, the SCEM-UA algorithm described in the previous section is set up such that the various steps and associated model evaluations are executed sequentially, one after another. Such an implementation works acceptably well for relatively simple optimization problems, and those optimization problems with models that do not require much computational time to execute. However, for high-dimensional optimization problems involving complex spatially distributed models, such as are frequently used in the field of earth science, this sequential implementation needs to be revisited (Abramson, 1991; Goldberg et al., 1995; Alba and Troya, 1999; Herrera et al., 1998; Vrugt et al., 2001; Alba et al., 2004; Eklund, 2004; de Toro Negro et al., 2004; Vrugt et al., 2004; amongst various others).

Most computational time required for calibrating parameters in complex environmental models is spent running the model code and generating the desired output. Thus, there should be large computational efficiency gains from parallelizing the algorithm so that independent model simulations are run on different nodes in a distributed computer system.

3.2. Parallel implementation

With modifications to the original sequential SCEM-UA code, the generation of the initial sample (Step 1) and evolution of sequences/complexes (Step 5) can be solved on different nodes. In this section we describe a parallel implementation of the SCEM-UA algorithm for computationally efficient use on multiprocessor distributed computer systems.

A condensed description of our parallel implementation of the SCEM-UA method is given below and is illustrated in Fig. 2. A distinction is made between MASTER and SLAVE processors.

1. **MASTER—Generate sample**: Sample s parameter combinations $\{\theta_1, \ldots, \theta_s\}$ randomly from the prior distribution.
   SLAVE—Compute posterior density: Each slave processor is assigned to evaluate a subset of s.
2. **MASTER—Rank points**: Sort the s points in order of decreasing posterior density and store them in array $D[1:s, 1:n+1]$, where n denotes the number of parameters.
3. **MASTER—Initialize Markov Chains**: Initialize the starting locations of k sequences using the first k elements of D; $S^k = D[k, 1:n+1]$.
4. **MASTER—Partition into complexes**: Partition the s points of D into k complexes $\{C^1, \ldots, C^k\}$, each containing m points. The first complex contains every $k(j-1)+1$ point of D, the second complex every $k(j-1)+2$ of D, and so on, where $j = 1, \ldots, m$.
5. **MASTER—Evolve complexes**: Generate and distribute new candidate points to slaves.
   SLAVE—Compute posterior density: Each slave processor is assigned to evaluate a different set of candidate points.
6. **MASTER—Shuffle complexes**: Unpack all complexes C back into D, and rank the points in order of decreasing posterior density.
7. **MASTER—Check convergence**: If convergence criteria are satisfied, stop; otherwise return to step 4.

This parallel implementation of the SCEM-UA algorithm is different from its sequential counterpart in two important ways. First, the evaluation of the fitness function for the individuals in the population is distributed over the slave processors, thereby avoiding excessively long execution times on a single processor. Second, each slave computer is set up to evolve a different sequence and complex, as this step does not require information exchange and communication between different nodes. In fact, this type of parallelism can be applied to almost any optimization algorithm which involves sequential evaluation of fitness functions (Cantú Paz and Goldberg, 1998). Both modifications are in line with our arguments set forth in the previous section, and significantly reduce the computational burden needed to solve complex high-dimensional optimization problems.

4. Parallel computing

4.1. The MPITB toolbox for Octave

Distributed computers have the potential to provide an enormous computational resource for solving complex environmental problems, and there is active research in this area to take better advantage of parallel computing resources. For example, in hydrology applications, parallel computing is being exploited to improve computational efficiency of individual, large-scale groundwater flow (Wu et al., 2002) and reactive transport (Hammond et al., 2005) models. Efforts to couple hydrologic models consisting of a network of individual submodels (groundwater, surface water, and atmospheric models) also are being designed in a way that submodels can be partitioned to different processors (Winter et al., 2004). Finally, parallel versions of model inversion and sensitivity analysis software such as PEST (Doherty, 2004) have been developed.

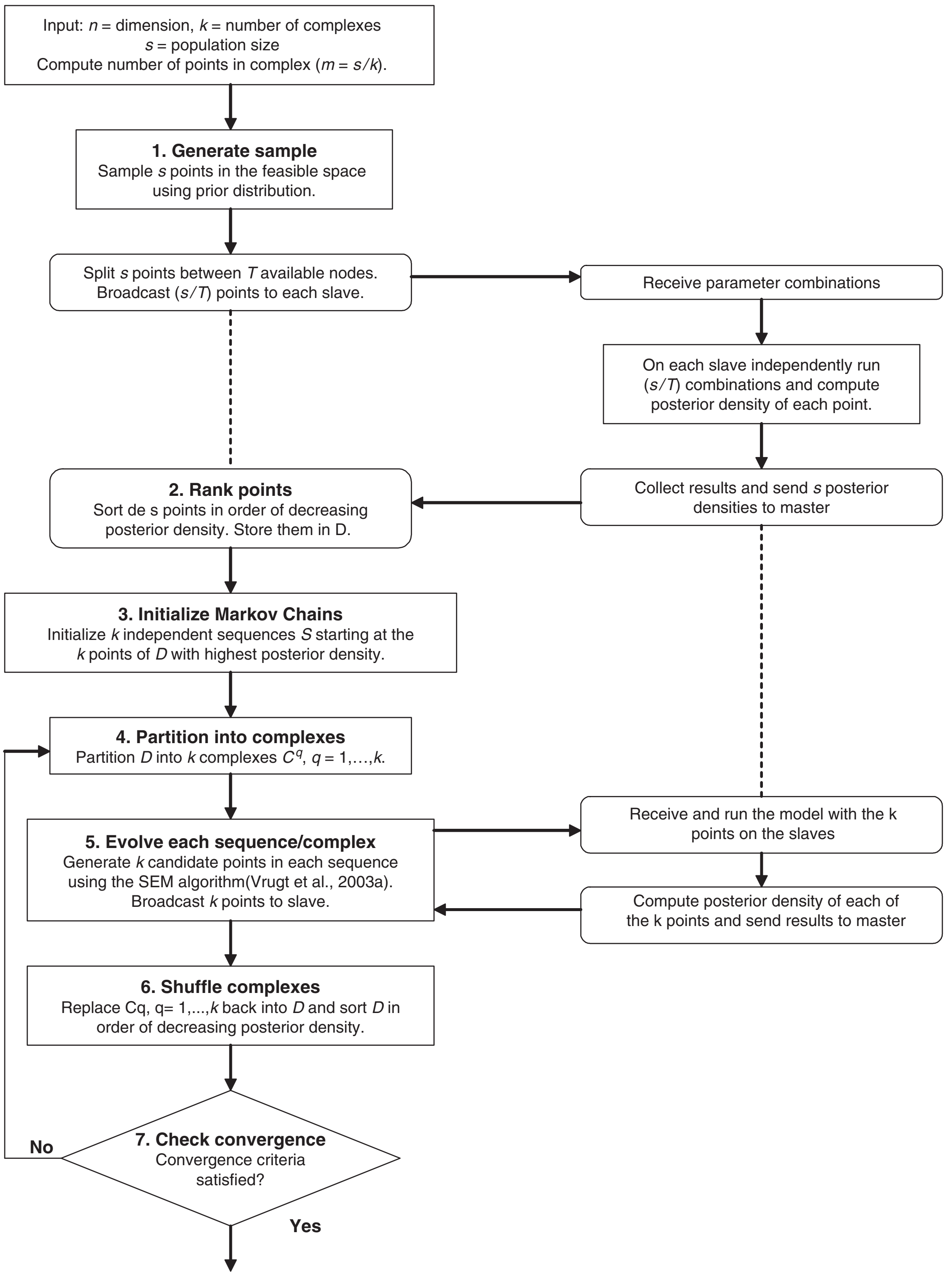

Input: n = dimension, k = number of complexes
s = population size
Compute number of points in complex (m = s/k).
1. Generate sample
Sample s points in the feasible space using prior distribution.
Split s points between T available nodes. Broadcast (s/T) points to each slave.
Receive parameter combinations
On each slave independently run (s/T) combinations and compute posterior density of each point.
Collect results and send s posterior densities to master
2. Rank points
Sort de s points in order of decreasing posterior density. Store them in D.
3. Initialize Markov Chains
Initialize k independent sequences S starting at the k points of D with highest posterior density.
4. Partition into complexes
Partition D into k complexes C^q, q = 1,…,k.
5. Evolve each sequence/complex
Generate k candidate points in each sequence using the SEM algorithm(Vrugt et al., 2003a). Broadcast k points to slave.
Receive and run the model with the k points on the slaves
Compute posterior density of each of the k points and send results to master
6. Shuffle complexes
Replace Cq, q= 1,...,k back into D and sort D in order of decreasing posterior density.
7. Check convergence
Convergence criteria satisfied?
No
Yes
STOP

Fig. 2. Flowchart of a parallel implementation of SCEM-UA algorithm. Master computer performs various algorithmic steps in SCEM-UA algorithm (on left-hand side of flowsheet), while slave computers run simulation model (on right-hand side of flowsheet).

Despite the potential, to date there is limited use of parallel computing resources in the field of earth science, probably because parallel computing requires familiarity with technical jargon, as well as major restructuring of existing source codes. Recognizing these problems, Fernandez Baldomero et al. (2004) recently developed a user-friendly MPI toolbox for Octave (Eaton, 1998), called MPITB that provides a hands-on, command-line environment for performing parallel computations. The excellent functionality and completeness of MPITB, along with the ease of use and free availability of the source code, make this toolbox an excellent choice for developers of parallel computing applications. Indeed, MPITB allows Octave users to build their own MPI-based parallel applications, by simply installing the required software and adding the required MPI calls to their source Octave code, without patching their Octave installation.

The Message Passing Interface (MPI: Message Passing Interface Forum, 1997) is a specification of a mechanism for passing instructions between different computational resources, which may run on different nodes of a cluster of computers. The MPITB for the GNU Octave program environment developed by Fernandez Baldomero et al. (2004) makes use of the LAM/MPI implementation. This LAM/MPI is a high-quality open-source implementation of the MPI specification, including all of MPI-1.2, and much of MPI-2 (LAM team, 2004[4]). GNU Octave (Eaton, 1998) is a freely available high-level language with a syntax that is mostly compatible with MATLAB™, and is primarily intended for numerical computations. It contains a large number of built-in functions, and additional extensions are available from the Octave-forge package available at http://octave.sourceforge.net/. Both Octave and octave-forge extensions run under the Windows, Linux, and Mac OS X operating systems.

4.2. Implementation of parallel SCEM-UA using MPITB

In this section we describe how the parallel SCEM-UA algorithm is implemented in Octave using MPITB. MPITB follows the LAM/MPI syntax, so function names, arguments, and returns are all the same if one were directly using the LAM/MPI C or FORTRAN libraries. Since extensive documentation is available for each of the MPI functions (http://www.lam-mpi.org/tutorials/) we will not go into detail of how to use MPI functions. Instead, we focus on the unique elements of our implementation, following the parallel outline of the SCEM-UA algorithm in the previous section.

Fig. 3 presents a flowchart of the parallel SCEM-UA implementation in Octave using pseudo-code. Figs. 3a and b describe the algorithmic steps undertaken by the Master computer, including the communication with the slaves, and Fig. 3c presents that part of the Octave code being run on the slave computers. The pseudo-code describes in detail each line of the Octave implementation of the parallel SCEM-UA code. Function names are indicated with capital letters, whereas lines that start with the symbols '##' represent comments. In short, the optimization problem is distributed over a prespecified number of computational nodes on which an independent search of the parameter space is conducted. After a sufficient number of evolutionary steps, information between processors is shuffled and new complexes are generated. This iterative process continues until convergence has been achieved.

The most important MPI calls that are used to facilitate communication between the master and slave computers are: (1) *MPI_Send*—to send a package with parameter combinations (master) or model simulation outputs (slave), (2) *MPI_Prob*—to check whether there are any incoming messages, (3) *MPI_Get_elements*—to track the number of basic elements in the package, and (4) *MPI_Recv*—to retrieve the information in the package. A detailed description of each of these functions appears in tutorials from the LAM team (2004: http://www.lam-mpi.org/tutorials/) and so will not be repeated here.

5. Case studies

We demonstrate the power and applicability of the parallel SCEM-UA algorithm for three case studies with increasing complexity. The first case study is a synthetic, highly non-linear benchmarking problem designed to illustrate the ability of our parallel implementation to infer the known posterior target distribution. This problem contains 20 parameters. The second case study considers the calibration of the SAC-SMA conceptual watershed model involving the estimation of 14 parameters.

[4]LAM team 2004. LAM/MPI parallel computing, http://lam-mpi.org/

```
                         PROGRAM RUNNING ON MASTER COMPUTER
IN:
                  s                       population size
                  n                       number of model parameters subject to optimization
                  minn                    vector with minimum value of each parameter
                  maxn                    vector with maximum value of each parameter
                  k                       number of complexes/sequences used in search
                  Measurement             calibration data time series
                  Nmeas                   number of measurements in calibration time series
                  nodes                   number of computational nodes
OUT:

                  ParSet                  (Posterior parameter distribution)
1. Generate sample
   ## Generate initial sample using Latin Hypercube ##
   x = LATIN(n,s,minn,maxn);
   ## Distribute individuals of population over different nodes (Fig. 3B) ##
   [SimAllData] = DISTTASK(x,s,n,nodes,Nmeas);

              SLAVE – Run model on slaves and return output for each point (Fig. 3C)

   ## Compute the posterior density of each point from simulation results ##
   [pset] = COMPUTEDENSITY(SimAllData, Measurement);
   ## Collect results in array ParSet ##
   ParSet = [x, pset(1:end, 1)];
2. Rank points
  ## Sort in order of decreasing posterior density and store results in array D ##
  D = -SORTROWS(-pset, 1);
3. Initialize Markov Chains
   ## Initialize the starting locations of the k sequences ##
   Seq = INITSEQUENCES(D, x, k); notconverged = 1;

while (notconverged == 1)
      4. Partition into complexes
        ## Partition D into k complexes ##
        C = PARTCOMPLEXES(D, x, k);
      5. Evolve complexes
              ------ Sequence Evolution Metropolis algorithm (see Vrugt et al., 2003a) ------
              ## First generate new candidate point for each complex and sequence ##
              [newpar, Ratio] = OFFMETRO(Seq, C, minn, maxn, Measurement);
              ## Distribute new points over different nodes ##
              [SimAllData] = DISTTASK(newpar, s, n, nodes, Nmeas);

                        SLAVE –Run model on slaves and return output for each point (Fig. 3C)

              ## Compute the posterior density of the new points ##
              [pset] = COMPUTEDENSITY(SimAllData, Measurement);
              ## Apply the Metropolis accept/reject rule ##
              [C, newgen] = METROPOLIS(pset, newpar, C, Seq, Ratio, Measurement);
              ## Now update sequences and add combinations to ParSet ##
              [Seq, ParSet] = COLLECTS(Sequences, newgen, ParSet);
      6. Shuffle complexes
        ## Reshuffle the points ##
        [D, x] = RESHUFFLE(C, k);
      7. Check convergence
        ## Computer Gelman and Rubin convergence statistics ##
        [notconverged] = GELMAN(Sequences);
end
```

(A)

Fig. 3. Pseudo-code describing our parallel implementation of SCEM-UA algorithm in octave: (A, B) algorithmic steps undertaken by master computer, and (C) octave program running on slave computers.

```
                         PROGRAM RUNNING ON MASTER COMPUTER

function [SimAllData] = DISTTASK(x,s,n,nodes,N);

## Distribute individuals population over different nodes and return model output for each point ##

IN:
                 n                    number of model parameters subject to optimization
                 x                    population of points
                 s                    population size
                 nodes                number of computational nodes
                 Nmeas                number of measurements in calibration time series
                 tag                  Unique tag id for message (just a constant number)
GLOBAL:

OUT:             SimAllData           (model output for each point)


## Distribute individuals over different slave nodes ##
for slave = 1:nodes-1,
    ## Divide the population so each slave is assigned a different subset ##
    [lhs rhs] = DIVVY(s, nodes-1, slave);
    ## Generate package for each slave ##
    package = RESHAPE(x(lhs:rhs, :),1, (rhs-lhs +1)*n);
    ## Send the package tothe slave with MPI ##
    MPI_SEND (package, slave, tag,-1);

    end

## Collect results from different nodes##
for slave = 1:nodes-1,
    ## Compute the indicesagain, so that output slaves corresponds to the right individual ##
    [lhs rhs] = DIVVY(s,nodes-1, slave);
    ## Checking for incoming message using MPI ##
    [minfomstat] = MPI_PROBE(slave, tag, -1);
    ## Compute and return the number of basic elements ofmessage using MPI ##
    [minfodbls] = MPI_GET_ELEMENTS(mstat,[]);
    ## Pre-generate a vector Nslavein which results fromslaves will be collected ##
    Nslave = zeros(1, dbls);
    ## Receive results fromslaves using MPI and store result in Nslave##
    [minfomstat] = MPI_RECV(Nslave, slave, tag,-1);
    ## Reshape Nslaveto generate desired shape for use in the algorithm ##
    SimAllData (lhs:rhs,:) =RESHAPE(Nslave, rhs-lhs +1, Nmeas);

  end
```
(B)

Fig. 3. (*Continued*)

Finally, the third case study focuses on the prediction of migratory trajectories of passerine birds over continental Europe using a two-dimensional, spatially explicit model having eight calibration parameters. In case studies 2 and 3, we examine algorithm efficiency, particularly the relationship between computational time and number of parallel processors used. The calculations reported in this paper were performed on the state-of-the-art 272 node (2 Intel® Xeon 3.4 GHz) Lisa cluster of the SARA Computing & Network Services in Amsterdam, The Netherlands.

5.1. *Case study 1: a 20-dimensional banana-shaped posterior target distribution*

This case study is a benchmarking problem for testing our parallel SCEM-UA algorithm. The 20-dimensional, non-linear, banana-shaped distribution is constructed from the standard multivariate

```
                         PROGRAM RUNNING ON SLAVE COMPUTERS

IN:
             n                        number of model parameters subject to optimization
             Nmeas                    number of measurements in calibration time series
             nodes                    number of computational nodes
             ModelName                name of the model to run

GLOBAL:      tag                      Unique tag id for message (just a constant number)

OUT:
             out                      (model output for each point)

while (1 == 1)
   ## Checking for incoming message using MPI ##
   [minfo mstat] = MPI_PROBE(-1,-1,-1);
   ## Checking for incoming message using MPI ##
   [minfo dbls] = MPI_GET_ELEMENTS(mstat,[]);
   ## Pre-generate a vector Nmaster in which parameter combinations from master are collected ##
   Nmaster = zeros(1,dbls);
   ## Receive parameter combinations from master using MPI and store result in Nmaster ##
   [minfo mstat] = MPI_RECV(Nmaster,-1,tag,-1);
   ## Reshape Nmaster to retrieve original array with parameter values ##
   x = RESHAPE(Nmaster,dbls/n,n);
   ## Compute the number of individuals (parameter combinations)
   [nrsets] = length(x);
   ## Compute model output for each of the individuals and store results in array out
   for i=1:nrsets
      evalstr = ['SimData = ',ModelName,'(x(i,1:n));'];
      EVAL(evalstr);
      out(i,:) = SimData;
   end
   ## Generate the package to be sent to the master computer
   package = RESHAPE(out,1,nrsets*Nmeas);
   ## Send the package to the master with MPI ##
   MPI_SEND(package,0,tag,-1);
   ## Clear out
   clear out;
end
```

(C)

Fig. 3. (*Continued*)

Gaussian distribution as follows (see Haario et al., 1999). Let f be the density of the multivariate normal distribution, $N(0, \Sigma)$ with covariance matrix given by $\Sigma = \mathrm{diag}(100, 1, \ldots 1)$. The twisted Gaussian density function with non-linearity parameter b is given by

$$f_b = f \circ \phi_b \tag{3a}$$

where the function ϕ_b is

$$\phi_b(\theta) = (\theta_1, \theta_2 + b\theta_1^2 - 100b, \theta_3, \ldots, \theta_p). \tag{3b}$$

Our test used $b = 0.1$ to generate a strongly twisted banana-shaped posterior target distribution. Given a combination of values for θ, Eq. (3) directly computes the corresponding posterior density. So, for this specific example, one does not need to evaluate an objective function, such as done in Eq. (2). The population size s in the SCEM-UA algorithm was set to 2000, and the number of parallel sequences k and slave processors *nodes* were set to 50.

Fig. 4 presents a scatterplot of the (θ_1, θ_2) sampled SCEM-UA points that were generated after convergence of the parallel sequences had been achieved to a stationary posterior distribution. The solid black line defines the theoretical posterior distribution. Notice that our parallel implementation of the SCEM-UA algorithm has sampled points that are fully consistent with the prior-

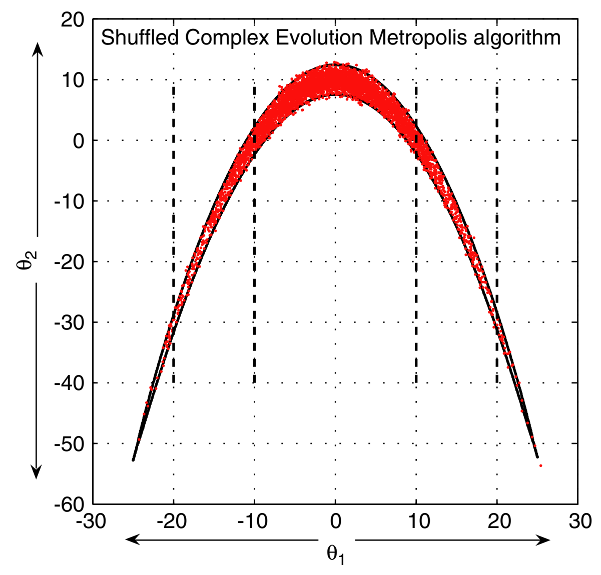


Fig. 4. A scatterplot of (θ_1, θ_2) sampled parameters using SCEM-UA algorithm. Dark line represents theoretical target distribution, whereas dashed lines denote one-dimensional 68.3% and 95% confidence region of parameters.

defined target distribution. This suggests that the sampler provides a correct estimate of the underlying posterior distribution, and that our parallel implementation has been successful.

5.2. Case study 2: the Sacramento soil moisture accounting (SAC-SMA) model

We investigate the benefits of using parallel computing by applying the method to the calibration of the SAC-SMA conceptual watershed model (see Fig. 5) using data from the Leaf River watershed (1950 km^2) near Collins, Mississippi. This model is used extensively by the National Weather Service for streamflow forecasting and flood warning throughout the United States. The model has 13 user-specified (and 3 fixed) parameters (see Table 1) and an evapotranspiration demand curve (or adjustment curve). Inputs to the model include mean areal precipitation (MAP) and potential evapotranspiration (PET), while the outputs are estimated evapotranspiration and channel inflow. A simple Nash Cascade (NC) of three linear reservoirs (with one tunable parameter) is used to route the upper zone (quick response) channel inflow (Z_1), while the lower zone (slow flow recession components Z_2 and Z_3) are passed directly to the gauging point. Therefore, our calibration problem has 14 time-invariant parameters. The feasible parameter space was defined by fixing the upper and lower bounds at their "level zero" estimates presented in Boyle et al. (2000).

The hydrologic data, obtained from the National Weather Service Hydrology Laboratory (HL), consists of 6 h MAP (mm/day), daily streamflow (m^3/s) and PET (mm/day). Because the SAC-SMA model and Leaf River data have been discussed extensively in previous work (Burnash et al., 1973; Brazil and Hudlow, 1981; Sorooshian et al., 1993; Yapo et al., 1996; Boyle et al., 2000; Vrugt et al., 2003a,b), details of the modeling and data set will not be reported here. In this study we use 11 years (WY 1953–1962) of data for calibration and uncertainty assessment of the SAC-SMA model parameters.

To estimate the model parameters we adopt a classical Bayesian approach using the following posterior density criterion, $p(\theta|Y)$ (Box and Tiao, 1973):

$$p(\theta|Y) = F_{SLS}(\theta)^{-(1/2)t}, \tag{4}$$

where t denotes the total number of streamflow observations, and F_{SLS} is computed according to Eq. (2). Based on recommendations in previous work (Vrugt et al., 2003b), the stationary posterior distribution corresponding to the density criterion defined in Eq. (4) was estimated using a population size of $s = 1000$ points and $k = 25$ parallel sequences.

Table 1 summarizes the SCEM-UA derived posterior uncertainty ranges of the SAC-SMA parameters. The listed ranges pertain to a confidence level of 99.5%. In relation to the Level Zero or prior parameter ranges, most of the SAC-SMA parameters are well identified by calibration to streamflow data. In particular, the capacity parameters UZTWM, UZFWM, LZTWM, LZFSM and LZFPM are well determined, while parameters ZPERC and REXP (that control percolation), ADIMP (additional impervious area), and the rate parameters LZSK and LZPK are less well determined. These results suggest that most uncertainty in the model structure is associated with percolation from the upper zone and depletion from the lower zone.

We now demonstrate the computational benefits of using parallel parameter estimation. Fig. 6 is a plot of computational time versus the number of computational nodes used in the Lisa cluster. The solid curve represents the average result of 10 trials,

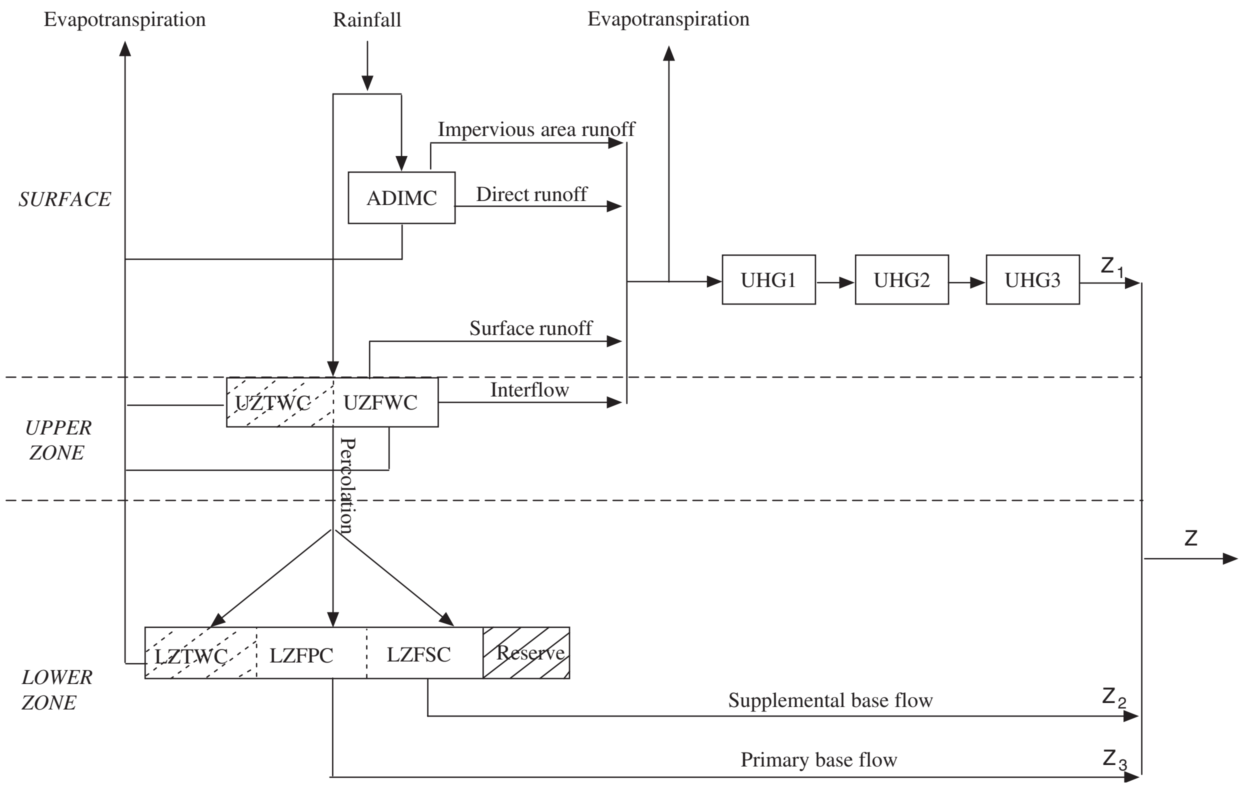


Fig. 5. Schematic representation of Sacramento Soil Moisture Accounting (SAC-SMA) model as implemented in this study. Rectangles refer to various model states, while arrows indicate fluxes between compartments. Z_1, Z_2, and Z_3 refer to three main channel components which sum together to become streamflow Z at watershed outlet. Crossed boxes represent tension water storages.

while the box plots denote the spread among the replicates. The dashed black line denotes a theoretical 1:1 speed up. The use of one computational node represents a classical sequential SAC-SMA model calibration, which for this case required approximately 25 min on a Pentium 3.4 GHz computer. With the use of more computational nodes, the time needed for SAC-SMA model calibration steadily decreases to about 3.5 min (a 90% reduction in computational time) when the number of slave processors is identical to the number of parallel sequences/complexes used to explore parameter space. Note that the reported computational time closely tracks the 1:1 speed up curve, suggesting that the communication time between master and slave computers is small compared to the time needed to run the SAC-SMA model. Finally, we note that the number of processors that can be efficiently used with the parallel algorithm, as currently implemented, is limited to no more than the number of parallel sequences/complexes chosen.

5.3. *Case study 3: bird migration modeling*

The third and final case study reported in this paper illustrates the application of parallel computing to help understand and examine spatial and temporal variations in optimal migration direction for migratory birds. For this purpose we developed a two-dimensional, spatially explicit, individual-based simulation model. The model combines the strengths of flight mechanical theory (Pennycuick, 1998) and two-dimensional, spatially explicit modeling (Erni et al., 2002) to simulate the time evolution of the spatial location and fat amount of an individual bird. For an extensive description of the model and its most important parameters see Vrugt and Bouten (2005), Vrugt et al. (2005).

The spatial domain of the model, which covers continental Europe and northern Africa, is discretized into a two-dimensional rectangular grid of cells ranging from $-20 \times 40^\circ$ longitude to $10 \times 70^\circ$ latitude using a grid resolution of 0.5° in both directions. Each spatial cell was assigned a different

Table 1
Parameter and state variables in Sacramento Soil Moisture Accounting (SAC-SMA) model

States	Description	Initial ranges	Posterior ranges
Capacity thresholds			
UZTWM	Upper zone tension water maximum storage (mm)	1.0–150.0	3.3–9.9
UZFWM	Upper zone free water maximum storage (mm)	1.0–150.0	28.3–36.3
LZTWM	Lower zone tension water maximum storage (mm)	1.0–500.0	249.9–316.3
LZFPM	Lower zone free water primary maximum storage (mm)	1.0–1000.0	67.4–93.1
LZFSM	Lower zone free water supplemental maximum storage (mm)	1.0–1000.0	1.1–6.9
ADIMP	Additional impervious area	0.0–0.40	0.03–0.23
Recession parameters			
UZK	Upper zone free water lateral depletion rate (d^{-1})	0.1–0.5	0.45–0.50
LZPK	Lower zone primary free water depletion rate (d^{-1})	0.0001–0.025	0.011–0.018
LZSK	Lower zone supplemental free water depletion rate (d^{-1})	0.01–0.25	0.12–0.25
Percolation and other			
ZPERC	Maximum percolation rate	1.0–250.0	164.7–250.0
REXP	Exponent of percolation equation	1.0–5.0	1.0–1.5
PCTIM	Impervious fraction of watershed area	0.0–0.1	0.0–0.003
PFREE	Fraction percolating from upper to lower zone free water storage	0.0–0.6	0.00–0.021
Not optimized			
RIVA	Riparian vegetation area	0.0	
SIDE	Ratio of deep recharge to channel base flow	0.0	
RSERV	Fraction of lower zone free water not transferable to tension water	0.3	
UZTWC	Upper zone tension water storage content (mm)		
UZFWC	Upper zone free water storage content (mm)		
LZTWC	Lower zone tension water storage content (mm)		
LZFPC	Lower zone free primary water storage content (mm)		
LZFSC	Lower zone free secondary water storage content (mm)		
ADIMC	Additional impervious area content (mm)		

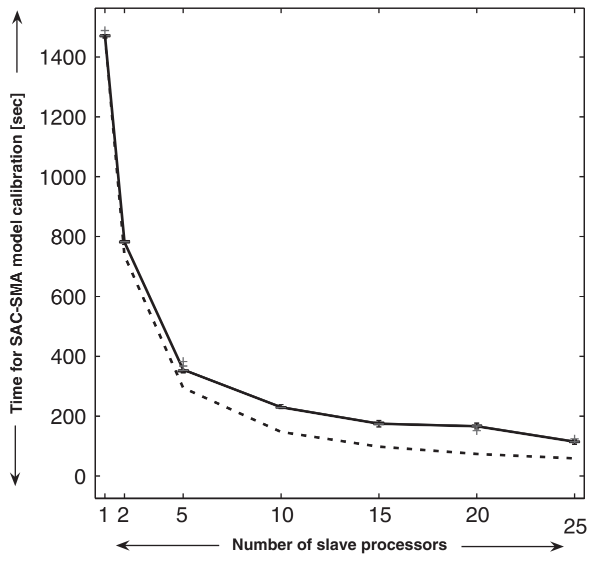


Fig. 6. Computational time needed for a complete SAC-SMA model calibration against number of Pentium IV 3.4 GHz processors used. Curve represents average of 10 independent trials. Box plots indicate spread among replicates, whereas dashed black line denotes a linear 1:1 speed up.

Fuel Deposition Rate (FDR: Erni et al., 2002). The meteorological conditions (wind direction and speed) at each spatial cell are updated hourly using linear interpolation between two consecutive 6-h predicted wind maps from reanalysis runs of the NCEP model of the National Oceanic and Atmospheric Administration (NOAA). The orientation and navigation direction of the bird is computed during the initialization of the model from the location of the breeding ground (60.0°N, 10.0°W) and endogenous direction (D_{endog}) using an arrival location at 15° north latitude. For autumn migration, the computer simulation is conducted at equal time intervals of 1 h, starting on August 1, 2004. This time-marching computation continues until the bird either covers the required distance to the wintering grounds, or runs out of consumable fat reserves, or when the simulation time exceeds 100 days. In this illustrative study we focus on the willow warbler, a small passerine bird.

Table 2 lists the most important parameters in the bird migration model. A distinction is made between default values and parameters subject to optimization. The uncertainty ranges of the calibration parameters correspond to those of the willow warbler. The total set of parameters in Table 2 defines how the willow warbler reacts to its fat reserve and spatially and temporally varying environmental conditions. Each parameter combination therefore results in a different simulated flight route trajectory.

To implement and test our parallel SCEM-UA algorithm, we must specify a life-history objective that the willow warbler is trying to satisfy, after which the algorithm can be used to estimate the parameters in Table 2. Most optimality models that have been developed in the avian migration literature consider flight time to be the main objective. In light of these considerations, we seek to identify all parameter combinations that result in a migration time close to the observed migration time of the willow warbler. Extensive ringing recoveries (Hedenström and Pettersson, 1987) have established this to be approximately 60 days. To conduct the parameter search with the SCEM-UA algorithm, we used a population size of 500 points, in combination with 10 parallel sequences. The results of our analysis are presented in Figs. 7 and 8 and discussed.

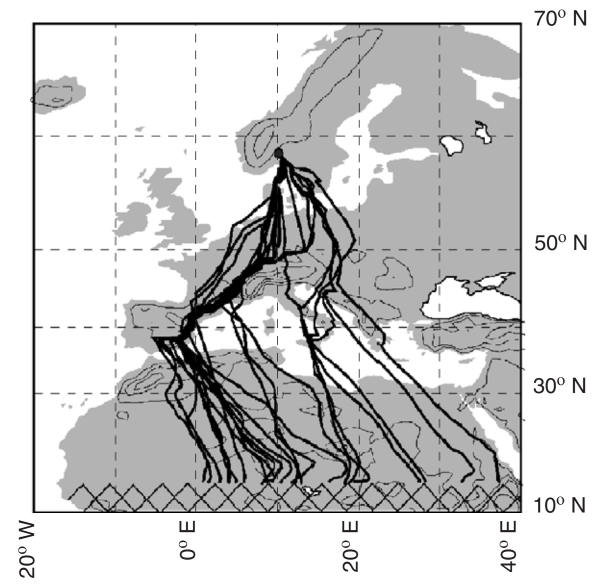
Fig. 7 depicts the model-predicted migratory pathways corresponding to a randomly chosen sample set of 25 SCEM-UA derived parameter combinations of the posterior distribution. The results presented illustrate that the willow warblers have significant flexibility in choosing an "optimal" migration strategy, as evidenced by the variability in simulated flight route trajectories. Nevertheless, this optimization suggests a co-existence of two main migratory pathways over continental Europe. This result corresponds very well with radar tracking data, ring recovery studies and other field investigations (Hedenström and Pettersson, 1987; Alerstam, 1996) supporting the validity of our model and optimization hypothesis.

To examine the computational efficiencies of the parallel algorithm for this case study, Fig. 8 presents a plot of the computational time needed to derive the posterior parameter distribution versus the number of slave processors used. The curve represents the average result of 10 independent trials. Note that parallel parameter estimation results in a considerable time savings when compared with a traditional sequential optimization run. Again, a reduction in computational time of about 90% is feasible when sufficient slave computers are used to evolve the parallel sequences. Also note that the computational requirements for calibration of this spatially explicit model are much higher than for the lumped conceptual SAC-SMA watershed model.

Table 2
Most important parameters in bird migration model: distinction is made between default values and those subject to optimization

Parameter	Description	Unit	Value/range
Default values[a]			
m_{musc}	Flight muscle mass	(g)	1.19
m_{frame}	Airframe mass	(g)	5.81
b	Wing span	(m)	0.193
S	Wing area	(m^2)	0.0070
m_{maxfat}	Maximum fat mass	(g)	6.30
Calibration parameters[a]			
$Init_{fat}$	Fat mass at start autumn migration	(g)	0–6.30
m_{minfat}	Minimum fat mass during flight	(g)	0–4.00
$m_{crossfat}$	Minimum fat mass to cross barrier	(g)	0–6.30
D_{endog}	Endogenous direction autumn	(deg)	130–230
P_{wind}	Wind compensation factor	(%)	0–100
V_{min}	Minimum net speed to take-off	($m\,s^{-1}$)	0–10
N_{fly}	Number of consecutive fly days	(d)	1–10
N_{rest}	Number of consecutive rest days	(d)	1–10

[a]Values and ranges are based on experiments with willow warblers reported in literature. See Vrugt et al. (2005) and Vrugt and Bouten (2005) for details.



⨂ Wintering grounds

● Breeding ground

Fig. 7. Model predicted migratory pathways corresponding to SCEM-UA derived posterior parameter distribution. Starting location is with a dot. Map is a Mercator projection.

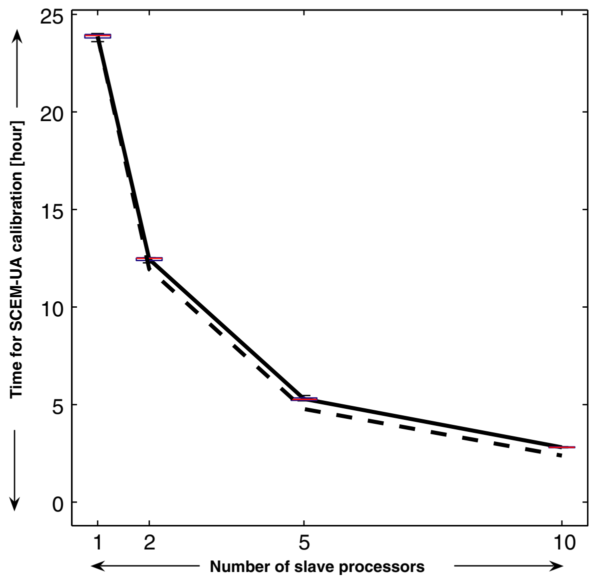


Fig. 8. Computational time needed for parameter estimation in spatially explicit bird migration model against number of Pentium IV 3.4 GHz processors used. Curve represents average of 10 independent trials. Box plots indicate spread among replicates, whereas dashed black line denotes a linear 1:1 speed up.

6. Summary and conclusions

This paper presents a parallel implementation of the SCEM-UA algorithm to facilitate a computationally efficient solution of complex optimization problems on distributed computer systems. The method implements the recently developed MPITB toolbox in GNU Octave. The algorithm operates by evolving each SCEM-UA generated parallel sequence on a different slave computer. The communication between the master and slave computer is conducted using the Message Passing Interface (MPI). Only minor modifications were needed to the original sequential SCEM-UA source code to facilitate implementation on a distributed computer system.

The power and applicability of the parallel SCEM-UA algorithm was demonstrated for three case studies of increasing complexity. The first study considered a classic mathematical highly non-linear 20-dimensional benchmark problem, and served to demonstrate that our parallel implementation does indeed successfully infer the known posterior target distribution. The second and third cases explored the use of parallel SCEM-UA for calibration of the Sacramento Soil Moisture Accounting (SAC-SMA) conceptual watershed model, and the prediction of flight route trajectories of migrating birds. Both studies clearly demonstrated that parallel parameter estimation results in a considerable time savings when compared with traditional sequential optimization runs. In fact, the reported speed up closely follows the theoretical 1:1 line, up to the point at which the number of processors equals the number of sequences being evolved.

In conclusion, the complexity and scientific diversity of the modeling problems that can be treated using the parallel SCEM-UA algorithm suggests that this work could open new avenues of research in environmental modeling. Our long-term goal is to improve the algorithms further, while making them available for use by others. The code for the parallel SCEM-UA algorithm is available from the first author (vrugt@lanl.gov). The MPITB toolbox and GNU octave are free software and can be downloaded from the internet.

Acknowledgments

The first author is supported by the LANL Director's Funded Postdoctoral program. We are grateful for using the Lisa cluster of the SARA

Computing & Network Services in Amsterdam, The Netherlands.

References

Abramson, D.A., 1991. Constructing school timetables using simulated annealing: sequential and parallel algorithms. Management Science 37 (1), 98–113.

Alba, E., Troya, J.M., 1999. A survey of parallel distributed genetic algorithms. Complexity 4, 31–52.

Alba, E., Luna, F., Nebro, A.J., Troya, J.M., 2004. Parallel heterogeneous genetic algorithms for continuous optimization. Parallel Computing 30, 699–719.

Alerstam, T., 1996. The geographical scale factor in orientation of migrating birds. Journal of Experimental Biology 199, 9–19.

Bahren, J., Protopopescu, V., Reister, D., 1997. TRUST: a deterministic algorithm for global optimization. Science 276, 1094–1097.

Box, G.E.P., Tiao, G.C., 1973. Bayesian Inference in Statistical Analyses. Addison-Wesley, Longman, Reading, MA 585pp.

Boyle, D.P., Gupta, H.V., Sorooshian, S., 2000. Towards improved calibration of hydrologic models: combining the strengths of manual and automatic methods. Water Resources Research 36 (12), 3663–3674.

Brazil, L.E., Hudlow, M.D., 1981. Calibration procedures used with the National Weather Service forecast system. In: Haimes, Y.Y., Kindler, J. (Eds.), Water and Related Land Resources. Pergamon, New York, pp. 457–466.

Burnash, R.J.C., Ferral, R.L., McGuire, R.A., 1973. A generalized streamflow simulation system: conceptual models for digital computers. National Weather Service, NOAA, and the State of California Department Of Water Resources Technical Report, Joint Federal-State River Forecast Center, Sacramento, CA, 68pp.

Byrd, R.H., Eskow, E., Schnabel, R.B., Smith, S.L., 1993. Parallel global optimization: numerical methods, dynamic scheduling methods, and application to nonlinear configuration. In: Ford, B., Fincham, A. (Eds.), Parallel Computation. Oxford University Press, New York, NY, pp. 187–207.

Cantú Paz, E., Goldberg, D.E., 1998. A survey of parallel genetic algorithms. Calculateurs Parallèles, Réseaux et Systèmes Répartis 10 (2), 141–171.

Coleman, T.F., Shalloway, D., Wu, Z., 1993a. A parallel build-up algorithm for global energy minimization of molecular clusters using effective energy simulated annealing. Journal of Global optimization 4, 171–185.

Coleman, T.F., Shalloway, D., Wu, Z., 1993b. Isotropic effective energy simulated annealing searches for low energy molecular cluster states. Computational Optimization and Applications 2, 145–170.

de Toro Negro, F., Ortega, J., Ros, E., Mota, S., Paechter, B., Martín, J.M., 2004. PSFGA: parallel processing and evolutionary computation for multiobjective optimization. Parallel Computing 30, 721–739.

Doherty, J., 2004. PEST Model-independent Parameter Estimation, User Manual: fifth ed. Watermark Numerical Computing Inc.

Duan, Q.Y., Gupta, V.K., Sorooshian, S., 1993. Shuffled Complex Evolution approach for effective and efficient global minimization. Journal of Optimization Theory and Applications 76 (3), 501–521.

Eklund, S.E., 2004. A massively parallel architecture for distributed genetic algorithms. Parallel Computing 30, 647–676.

Erni, B., Liechti, F., Bruderer, B., 2002. Stopover strategies in passerine bird migration: a simulation study. Journal of Theoretical Biology 219, 479–493.

Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research 5, 533–549.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine learning. Addison-Wesley, Reading, MA, pp. 1–403.

Goldberg, D.E., Kargupta, H., Horn, J., Cantú-Paz, E., 1995. Critical deme size for serial and parallel genetic algorithms. Technical Report 95002, GA Laboratory, University of Illinois, Urbana-Campaign, IL.

Haario, H., Saksman, E., Tamminen, J., 1999. Adaptive proposal distribution for random walk Metropolis algorithm. Computational Statistics 14 (3), 375–395.

Hammond, G.E., Valocchi, A.J., Lichtner, P.C., 2005. Application of Jacobian-free Newton–Krylov with physics-based preconditioning to biogeochemical transport. Advances in Water Resources 28, 359–376.

Hastings, W.K., 1970. Monte-Carlo sampling methods using Markov Chains and their applications. Biometrika 57, 97–109.

Hedenström, A., Pettersson, J., 1987. Migration routes and wintering areas of willow warblers *Phylloscopus trochilus* ringed in Fennoscandia. Ornis Fennica 64, 137–143.

Herrera, F., Lozano, M., Moraga, C., 1998. Hybrid distributed real-coded genetic algorithms. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.P. (Eds.), Parallel Problem Solving from Nature (V). pp. 879–888.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680.

Levenberg, K., 1944. A method for the solution of certain problems in least squares. Quaternary Applied Mathematics 2, 164–168.

Marquardt, D., 1963. An algorithm for least-squares estimation of nonlinear parameters. SIAM Journal on Applied Mathematics 11, 431–441.

Message Passing Interface Forum, 1997. MPI-2: Extensions to the Message-passing Interface. University of Tennessee, Knoxville, TN.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., 1953. Equations of state calculations by fast computing machines. Journal of Chemical Physics 21, 1087–1091.

Moré, J.J., Wu, Z., 1995. ε-optimal solutions to distance geometry problems via global continuation. In: Pardalos, P.M., Shalloway, D., Xue, G. (Eds.), Global Minimization of Nonconvex Energy Functions: Molecular Confirmation and Protein Folding. American Mathematical Society, pp. 151–168.

Nelder, J.A., Mead, R., 1965. A Simplex method for function minimization. Computer Journal 7, 308–313.

Pennycuick, C.J., 1998. Computer simulation of fat and muscle burn in long-distance bird migration. Journal of Theoretical Biology 191, 47–61.

Sorooshian, S., Duan, Q., Gupta, V.K., 1993. Calibration of rainfall-runoff models: Application of global optimization to the Sacramento Soil Moisture Accounting model. Water Resources Research 29, 1185–1993.

Vrugt, J.A., Bouten, W., 2005. Differential flight patterns of migratory birds explained by Pareto optimization of flight time and energy use. To be submitted.

Vrugt, J.A., van Wijk, M.T., Hopmans, J.W., Šimunek, J., 2001. One-, two-, and three-dimensional root water uptake functions for transient modeling. Water Resources Research 37 (10).

Vrugt, J.A., Gupta, H.V., Bastidas, L.A., Bouten, W., Sorooshian, S., 2003a. Effective and efficient algorithm for multiobjective optimization of hydrologic models. Water Resources Research 39 (8), 1214.

Vrugt, J.A., Gupta, H.V., Bouten, W., Sorooshian, S., 2003b. A Shuffled Complex Evolution Metropolis algorithm for optimization and uncertainty assessment of hydrologic model parameters. Water Resources Research 39 (8), 1201.

Vrugt, J.A., Schoups, G., Hopmans, J.W., Young, C., Wallender, W.W., Harter, T., Bouten, W., 2004. Inverse modeling of large-scale spatially distributed vadose zone properties using global optimization. Water Resources Research 40, W06503.

Vrugt, J.A., van Belle, J., Bouten W., 2005. Multi-criteria optimization of animal behavior: Pareto front analysis of flight time and energy-use in long-distance bird migration. Journal of Theoretical Biology, in review.

Winter, C.L., Springer, E.P., Costigan, K., Fasel, P., Mniszewski, S., Zyvoloski, G., 2004. Virtual watersheds: simulating the water balance of the Rio Grande basin. Computing in Science and Engineering 6 (3), 18–26.

Wu, Y.S., Zhang, K., Ding, C., Pruess, K., Elmroth, E., Bodvarsson, G.S., 2002. An efficient parallel-computing method for modeling nonisothermal multiphase flow and multicomponent transport in porous and fractured media. Advances in Water Resources 25, 243–261.

Yapo, P., Gupta, H.V., Sorooshian, S., 1996. Automatic calibration of conceptual rainfall-runoff models: sensitivity to calibration data. Journal of Hydrology 181, 23–48.

Zitzler, E., Thiele, L., 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation 3 (4), 257–271.