Xu, T.; Gómez-Hernández, JJ.; Li ., L.; Zhou ., H. (2013). Paralellized ensemble Kalman filter for hydraulic conductivity characterization. Computers and Geosciences. 52:42-49. doi:10.1016/j.cageo.2012.10.007.

Parallelized Ensemble Kalman Filter for Hydraulic Conductivity Characterization

Teng Xu[a,*], J. Jaime Gómez-Hernández[a], Liangping Li[a], Haiyan Zhou[a]

[a]Group of Hydrogeology, Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain

Abstract

The ensemble Kalman filter (EnKF) is nowadays recognized as an excellent inverse method for hydraulic conductivity characterization using transient piezometric head data. Its implementation is well suited for a parallel computing environment. A parallel code has been designed that uses parallelization both in the forecast step and in the analysis step. In the forecast step, each member of the ensemble is sent to a different processor, while in the analysis step, the computations of the covariances are distributed between the different processors. An important aspect of the parallelization is to limit as much as possible the communication between the processors in order to maximize execution time reduction.

Four tests are carried out to evaluate the performance of the parallelization with different ensemble and model sizes. The results show the savings provided by the parallel EnKF, especially for a large number of ensemble realizations.

Keywords: Parallel EnKF; Cluster; Hydraulic conductivity; Parallel computing;

1. Motivation

The ensemble Kalman filter (EnKF) proposed by Evensen (Evensen, 2003) has proven to be an effective inverse method. It has been applied in many fields such as petroleum

*Corresponding author. Tel: +34 963879615 Fax: +34 963879492

Email addresses: tenxu@posgrado.upv.es (Teng Xu), jaime@dihma.upv.es (J. Jaime Gómez-Hernández), liali@upvnet.upv.es (Liangping Li), haizh@upvnet.upv.es (Haiyan Zhou)

engineering, oceanography, meteorology or hydrology (e.g., Aanonsen et al., 2009; Evensen, 2003; Dowell et al., 2010; Hendricks Franssen and Kinzelbach, 2008; Li et al., 2012b). While it has proven to be more effective than alternative inverse methods (Hendricks Franssen and Kinzelbach, 2009), it still has important computational needs.

The ensemble Kalman filter is based on the simultaneous analysis of a large number of realizations. The first attempts to reduce CPU time usage were aimed at reducing the number of realizations in the ensemble. The covariance localization is a modification of the initial EnKF implementation that serves to reduce the ensemble size without compromising much the quality of the outcome (e.g., Houtekamer and Mitchell, 1998; Anderson, 2007; Devegowda et al., 2010; Zhang and Oliver, 2010). Other authors have proposed to reduce the size of each realization, for instance, Li et al. (2012a) couple upscaling and the EnKF for the inverse modeling of groundwater flow.

In any case, even for the most efficient implementation of the EnKF, the fact that it works on each one of an ensemble of realizations makes it amenable to parallelization and thus take advantage of multi-processor computers or of grid computing to reduce even further the time needed for the algorithm to run. Although it will be reviewed in more detail later in the paper, recall that the EnKF takes an ensemble of realizations, runs a forward model in each realization, collects state data at observation locations and uses the difference between the predicted and observed values to update each one of the realizations. The updating step requires using all the ensemble realizations to compute the Kalman gain.

Keppenne (2000) proposed a parallel algorithm in which the forward model for each ensemble member is run in a different processor, then, to compute the Kalman gain, a domain decomposition is performed and each processor ends up with a small portion of all ensemble realizations, finally, for the updating, each processor is responsible of the update of a realization. (Keppenne and Rienecker, 2002a,b, 2003) also apply a domain decomposition in the updating step and each processor is responsible of updating the subdomain used

for the Kalman gain computation in all realizations. This approach has the advantage of avoiding the ensemble transpositions across processor that would be required if, after the computation of the Kalman gain, each processor updates a full realization.

Recently, Tavakoli et al. (2011) have proposed a parallel EnKF algorithm applied to petroleum engineering using a three-level parallelization. On the first level, each ensemble member runs on a separate processor during the forecast step, on the second level uses a parallel implementation of the reservoir simulator, and, on the third level, it handles the matrix-vector multiplications involved in the Kalman gain computation and the updating of the ensemble members.

In this paper we propose an alternative parallel EnKF algorithm and provide the computer code to run it on a parallel environment using MODFLOW (McDonald and Harbaugh, 1988; Harbaugh et al., 2000) as the forward model. To the best of our knowledge this is the first application of a parallel EnKF algorithm in the field of hydrogeology.

The paper proceeds with an overview of the EnKF, the strategy for parallelization, and the evaluation of the algorithm on two examples.

2. Overview of the EnKF

The EnKF is the evolution of the Kalman filter to better address nonlinear state transfer functions using a Monte-Carlo approach. The Kalman filter was proposed by Kalman et al. (1960) as a data assimilation filter to improve the estimation of the state of a dynamic linear system. Later, the extended Kalman filter (EKF) was proposed to address nonlinear systems; the extension is based on a linearization of the nonlinear model, using a Taylor expansion, for the computation of the time evolution of the covariance (McElhoe, 1966). The EKF has been used in many fields, including hydrology (e.g., Hantush, 1997; Leng and Yeh, 2003; Yeh and Huang, 2005), however, it has severe shortcomings in dealing with highly nonlinear functions due to the accumulative error induced by the linearization process. Besides, the

algorithm itself is time and storage consuming, yielding it infeasible for large-scale system. To overcome these problems, the EnKF was proposed, the specifics of which are introduced next.

There are many uncertain factors in groundwater modeling: initial and boundary conditions, forcing terms, parameter values, ... (Hendricks Franssen and Kinzelbach, 2009). In this paper, we focus on the uncertainty about the parameter log-hydraulic conductivity. The state-transition equation is the standard three-dimensional groundwater flow equation, which is solved by MODFLOW, one of the most popular three-dimensional finite-difference groundwater flow simulators (e.g., McDonald and Harbaugh, 1988; Harbaugh et al., 2000). The filter has two steps. The first one is the prediction step, given by

$$s_i^f(t) = M(s_i^a(t - \Delta t) + w_i(t), w(t) \sim N(0, \sigma) \tag{1}$$

where $s_i^f(t)$ is the forecasted state of the system for a given set of parameters i, this state is function of the last estimate of the state of the system at $t - \Delta t$, $s_i^a(t - \Delta t)$ through a state-transition equation represented by M. Equation M is only an approximation of how the system behaves, therefore a model error w is added to the forecast, which is assumed to be Gaussian distributed, with zero mean and covariance σ.

The second step is the analysis step (Burgers et al., 1998), whereby the observed state measurements are used to update the forecast state to come up with a better estimate of the forecasted state.

$$s_i^a(t) = s_i^f(t) + k(t)[d_o(t) + e_i(t) - Hs_i^f(t)] \tag{2}$$

$$k(t) = p^f(t)H^T[Hp^f(t)H^T + R]^{-1} \tag{3}$$

where the forecasted state $s_i^f(t)$ is updated as a function of the difference between the predicted state $s_i^f(t)$ and the observations $(d_o(t) + e_i(t)$; matrix H is a measurement matrix

that serves to map the model predictions (generally over a fixed grid) onto the locations where the observations are taken. Note that the observations are composed of two terms the "real" state of the system $d_o(t)$ plus a measurement error $e_i(t)$ of mean zero and covariance R. The amount by which the forecasted state should be modified is controlled by the Kalman gain $k(t)$, which is a function of the state covariance $p^f(t)$. The ensemble Kalman filter was developed precisely to avoid the shortcomings in computing the state covariance by linearizing the state-transfer equation. For this purpose, an ensemble of realizations is generated, and their state is forecasted and updated sequentially in time as data are collected. At each time step, the covariance is numerically inferred from the ensemble of realizations,

$$p^f(t) = \frac{1}{N_e - 1} \sum_{i=1}^{N_e} (s^f(t) - \langle s^f(t) \rangle)(s^f(t) - \langle s^f(t) \rangle)^T \tag{4}$$

$$\langle s^f(t) \rangle = \frac{1}{N_e} \sum_{i=1}^{N_e} s^f(t) \tag{5}$$

where N_e is the number of realizations of the ensemble, $\langle \cdot \rangle$ denotes ensemble average, and p is a matrix of dimensions $n \times n$, with n is the number of nodes at which the state of the system is predicted by the numerical forecast model.

In inverse modeling applications, the state of the system is augmented so that it not only includes the properly-speaking state of the system but also the parameters defining the transfer function. In hydrogeology it is common to use such an augmented state, in our case we use piezometric heads and log-hydraulic conductivities as state and parameters, respectively. In our implementation we use an augmented vector, the state transfer function in Eq. (1) leaves unchanged the logconductivity values, and updates the piezometric heads according to the groundwater flow model; then, in the analysis step, we limit ourselves to update the logconductivity values as explained next. For the sake of demonstration, we will assume that observations are taken at groundwater model prediction locations; limiting the

97 component of the measurement matrix H to be 0's and 1's, and simplifying the expressions
98 in Eqs. (2) and (3) as follows. We may partition the covariance matrix

$$p = \begin{bmatrix} p_{hh} & p_{Yh} \\ p_{hY} & p_{YY} \end{bmatrix} \tag{6}$$

99 where p_{hh}, p_{YY} are the covariance between hydraulic heads at measurement locations and
100 the covariance between log-conductivities at all model gridnodes, respectively, and $p_{Yh}$and
101 p_{hY} are the cross-covariances between a log-conductivity and a hydraulic head. And Eqs.
102 (2) and (3) become

$$Y_i^a(t) = Y_i^f(t) + k(t)[d_o(t) + e_i(t) - h_i^f(t)] \tag{7}$$

103

$$k(t) = p_{Yh}[p_{hh} + R]^{-1} \tag{8}$$

104 where $Y_i^a(t)$ and $Y_i^f(t)$ are the elements of the augmented state vector corresponding to the
105 logconductivities, and $h_i^f(t)$ is the predicted piezometric heads at measurement locations.

106 The output of the EnKF is an ensemble of realizations of hydraulic conductivity all
107 of which are "conditioned" to the observation data. From this ensemble we can obtain
108 average estimates and uncertainty estimates about the hydraulic conductivity, or we can
109 post-process these fields to obtain optimal estimates, with their associated uncertainty, of
110 response functions based on the ensemble, such as optimal pumping rates for the dewatering
111 of a construction site. As more observational data are assimilated, the ensemble of hydraulic
112 conductivities become more alike, and therefore the uncertainty associated is smaller, since
113 there are less alternative realizations capable of reproducing the observed entire transient
114 state information.

115 We end this introduction mentioned some of the main advantages and drawbacks of the
116 EnKF. The main advantage of the EnKF is that is not an optimization approach but rather

a filtering approach. Therefore, there is no need for recursive evaluations of expensive cost functions, just the need to compute the ensemble covariance and the mismatch between predictions and observations, which will be used to update the conductivity fields. This characteristic makes the EnKF very easy to implement and to apply in cases with many different sources of information. The main drawback, leaving apart the Monte-Carlo aspect of the method and the need to handle many realizations, is that the EnKF has been found to collapse underestimating the final uncertainty (that is, as the fields keep been updated, they tend to become more and more similar). There are two main reasons for this behavior, one is the appearance of spurious correlations between distant points due to the numerical nature of the covariance calculations, the other has to do with the number of realizations, if it is not large enough, the empirical covariance based on a reduced number of realizations tends to decrease as new updating steps are performed. The approaches to deal with these problems are covariance localization and covariance inflation.

3. Parallelization approach

There are mainly three kinds of parallel computer architectures, the first kind is based on shared memory, the second one is based on distributed memory, and the third one is a hybrid in between both of them. The first type corresponds to multi-core computers, the benefits are that the communication between the different ranks is very fast, and that it is very easy to share data among them. However, this type of architecture is limited by the total amount of memory available, and may be unable to address large models. The second type corresponds to sharing resources among a grid of computers, the grid could easily be enlarged attaching new computers to it, and therefore the limitation about the size of the model they can handle disappears, on the contrary, the communication between the different processors is much slower than in the shared-memory architecture. The best architecture is the third one that uses a grid of multi-core computers, balancing the advantages and

disadvantages of the first and second architecture kinds.

In this paper, we employ a hybrid architecture. The cluster of computers, known as "Pleiades", consists of three HP Proliant DL 580, each with six-core four processors AMD Opteron Model 8439 SE (six-core, 2.8GHz, 6MB L3, 105W), which amounts to a total of 24 cores per computer. All cores are 64-bit, and each computer has 256 GB of RAM. The operating system is CentOS. Communication between processors is via message passing interface (MPI). The cluster is networked via Ethernet TCP/IP.

As already explained previously, the EnKF consists of two steps: forecast and analysis (or updating). We have analyzed the potential for parallelization of both steps.

Fig. 1 shows a flowchart of the proposed parallelization, which is explained next.

3.1. Forecast step

The most straightforward way to parallelize the forecast step in the EnKF is at the realization level (Chen and Zhang, 2006). See box 1 of Fig. 1, let $s(n \cdot N_e)$ be the state vector including all the ensemble members, where n represents state-vector size and N_e is the number of realizations of the ensemble; if there are k processors, then the metric can be decomposed into sub-metrics $s(n \cdot m_0)$, $s(n \cdot m_1)$,..., $s(n \cdot m_k)$, where m_0,m_1,...m_k (m_0+m_1+...+m_k=N_e) denotes the number of ensemble members that must be processed by each processing element, PE_0, PE_1,...PE_k respectively. Once the realizations are distributed among the processors, the MODFLOW simulator is called to generate the forecast piezometric heads in all realizations. The distribution of the realizations to the processors must be so that the load on all processors is as similar as possible. If N_e is a multiple of k, then each processor will receive N_e/k realizations, otherwise there will be $N_e\%k$ processors receiving one extra realization.

Since the processors operate asynchronously, and the analysis step cannot be performed until the forecast is performed in all realization, it is necessary to call the $MPI\_Barrier$

command before starting the update step, to block each processor until all processors reach the barrier point in the code (Dewaraja et al., 2002).

3.2. Update step

After the forecast step, the state vector s^f, which contains all realizations, is distributed between the processors. Next, we have to compute the covariances p_{Yh}, and p_{hh}. This calculation is distributed as follows, first accumulate the distributed state vector s^f in each processor, $\sum s^{f0}$, $\sum s^{f1}$, ..., $\sum s^{fk}$, (recall that each processor is in charge of the forecast of a subset of all realizations) send these accumulated values to one of the processors PE_0 and compute the mean value of each component. Broadcast the mean values to all of the processors, and accumulate the products of the differences of the state vector with respect to their means in each one of the processors $\sum(s^{f0} - < s^f >)(s^{f0} - < s^f >)^T$,...,$\sum(s^{fk} - < s^f >)(s^{fk} - < s^f >)^T$. Then, send the accumulated products of differences to PE_0 and compute the covariances in this processor. Notice that the accumulated products of differences must be computed only for the individual entries in p_{Yh}, and p_{hh}, not for all possible entries of p^f in Eq. (6). With the covariances computed, the Kalman gain is obtained and broadcasted back to all processors so that the updating Eq. (7) is applied to each member of the state vector distributed between the processors.

4. Application

The 3-D transient groundwater flow equation is:

$$S_s \frac{\partial h}{\partial t} - \nabla \cdot (k \nabla h) = W \tag{9}$$

where S_s is specific storage coefficient $[L^{-1}]$; h is the hydraulic head $[L]$; k is the hydraulic conductivity $[LT^{-1}]$; t is the time $[T]$; W denotes sinks and sources per unit volume $[T^{-1}]$.

We use four test cases corresponding to two different model sizes. Cases 1, 2 and 3 use a synthetic model with 50 by 50 by 5 cells, and the difference between them is the number of realizations in the ensemble, which are 1200, 720 and 240. Case 4 uses a synthetic model with 50 by 50 by 1 cell and 1200 realizations in the ensemble. All cells are 5 m by 5m by 2 m.

For the small model of size 50 by 50 by 1 cells, there are 75 observation wells in the domain located as shown in Fig. 2. The left boundary has a specified head boundary equal to 8 m, the right boundary has a specified flow of -0.0008 d^{-1} and the top and bottom boundaries are no flow.

For the large model of size of 50 by 50 by 5 cells, there are 25 observation wells in the domain, as shown in Fig. 3, that monitor the piezometric heads at the first, third and fifth layers, for a total of 75 measurements. A verification well, located at row 30, column 20 and layer 3 (see Fig. 3) is used to test the evolution of the piezometric head at an unsampled location. The first layer of the left boundary is given a specified head of 8 m, the fifth layer of the right boundary is given a prescribed flow of -0.008 d^{-1}. The rest of the boundary are no flow boundaries.

For both models, the initial head is set 8 m throughout the domain. Specific storage S_s is set as 0.0008 m^{-1}. The total simulation time is 500 days, discretized into 100 time steps according to the following progression

$$\triangle t_k = \delta \triangle t_{k-1} \tag{10}$$

where δ is 1.05.

Log-conductivity $\ln(k)$ is assumed to be second-order stationary following a multi-Gaussian distribution of zero ln m/d mean, standard deviation σ=1.5 ln m/d, and an exponential covariance function.

$$r(h) = \sigma^2[1 - exp(-\frac{|h_x|}{\lambda_x} - \frac{|h_y|}{\lambda_y} - \frac{|h_z|}{\lambda_z})] \quad (11)$$

where the integral scales in the x, y, z directions are respectively $\lambda_x = 90$ m,$\lambda_y = 30$ m,$\lambda_z = 5$ m.

The sequential Gaussian simulation module of the GSLIB software (Deutsch and Journel, 1998) is used to generate the log-conductivity realizations. One of the realizations is chosen as the reference field.

4.1. Analysis

Fig. 4 shows the reference log-conductivity field, and Fig. 5a,5b,5c,5d,5e,5f shows the ensemble mean field for the tests 1, 2 and 3, at the beginning of the simulation (when no piezometric head data has been assimilated yet) and at the 50th time step. While comparing Fig.5b,5d,5f and Fig. 4, we find that the main features of the reference field are captured by the EnKF after the 50th data assimilation step. The larger the ensemble size, the smoother is the ensemble mean.

Fig. 6a, 6c and 6e shows the piezometric heads in the control well (not used for conditioning) computed on the initial ensemble of logconductivity realizations. They display a very large variability indicative of large prediction uncertainty. This variability is reduced when the piezometric heads are computed on the updated realizations, as seen in Fig. 6b, 6d and 6f.

We can evaluate the goodness of the estimated field using the average deviation between the average of the ensemble members and the reference field using the root mean square error ($RMSE$).

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(s^{ref} - \langle s^a \rangle)^2} \quad (12)$$

Where N is the number of model gridblocks; s^{ref} is the value of the reference field; $\langle s^a \rangle$ denotes the mean estimation of the ensemble fields.

Fig. 7 shows that the $RMSE$ of test 3 has some fluctuations at early assimilation steps, yet after the 31st assimilation step, it begins to increase, becoming even larger than the starting value, which implies that for a small ensemble size the estimation of the covariance is poor (Hendricks Franssen and Kinzelbach, 2008). In the comparison of the $RMSE$ between tests 1 and 2, the $RMSE$ of test 1 is lower, plus, it is gradually smoother after the 27th assimilation step. So it can be concluded that the larger the ensemble, the better the estimation.

Similar results and conclusions can be reached for fourth test case, for which the model size is smaller while retaining the same number of measurements.

4.2. Parallelization analysis

Speedup and efficiency are usually used to evaluate the performance of parallel algorithms

$$\text{Speedup:} S_P = \frac{T_s}{T_P} \tag{13}$$

$$\text{Efficiency:} E_P = \frac{S_P}{P} \tag{14}$$

P is the number of the processors, T_s is the CPU time consumed under a serial implementation of the algorithm, T_P is the CPU time consumed under a parallel implementation with P processors.

There is an obvious trade-off between the ensemble size and the CPU time, and also between the size of the numerical model and the CPU time. Table 1 and Fig. 8 show the performance of the parallel algorithm for test cases 1 (three-dimensional domain and 1200 realizations), 3 (three-dimensional domain and 240 realizations) and 4 (two-dimensional domain and 1200 realizations).

From the table and figure we can see that the speedup is far from the ideal line for which the improvement in CPU should follow the same proportion as the number of processors. This is not a surprising result, there are two causes for this departure from the ideal performance: there is a need for all processors to wait until they have finished certain tasks for them to proceed on to the next task, and there is extra time needed for communication between processors.

We notice some differences between the tests. Comparing tests 1 and 3, we notice that the speedup is better for the case with the larger number of realizations, this is because for test 3, each processor receives a smaller number of realizations and thus, proportionally, the time spent in communication is larger for test 3 than for test 1. Apparently, for both tests, the speedup could keep increasing if more processor were available. For test 4, however, the speedup is similar to that of test 1 up to eight processors, then it appears to worsen, again, the cause is found in the increase in the communication time among processors. It does not seem that increasing the number of processors past 8 will increase the speedup (although the overall CPU time will be still reduced) for test 4.

The efficiency curves have a similar behavior as the speedup curves. Efficiency worsens past 8 processors for test 4, and for tests 1 and 3 we can conclude that the parallel algorithm is more efficient the larger the ensemble size.

We can conclude that the parallel implementation of the EnKF runs with higher efficiency for large size models and large ensembles than for small ones. In all cases, the final CPU time is smaller than for the serial implementation, although the speedup is still far from ideal.

In this parallel algorithm there are two inherent barriers to its processing capabilities, one is data asynchrony, and the other is data-dependency. Data asynchrony makes all processors be as slow as the slowest one, since they have to wait for all processors to finish a certain task before they can proceed to the next one. Data dependency refers to the fact that the

distribution of realizations to the processors does not necessarily leaves all processors with the same number of realizations. Data asynchrony mainly affects the forecast step, while data dependency affects all steps and it is less noticed for large number of realizations.

5. Summary and suggestion

A parallel algorithm for the forecast and analysis steps of the EnKF has been presented that reduces significantly the time to run the EnKF for large ensemble sizes. The efficiency remains over 0.40 when using up to 12 processors for the two tests using 1200 realizations.

There are many measures that could be taken to reduce the communication time and increase the efficiency such as decreasing the communication traffic, boosting communication granularity or using a high-speed internet protocol to reduce the information transfer delay. In addition, we must attempt to keep the load as balanced as possible among the processors, that is, all processors should work on the same (or very similar) number or realizations.

Furthermore, the algorithm could be improved using a parallelized version of the groundwater flow simulator, such as one employing domain decomposition, especially for large size models.

Acknowledgements

The first author acknowledges the financial support from China Scholarship Council (CSC). Financial support to carry out this work was also received from the Spanish Ministry of Science and Innovation through project CGL2011-23295, and from the Universitat Politècnica de València through project PERFORA.

Aanonsen, S., Nævdal, G., Oliver, D., Reynolds, A., Vallès, B., 2009. The ensemble Kalman filter in reservoir engineering–a review. SPE Journal 14, 393–412.

Anderson, J., 2007. Exploring the need for localization in ensemble data assimilation using a hierarchical ensemble filter. Physica D: Nonlinear Phenomena 230, 99–111.

303 Burgers, G., van Leeuwen, P., Evensen, G., Instituut, K.N.M., 1998. Analysis scheme in the
304 ensemble Kalman filter. Monthly weather review 126, 1719–1724.

305 Chen, Y., Zhang, D., 2006. Data assimilation for transient flow in geologic formations via
306 ensemble Kalman filter. Advances in Water Resources 29, 1107–1122.

307 Deutsch, C., Journel, A., 1998. Gslib: Geostatistical Software Library and user's Guide.
308 New York 369.

309 Devegowda, D., Arroyo-Negrete, E., Datta-Gupta, A., 2010. Flow relevant covariance local-
310 ization during dynamic data assimilation using enkf. Advances in Water Resources 33,
311 129–145.

312 Dewaraja, Y., Ljungberg, M., Majumdar, A., Bose, A., Koral, K., 2002. A parallel Monte
313 Carlo code for planar and spect imaging: implementation, verification and applications in
314 131i spect. Computer Methods and Programs in biomedicine 67, 115–124.

315 Dowell, D., Zhang, F., Wicker, L., Snyder, C., Crook, N., 2010. Wind and temperature
316 retrievals in the 17 May 1981 arcadia, oklahoma, supercell: Ensemble Kalman filter ex-
317 periments .

318 Evensen, G., 2003. The ensemble Kalman filter: Theoretical formulation and practical
319 implementation. Ocean Dynamics 53, 343–367.

320 Hantush, M., 1997. Estimation of spatially variable aquifer hydraulic properties using
321 Kalman filtering. Journal of Hydraulic Engineering 123, 1027.

322 Harbaugh, A., et al., 2000. MODFLOW-2000, the US Geological Survey modular ground-
323 water Model: User Guide to Modularization Concepts and the Ground-water flow process.
324 US Geological Survey.

Hendricks Franssen, H., Kinzelbach, W., 2008. Real-time groundwater flow modeling with the ensemble Kalman filter: Joint estimation of states and parameters and the filter inbreeding problem. Water Resources Research 44, W09408.

Hendricks Franssen, H.J., Kinzelbach, W., 2009. Ensemble Kalman filtering versus sequential self-calibration for inverse modelling of dynamic groundwater flow systems. Journal of Hydrology 365, 261–274.

Houtekamer, P., Mitchell, H., 1998. Data assimilation using an ensemble Kalman filter technique. Monthly Weather Review 126, 796–811.

Kalman, R., et al., 1960. A new approach to linear filtering and prediction problems. Journal of Basic Engineering 82, 35–45.

Keppenne, C., 2000. Data assimilation into a primitive-equation model with a parallel ensemble Kalman filter. MONTHLY WEATHER REVIEW-USA 128, 1971–1981.

Keppenne, C., Rienecker, M., 2002a. Development and initial testing of a parallel ensemble Kalman filter for the poseidon isopycnal ocean general circulation model. Monthly Weather Review 130, 2951–2965.

Keppenne, C., Rienecker, M., 2002b. Initial testing of a massively parallel ensemble Kalman filter with the poseidon isopycnal ocean general circulation model. Monthly Weather Review(0027-0644) 130, 2951–2965.

Keppenne, C., Rienecker, M., 2003. Assimilation of temperature into an isopycnal ocean general circulation model using a parallel ensemble Kalman filter. Journal of marine systems 40, 363–380.

Leng, C., Yeh, H., 2003. Aquifer parameter identification using the extended Kalman filter. Water resources research 39, 1062.

Li, L., Zhou, H., Franssen, H., Gómez-Hernández, J., 2012a. Modeling transient groundwater flow by coupling ensemble Kalman filtering and upscaling. Water Resources Research 48, W01537.

Li, L., Zhou, H., Gómez-Hernández, J., Hendricks Franssen, H., 2012b. Jointly mapping hydraulic conductivity and porosity by assimilating concentration data via ensemble Kalman filter. Journal of Hydrology 428-429, 152–169.

McDonald, M., Harbaugh, A., 1988. A modular three-dimensional finite-difference groundwater flow model, techniques of water-resources investigations, book 6, chapter a1. US Geological Survey 1, 988.

McElhoe, B.A., 1966. An assessment of the navigation and course corrections for a manned flyby of Mars or Venus, in: IEEE Transactions on Aerospace and Electronic Systems, Supplement, IEEE. pp. 613–623.

Tavakoli, R., Pencheva, G., Wheeler, M., 2011. Multi-level parallelization of ensemble Kalman filter for reservoir history matching, in: SPE Reservoir Simulation Symposium.

Yeh, H., Huang, Y., 2005. Parameter estimation for leaky aquifers using the extended Kalman filter, and considering model and data measurement uncertainties. Journal of Hydrology 302, 28–45.

Zhang, Y., Oliver, D., 2010. Improving the ensemble estimate of the Kalman gain by bootstrap sampling. Mathematical Geosciences 42, 327–345.

Figure 1: The parallel EnKF flow chart: box 1 is for the forecast step; box 2 is for the updating step.

Figure 2: Flow domain and location of the 75 observation wells in the small model.

Figure 3: Flow domain and location of the 25 observation wells in the large model. The filled triangles are observation wells measuring the piezometric heads at the first, third and fifth layers, and the filled circle is a verification well.

Figure 4: Reference realization of $ln(k)$.

Figure 5: initial and updated ensemble means facies fields.(a), (b) show the ensemble mean facies fields of the initial realizations and the 50th updated realizations of test 1; (c), (d) show the ensemble mean facies fields of the initial realizations and the 50th updated realizations for test 2; (e), (f) show the ensemble mean facies fields of the initial realizations and the 50th updated realizations for test 3.

Figure 6: Piezometric head time evolution at the control well. The red curve is for the reference field, the gray curves for each realization of the ensemble, and the green curve is the ensemble mean. (a) and (b) show piezometric heads of the the initial realizations and the 50th updated realizations for test 1; (c) and (d) show piezometric heads of the initial realizations and the 50th updated realizations for test 2; (e) and (f) show piezometric heads of the initial realizations and the 50th updated realizations for test 3. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

Figure 7: The root mean square error for three test.

Figure 8: Parallel speed-up and efficiency. (a) Speed up (b) efficiency.

Table 1: Parallel performance of three of the tests.

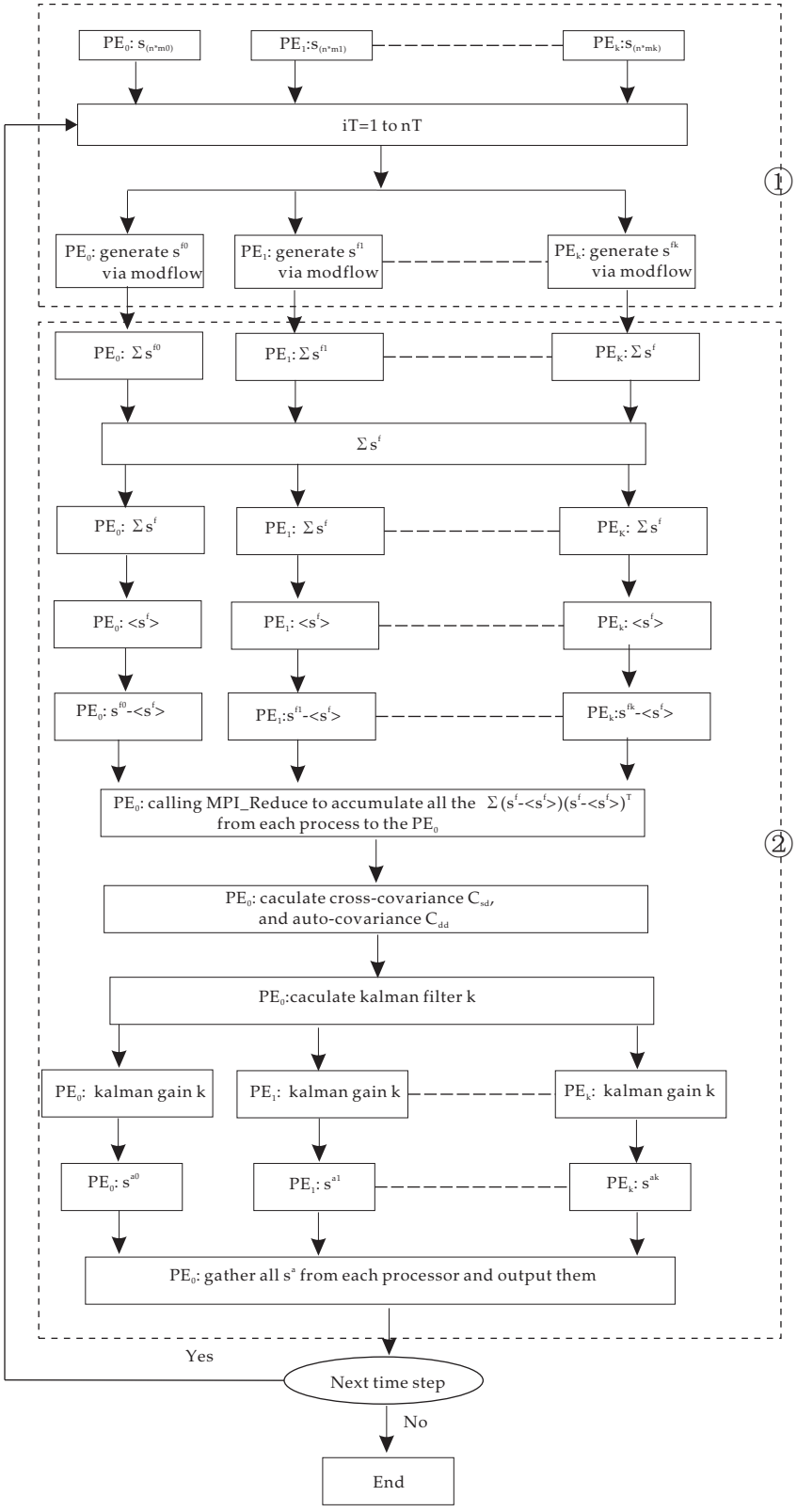

PE0: s(n*m0)
PE1:s(n*m1)
PEk:s(n*mk)
iT=1 to nT
1
PE0: generate sf0 via modflow
PE1: generate sf1 via modflow
PEk: generate sfk via modflow
PE0: Σsf0
PE1:Σsf1
PEk:Σsf
Σsf
PE0: Σsf
PE1: Σsf
PEK: Σsf
PE0: <sf>
PE1: <sf>
PEk: <sf>
PE0: sf0-<sf>
PE1:sf1-<sf>
PEk:sfk-<sf>
PE0: calling MPI_Reduce to accumulate all the Σ(sf-<sf>)(sf-<sf>)T from each process to the PE0
2
PE0: caculate cross-covariance Csd, and auto-covariance Cdd
PE0:caculate kalman filter k
PE0: kalman gain k
PE1: kalman gain k
PEk: kalman gain k
PE0: sa0
PE1: sa1
PEk: sak
PE0: gather all sa from each processor and output them
Yes
Next time step
No
End

Figure 1

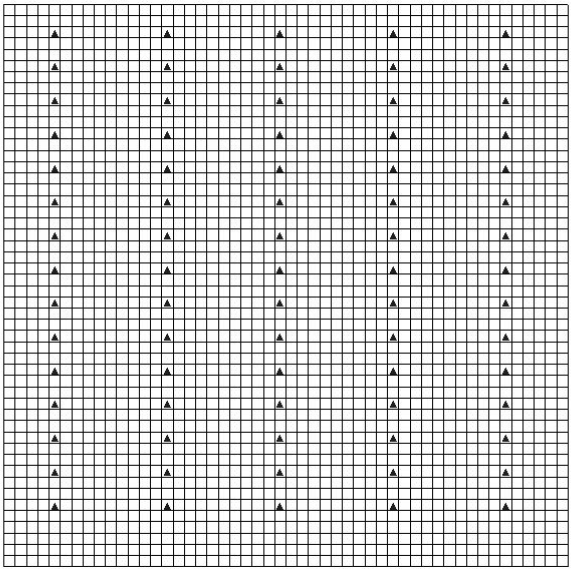


Figure 2

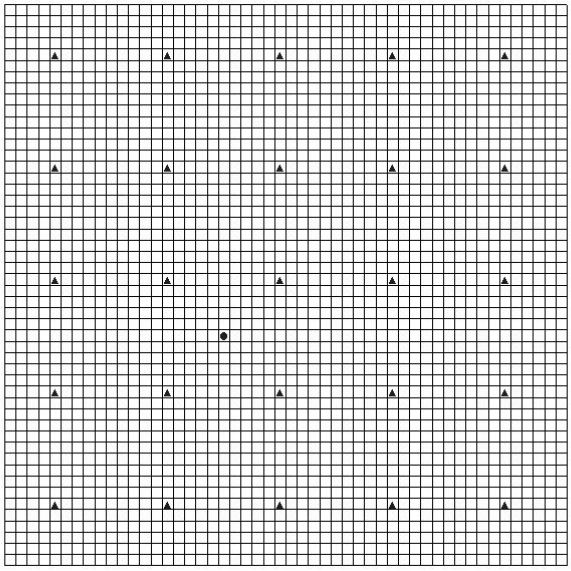


Figure 3

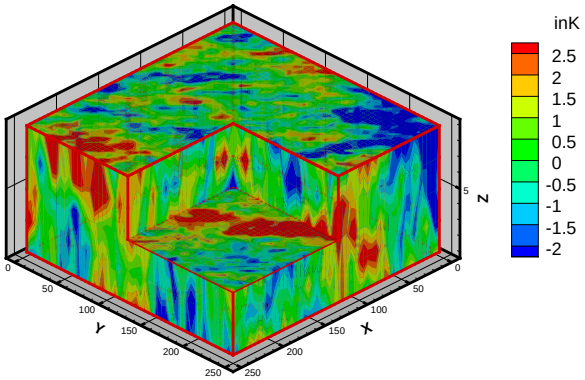


Figure 4

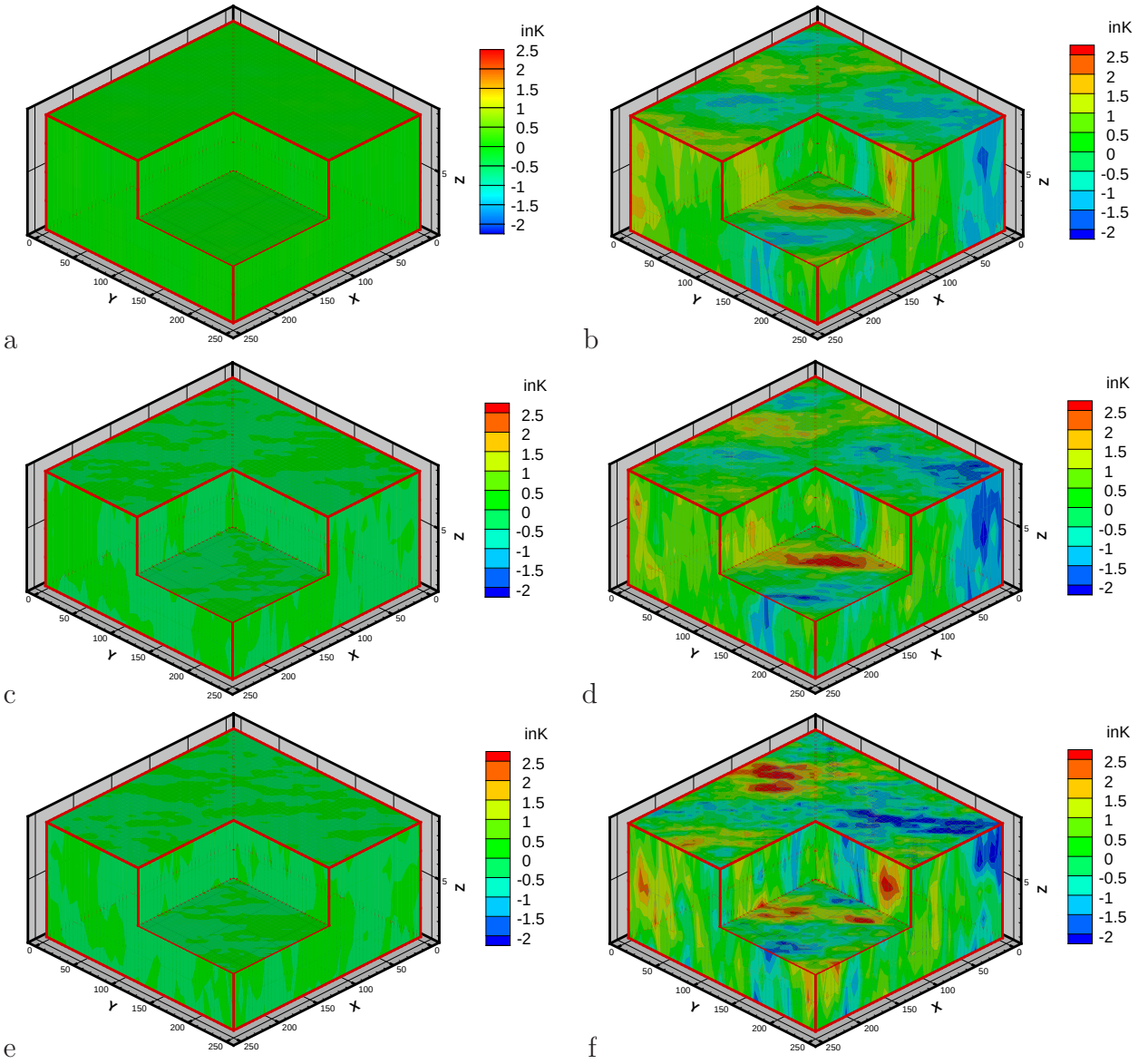


Figure 5

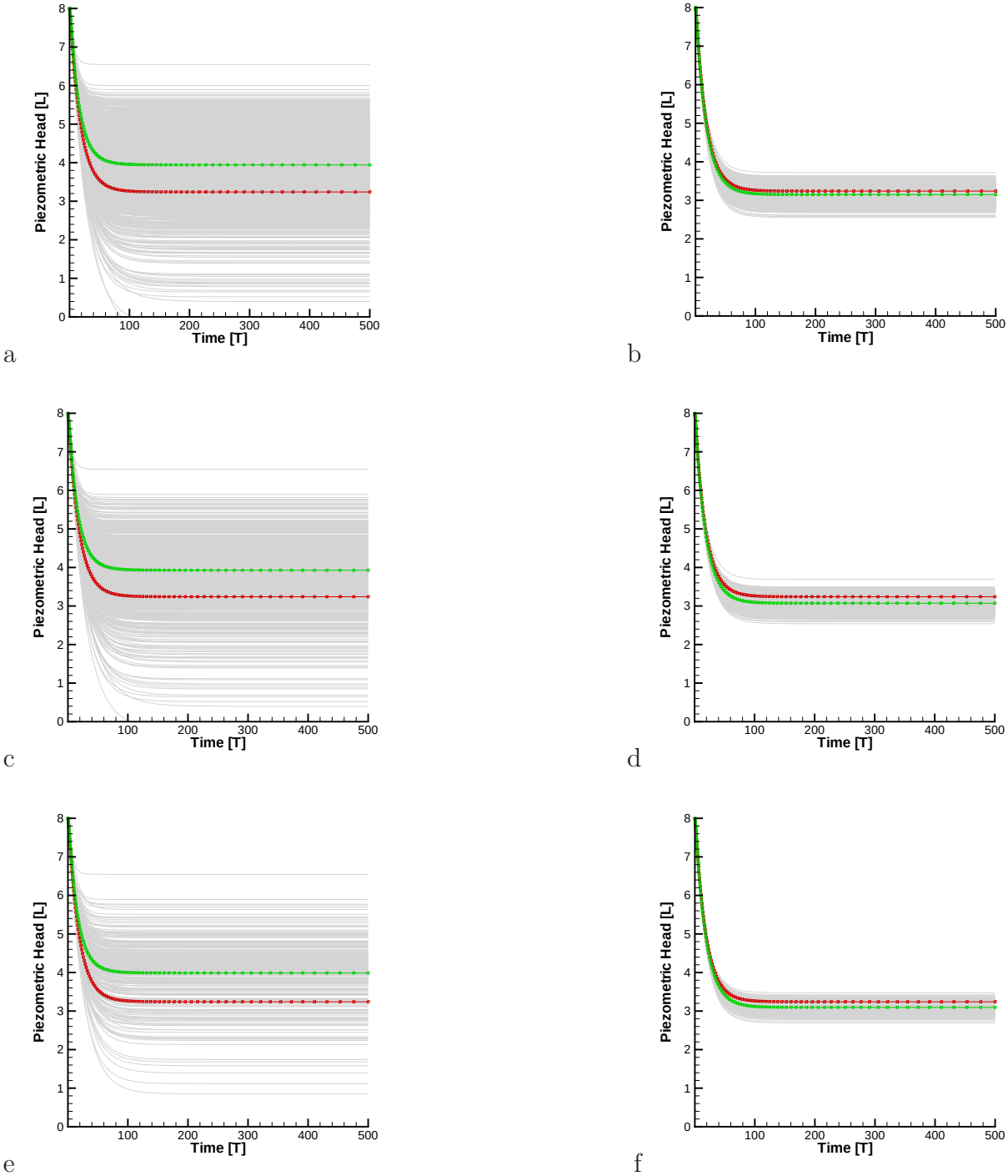


Figure 6

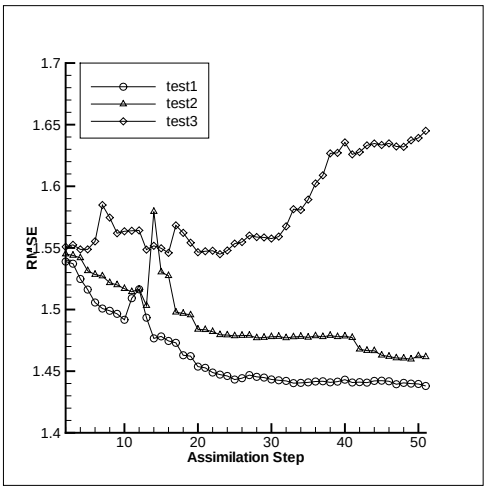
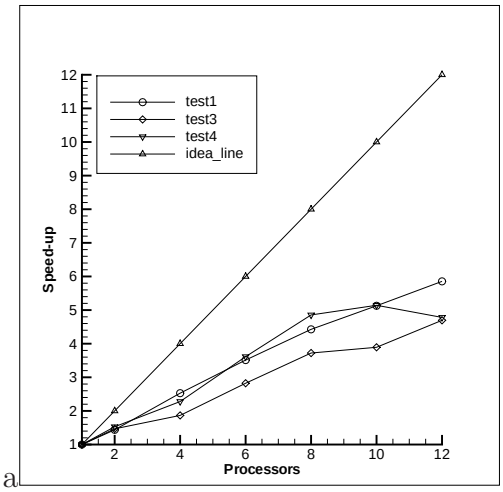

test1
test2
test3
RMSE
Assimilation Step
1.7
1.65
1.6
1.55
1.5
1.45
1.4
10
20
30
40
50

Figure 7

a 
test1
test3
test4
idea_line
Speed-up
Processors
1
2
3
4
5
6
7
8
9
10
11
12
2
4
6
8
10
12

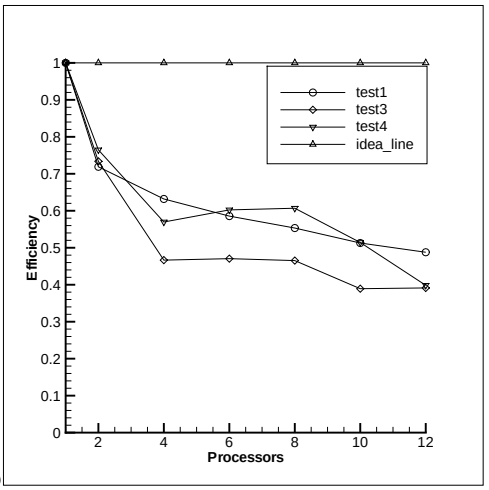
b 
test1
test3
test4
idea_line
Efficiency
Processors
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
2
4
6
8
10
12

Figure 8

Table 1

Processors	Test1CPUs	Speedup	Efficieny	Test3CPUs	Speedup	Efficieny
1	20539.92	1.00	1.00	4866.50	1.00	1.00
2	14284.37	1.44	0.72	3316.35	1.47	0.73
4	8125.83	2.53	0.63	2607.83	1.87	0.47
6	5846.27	3.51	0.59	1723.75	2.82	0.47
8	4640.38	4.43	0.55	1307.34	3.72	0.47
10	4005.09	5.13	0.51	1250.23	3.89	0.39
12	3508.46	5.85	0.49	1036.38	4.70	0.39

Processors	Test4CPUs	Speedup	Efficieny
1	10087.23	1.00	1.00
2	6595.59	1.53	0.76
4	4426.79	2.28	0.57
6	2790.85	3.61	0.60
8	2077.76	4.85	0.61
10	1961.45	5.14	0.51
12	2109.46	4.78	0.40