Parallel Parameter Estimation in Groundwater Flow Models

Lutz Lämmer[1], Michael Burghardt[2], Carola Blömer[3] and Christoph König[3]
[1]Department of Mechanical and Chemical Engineering,
Heriot-Watt University, Edinburgh, UK,
[2]Institute for Numerical Methods and Informatics in Civil Engineering,
University of Technology Darmstadt
[3]GKW Ingenieurgesellschaft mbH, Bochum, GERMANY

Summary

The paper presents a parallel approach to the parameter identification problem for groundwater flow modeling formulated as an optimization problem. This approach is feasible as an add-on to existing modeling software. After reviewing the possibilities and the theoretically obtainable speed-up a solution is presented suitable both for homogeneous and heterogeneous parallel distributed memory architectures.
Key words: finite element method, groundwater modeling, parameter estimation, parallelization

1 Introduction

Numerical simulation of groundwater flow is essential for a number of highly sensitive decision processes using natural resources. Managing complex ecological systems, e.g. groundwater as a major source for drinking water supply relies on sufficiently validated numerical models. The finite element method as a numerical method to solve the governing equations for groundwater flow is well established. But designing a valuable model suitable to investigation of the natural phenomena is not straightforward. The information on the natural system is often rather isolated and the quality obtained by the numerical simulation mainly depends on the experience and intuition of the engineer. The model idealizations and the usage of a chosen discretisation have to be justified by a calibration process. Material properties, source terms, as well as boundary and initial conditions are subject to special investigation in a process of adjusting the mathematical model of the given physical system to allow for certain desired system outputs. The objective function of the optimization problem is the minimization of a quadratic functional - the least square of the error in the observation data.

After describing the governing equations and the types of parameter to adjust to a certain model answer in Section 2 we review available non-linear optimization methods in Section 3. In the finite element context the application of deterministic approaches using derivatives of the functional is highly recommended. The Jacobian of the functional is feasible. A wide range of efficient solution procedures, namely of Gauss-Newton type, especially the Levenberg-Marquardt Algorithms, or the cheaper but sometimes not that efficient Quasi-Newton approaches computing only approximate derivatives, is available. The main advantage of deterministic optimization methods is the availability of sensitivity information on the estimated parameters at almost no extra cost. This is the key to identifying problems in the model, justifying assumptions on material properties and boundary conditions.

The parallel approach to this type of optimization methods is straightforward. We use the master-worker concept to exploit the parallelism in the independent computations of the elements of the Jacobian matrix. The possible gain consists in the simultaneous computation by the number of available worker programs. We have to pay with an increased data organization overhead to initialize and control the parallel processes. If these costs become larger than the advantage gained by distributing the work it is not advised to increase the number of workers furthermore.

After evaluating the theoretical parallel properties of the proposed solution in Section 4 we describe one possible implementation into an existing production finite element code in Section 5. The selection of PVM as the message-passing library is motivated and the impact onto the existing code is outlined. Finally, we show test results of the implementation on a homogeneous multiprocessor system.

2 Governing equations

In this section we describe the momentum and continuity equations resulting in the partial differential equation for groundwater flow.

2.1 Momentum equation - Darcy's law

The momentum equation

$$q = -\mathbf{K}\nabla h \tag{1}$$

is derived form the general equation of motion (Navier-Stokes) under the following assumptions (compare [6]):

- only gravity and pressure gradient are forcing onto the fluid,

- fluid velocity is small enough to neglect effects of inertia,
- flux resistance is caused by viscosity of the fluid and friction, and is proportional to the volume flux (Darcy), and
- a representative element volume defines macroscopic parameter and properties of the fluid.

$\mathbf{K}$ describes the hydraulic conductivity tensor

$$\mathbf{K} = \begin{bmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{yx} & K_{yy} & K_{yz} \\ K_{zx} & K_{zy} & K_{zz} \end{bmatrix} = \frac{\rho g}{\mu}\mathbf{k} \tag{2}$$

with ρ the fluid density, g the gravity constant and μ the Eddy viscosity, and is derived from $\mathbf{k}$, the symmetric tensor of intrinsic permeability, given by

$$\mathbf{k} = \begin{bmatrix} k_{xx} & k_{xy} & k_{xz} \\ k_{yx} & k_{yy} & k_{yz} \\ k_{zx} & k_{zy} & k_{zz} \end{bmatrix}. \tag{3}$$

h describes the pressure head which can be written for incompressible fluids as

$$h = \frac{p}{\rho g} + z. \tag{4}$$

2.2 Continuity equation

Flux in saturated porous media is described by

$$\frac{\partial(\phi\rho)}{\partial t} + div(\rho q) + \rho q' = 0 \tag{5}$$

with ϕ the specific porosity, and q' the discharge. The first term describes the change of mass and can be written as

$$\frac{\partial(\phi\rho)}{\partial t} = \left(\rho\frac{\partial\phi}{\partial p} + \phi\frac{\partial\rho}{\partial p}\right)\frac{\partial p}{\partial t} = \rho(\alpha + \phi\beta)\frac{\partial p}{\partial t} \tag{6}$$

with the specific compressibility of the solid

$$\alpha = \frac{\partial\phi}{\partial p} \tag{7}$$

and the fluid

$$\beta = \frac{\partial\rho}{\rho\partial p}. \tag{8}$$

Constant density ($\rho = \rho_0$) yields

$$\frac{\partial h}{\partial t} \approx \frac{\partial p}{\rho g \partial t}. \tag{9}$$

Introducing the specific storage S by

$$S = \rho g(\alpha + \phi\beta) \tag{10}$$

results finally in

$$divq = S\frac{\partial h}{\partial t} + q'. \tag{11}$$

2.3 Partial differential equation for groundwater flow

Combining Equations 1 and 11 yields the general formulation for the partial differential equation of groundwater flow with

$$div(-\mathbf{K}\nabla h) = S\frac{\partial h}{\partial t} + q'. \tag{12}$$

In the stationary case with $\frac{\partial h}{\partial t} = 0$ Equation (12) reduces to

$$div(-\mathbf{K}\nabla h) = q'. \tag{13}$$

Three types of boundary conditions are imposed,

essential boundary conditions (prescribed fixed pressure heads)

$$h(x,t) = h_{BC}(x,t), \quad x \in \Gamma_1 \tag{14}$$

natural boundary conditions (Neumann BC, boundary discharge)

$$-\mathbf{K}\frac{\partial h}{\partial n}(x,t) = q(x,t), \quad x \in \Gamma_2 \tag{15}$$

mixed boundary conditions (Cauchy BC, leakage coefficients reformulated as $q_{leak} = \alpha(h_{leak} - h)$)

$$-\mathbf{K}\frac{\partial h}{\partial n}(x,t) = u(h,x,t), \quad x \in \Gamma_3 \tag{16}$$

Additionally, for transient problems initial values originating from a stationary, measured or previous transient state are imposed by

initial conditions

$$h(x,0) = h_0(x), \quad x \in \Omega. \tag{17}$$

3 Nonlinear optimization

In the following, we are giving a general framework for minimization problems as later applied to the parameter estimation problem. For a given problem

$$y = f(p) \tag{18}$$

with $y = [y_1, y_2, \ldots, y_m]$ model answers, and $p = [p_1, p_2, \ldots, p_n]$ model parameters the optimization problem consists of determining model parameters in such a way, that the model answers correlate to the measured data. For the problem described in Equations (13) or (13) model parameters comprise a selection of permeability $\mathbf{K}$, specific storage S, discharge q', prescribed hydraulic heads as essential boundary conditions, boundary discharge and leakage coefficients as mixed boundary conditions. Model answers comprise calculated pressure heads and fluxes.

The optimization is performed by minimizing in a least square sense

$$F(p) = \sum_{i=1}^{m} w_i r_i^2(p), \tag{19}$$

with w_i as a weight chosen by the user and r_i the residual

$$r_i(p) = f_i(p) - y_i. \tag{20}$$

The minimization problem describes the the sensitivity of the m model variables from the n model parameters. Therefore, it describes a special mapping problem $\Re^m \mapsto \Re^n$.

Taylor expansion of Equation (19) yields

$$\begin{aligned} F(p+d) &= F(p) + \nabla f(p)^T d + \\ &\quad + \frac{1}{2} d^T \nabla^2 f(p) d + \mathbf{O}(\| \mathbf{d} \|^3) \end{aligned} \tag{21}$$

Introducing the gradient $g(p) = \nabla f(p)$ yields

$$g_k = \frac{\partial F}{\partial p_k} = 2 \sum_{i=1}^{m} w_i r_i \frac{\partial r_i}{\partial p_k}. \tag{22}$$

Partial derivatives of the residuals to the model parameters are arranged in the so-called Jacobian matrix $J \in \Re^{n \times m}$ with the elements

$$J_{ik} = \frac{\partial r_i}{\partial p_k} = \frac{\partial f(p)}{\partial p_k}. \tag{23}$$

The matrix is dense and has the following structure

$$J^T(p) = \begin{bmatrix} \frac{\partial f_1(p)}{\partial p_1} & \frac{\partial f_2(p)}{\partial p_1} & \ldots & \frac{\partial f_m(p)}{\partial p_1} \\ \frac{\partial f_1(p)}{\partial p_2} & \frac{\partial f_2(p)}{\partial p_2} & \ldots & \frac{\partial f_m(p)}{\partial p_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1(p)}{\partial p_n} & \frac{\partial f_2(p)}{\partial p_n} & \ldots & \frac{\partial f_m(p)}{\partial p_n} \end{bmatrix}. \tag{24}$$

The Hessian $H \in \Re^{m\times m} = \nabla^2 f(p)$ consists of the elements

$$H_{kl} = \frac{\partial^2 F}{\partial p_k \partial p_l} = 2\sum_{i=1}^{m} w_i \left(\frac{\partial r_i}{\partial p_k}\frac{\partial r_i}{\partial p_l} + r_i \frac{\partial^2 r_i}{\partial p_k \partial p_l} \right). \tag{25}$$

Combining weights and residuals into the vectors w and r and neglecting in Equation (25) mixed derivatives in the second term of the sum yields

$$H_{kl} \approx 2\sum_{i=1}^{m} w_i \left(\frac{\partial r_i}{\partial p_k}\frac{\partial r_i}{\partial p_l} \right), \tag{26}$$

and for gradient and Hessian matrix

$$\begin{aligned} g &= 2J^T wr \\ H &\approx 2J^T wJ. \end{aligned} \tag{27}$$

The required exact derivatives of the numerical problem, see Equation (12), can be evaluated either by finite difference approximation, direct differentiation or by an adjoint state method. We are using the direct differentiation method resulting in a special finite element problem for every parameter to be optimized. Details can be found in [4].

calculate initial solution	$Z(p)$	$= \|\|f(p)\|\|^2$
† calculate Jacobian		
solve	0	$= g^k + H^k d^k$
update	p^{k+1}	$= p^k + \alpha d^k k$
calculate improved solution	$Z(p)$	$= \|\|f(p)\|\|^2$
repeat from † until convergence.		

Figure 1: Gauss-Newton algorithm

Gauss-Newton method Classical Gauss-Newton iteration assumes exact quadratic approximation in Equation (21) and ignores higher order terms. The minimum is achieved for

$$0 = g^k + H^k d^k \tag{28}$$

The line search modification accounting the minimum of $f(p^k + \alpha d^k)$ for additional damping speeds up the algorithm. The computational requirements for computing and factorising the Hessian can be reduced with the Quasi-Newton approach (see below).

For non-positive definite Hessian H^k Equation (28) has more than one solution. Significant improvements to the actual solution require the application of a heuristically determined matrix S in

$$(g^k + S)d^k \quad = \quad -g^k. \tag{29}$$

This approach is used for example by the Levenberg-Marquardt-algorithm.

$$\begin{array}{rll}
\text{calculate initial solution} & Z(p) & = ||f(p)||^2 \\
\dagger \quad \text{calculate Jacobian} & & \\
\text{chose step-width} & \lambda & = \begin{cases} \text{if} \quad F^{k+1} < F^k & \lambda^{k+1} = \lambda^k / \delta \\ \text{else} & \lambda^{k+1} = \lambda^k * \delta \end{cases} \\
\text{solve} & 0 & = g^k + (H^k + \lambda^k D)d^k \\
\text{update} & p^{k+1} & = p^k + \alpha d^k \\
\text{if} \quad F^{k+1} < F^k \quad \text{calculate improved solution} & Z(p) & = ||f(p)||^2 \\
\text{repeat from} \quad \dagger \quad \text{until convergence.} & &
\end{array}$$

Figure 2: Levenberg-Marquardt method

Levenberg-Marquardt method [5] This method applies Equation (29) to a matrix S defined as a diagonal matrix D, identical to the main diagonal of the Hessian H^k. With a chosen factor λ^k the new direction d^k is obtained by

$$(g^k + \lambda^k D)d^k \quad = \quad -g^k \tag{30}$$

The scalar value λ^k adjusts the method between a Gauss-Newton step (for $\lambda \mapsto 0$) and an iteration step in the steepest descent (for $\lambda \mapsto \infty$) and restricts the step-width in the parameter space. Therefore, the method is characterized as a trust-region algorithm.

Quasi-Newton method This algorithm approximates the Hessian H^{k+1} from the Hessian H^k significantly reducing the computational requirements of the methods. Of the several different updating schemes for determining H^k, the two most common ones are the Broyden, Fletcher, Goldfarb, and Shanno (BFGS) and the Davidson, Fletcher, and Powell (DFP) algorithms. Defining two vectors $s = p^{k+1} - p^k$ and $t = g^{k+1} - g^k$ describing the changes in the co-ordinates and in the gradients, respectively, the approximate Hessian is given by the following in the DFP method:

$$H^{k+1} = H^k - \frac{H^k t (H^k t)^T}{t^T H^k t} + \frac{ss^T}{s^T t}. \tag{31}$$

Alternatively, the BFGS method uses

$$H^{k+1} = H^k + \left(1 + \frac{t^T H^k t}{s^T t}\right) \frac{ss^T}{s^T t} - \left(\frac{st^T H^k + H^k t s^T}{s^T t}\right). \tag{32}$$

In practice, the BFGS method is preferred over the DFP method, because BFGS has been shown to converge globally with inexact line searches, while DFP has not.

4 Parameter estimation as part of the modeling process

From the numerical point of view, solving Equations (12) or (13) requires the conductivity tensor $\mathbf{K}$, the specific storage S and the appropriate boundary conditions and initial values for the transient model, respectively. These physical parameters are determined by measurements or long-term observation data, in-situ experiments, or by extrapolation from laboratory experiments. Whereby the discretised physical data for the numerical model are required for the finite element model, the properties of the aquifer are assumed just at a small number of selected points. Extra- and interpolation or the assumption of regions with constant properties or the combination of both approaches are common techniques to define the required model data [7].

The verification and justification of an assumed property distribution by trial simulations and the minimization of the difference between observed data and modeled data are the essential steps for the calibration of the chosen model. The minimization problem can be directly formulated as an optimization model. Primary model unknowns, boundary conditions, and initial values are the unknowns in the process known as a non-linear optimization problem to be solved with the algorithms shown before.

The formulation of the calibration as a non-linear optimization problem is an essential step to extend an existing simulation code for the parameter estimation. The optimization kernel works as a driver which calls modules to solve the finite element problem (the so-called direct problem) and calculates the elements of the Jacobian matrix. The structure chart of the program is given in Figure 3. The emphasized step of calculation of the Jacobian matrix requires the solution to a special finite element problem the same size of the direct problem for every parameter to be optimized.

For two benchmark problems we have measured the runtime spent in the most time consuming part of the optimization, namely the calculation of the

Initialization	
	Solution of the direct problem
	Calculation of the Jacobian matrix
	Calculation of the gradient
	Solution to the minimization step
	Update of the model parameter
	Error analysis
Convergence check	
Sensitivity analysis	

Figure 3: Structure graph of the serial parameter estimation

Jacobian matrix. Results are shown in Table 1. The data in the column "Jacobian" describe the fraction of the total runtime of the simulation spent in this part of the program.

Table 1: Runtime distribution in the serial parameter estimation

	No. of params.	No. of nodes	No. of elems.	Problem type	Jacobian [%]
3	14	9040	9055	transient	79.8
4	47	6554	6130	stationary	70.7

5 Parallel parameter estimation

The straightforward approach to parallelize the parameter estimation is to distribute the work spent in the calculation of the Jacobian matrix. Without parallelizing the core solver of the direct problem, we have the choice to distribute the work load accordingly to the number of parameters. An appropriate model is the master-worker concept. The master program comprises the standard simulation package extended by the organization of and communication with simultaneously running worker programs. The worker program consists of organization and communication routines and the core finite element solver to calculate the system responses. Therefore, the program structure graph of the serial version reflects the parallel version structure graph of the master program very well. The only difference is that in the parallel version the Jacobian matrix entries are actually computed by the worker programs and stored only by the master program. Furthermore, this approach requires little if none changes to the existing program version.

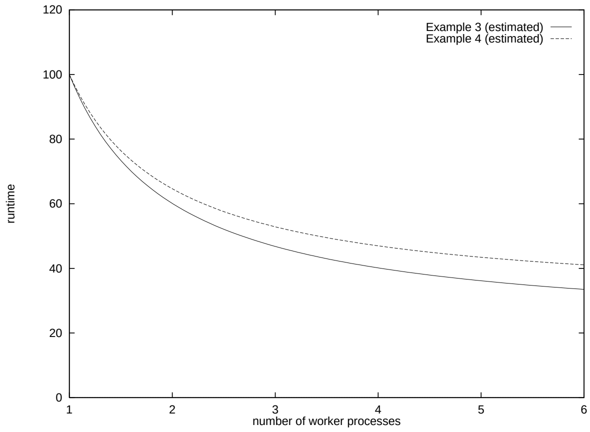

Figure 4: Estimated parallel runtime

5.1 Parallel runtime estimation

The parallelization efficiency of the chosen approach depends on the ratio between parallel and serial fractions of the program and the overhead imposed by communication and control of the parallel processing. To estimate the efficiency a rather simple model is introduced. We assume a fixed problem size. The parallel algorithm is equivalent to the serial algorithm. Therefore, both of the serial and the parallel algorithms perform the same calculations beside the communication. The runtime with p worker processes can be estimated by

$$T_p = t_{ser} + t_{extra} + \frac{1}{p} t_{par}, \tag{33}$$

whereby t_{ser} and t_{par} are the serial and the parallel fraction of the program respectively, and t_{extra} the extra organization and communication work introduced by the parallelization.

Normalizing the total runtime $t_{tot} = t_{ser} + t_{par}$ of the original algorithm, substituting $t_{ser} = 1 - t_{par}$, and assuming an extra time t_{extra} independent from the number of workers involved, Equation (33) is transformed to

$$\begin{aligned} T_p &= 1 - t_{par} + t_{extra} + \frac{1}{p} t_{par} \\ &= 1 + t_{extra} - \frac{p-1}{p} t_{par} \end{aligned} \tag{34}$$

From Equation 33 derives the approximated efficiency of the solution with

$$E_p = \frac{T_0}{pT_p} = \frac{1}{p(t_{ser} + t_{extra}) + t_{par}} \tag{35}$$

and for one worker the efficiency drops immediately below 100% with

$$E_1 = \frac{1}{1 + t_{extra}} \tag{36}$$

This is due to the fact that organizing and communicating with one worker imposes an overhead compared with the in-core solution in the serial version. On the other hand, the master-worker concept applied leaves the master idle most of the time of the computation of the Jacobian matrix elements by the workers. This implies the master and one worker can be executed on the same physical processor in time sharing mode provided by a multitasking operating system like UNIX without any further drop-back in performance. This assumption holds for an increasing number of workers because the master is almost all the time of organizing the workers busy with system tasks (input and output) and does not consume almost no CPU-time. Therefore, the Equation used reflects properly the actual approach resulting in the unfamiliar drop down in the efficiency for the parallel version on one processor. Obviously, we compare our parallel solution with the equivalent serial approach.

On the other hand, analysis of the estimated runtime behavior shows the lack of efficiency for increasing number of workers (compare Figure 4). It is not advised to use more than 4 workers on a homogeneous multiprocessor system with the proposed method. This asymptotic behavior is due to the poor ratio of parallelisable to serial code as reflected in the analysis of the examples shown in Table 1.

5.2 Message-passing implementation

The master-worker concept requires the implementation of exchange routines transporting data between master and worker. In the distributed memory paradigm message-passing primitives transferring copies of memory storing data are feasible. Commonly used implementations are the Message Passing Interface (MPI) [2] standard and the de-facto industry standard Parallel Virtual Machine (PVM) [1]. For comparisons of both message-passing libraries see the detailed description in [3].

Our parallel implementation is based on PVM to meet the following requirements:

Interoperability across a heterogeneous network of workstations This is the main requirement arising from the existing hardware platforms of the prospective end users. This requirement is meet by PVM only, because PVM hides the details of the data transfer and co-operates via the virtual machine interface. In contrast, the protocol for exchanging data between MPI processes is not specified.

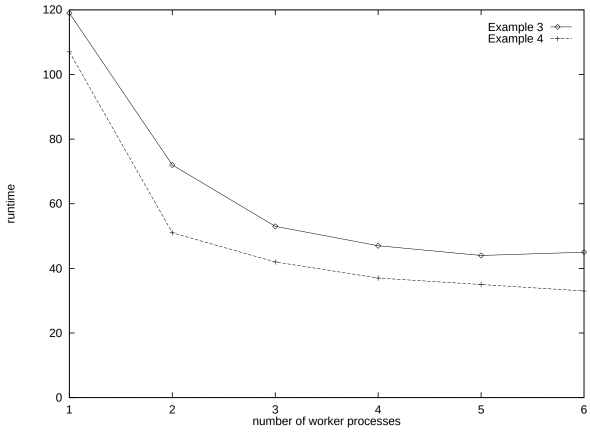

Figure 5: Parallel runtime

Dynamic machine configuration Whereby the previous requirement may be meet by a portable MPI implementation supporting a number of hardware platforms, process invocation and dynamic process control is not part of existing MPI implementations. Although, these features are standardized in the maturing MPI-2 but no implementation supporting this features is known up to now. This feature is essential in the prospective setting within a general purpose network of workstations exploiting free computational resources and handling dynamic load balancing problems.

Fault tolerance Failure of workstations and changing computational load are common observations within production workstation networks but not apparent within commonly used single-user multiprocessor systems. Assuming a constantly running master control process, the virtual machine can be programmed in such a way, that all faults like workstation or program failure, slow network connection, slow computation, can be easily detected and the appropriate reactions can be performed.

Although not essential, all these features of PVM hold for homogeneous multi-processor machines as well. Tests are carried out on these machines to speed-up program development time and the results are documented below.

6 Results

The program implementation and the test studies are carried out on a shared memory Silicon Graphics Multiprocessor Origin2000. The shared memory facilities of the machine are exploited by the internal implementation of the PVM message passing mechanism only. Assignment of processes to particular pro-

Table 2: Parallel runtime for Example 3

Configuration	user [sec]	sys [sec]	total [%]	usage [%]
Serial	10957.6	142.6	100	98
1 Worker	1470.9	242.2	119	12
2 Worker	1510.1	325.3	72	22
3 Worker	1552.0	301.3	53	31
4 Worker	1598.1	504.2	47	39
5 Worker	1642.2	574.7	44	44
6 Worker	1688.4	693.6	45	47

cessors is transparent to the user and managed automatically by the operating system.

We have analyses the runtime for the serial and for the parallel configuration. The information results from the operating system command `time`. A detailed overview of the results is given in Tables 2 and 3. The tables show the user and system time as well as the percentage of elapsed time with respect to the serial run. In the last column the usage of the master process is described by the total CPU time, sum of user and system time, as a percentage of the elapsed time. Therefore, it indicates the justification of the assumption, that it is feasible to place both the master and a worker process on one processor.

More in-depth analysis shows the expected draw back of the proposed solution for small problems. The solution time per direct problem indicating the granularity of the problem is usually below 5 seconds resulting in a significant overhead for the communication.

Table 3: Parallel runtime for Example 4

Configuration	user [sec]	sys [sec]	total [%]	usage [%]
Serial	421.7	2.9	100	98
1 Worker	101.3	3.5	107	22
2 Worker	104.5	5.4	51	37
3 Worker	107.3	6.5	42	47
4 Worker	110.4	8.0	37	55
5 Worker	112.7	7.5	35	60
6 Worker	111.7	7.9	33	63

7 Conclusion

The paper presents the innovative approach to parallelize the time consuming parameter estimation problem for groundwater flow problems by exploiting the existing program structure. For a moderate number of parallel workers runtime reductions up to 50% percent are feasible. The implementation was tested on a homogeneous multiprocessor system. Future work will focus on extensions to a heterogeneous workstation cluster and will require special attention to the load balancing problem.

Acknowledgments

The work presented is part of a project concerned with parallelization of groundwater flow simulation. This project is partially funded by the GKW GmbH, Bochum, Germany and the co-operation is gracefully acknowledged. The main motivation for the work was given by U. Kuhlmann, TK Consult, Zürich, who provided the theoretical framework for the presented parameter estimation strategy.

The first author is graceful to the Alexander von Humboldt-Foundation for the Feodor–Lynen–scholarship to support major part of this work. The authors acknowledge the co-operation and the assistance of the Structural Engineering Computational Technology Group at the Mechanical & Chemical Engineering Department of the Heriot–Watt–University Edinburgh under the leadership of Prof. B.H.V. Topping, especially the access to the high–performance SGI Origin2000 system. Co-operation with this group is funded by the Academic Research Council and the Deutsche Akademische Austauschdienst under the ARC-project scheme.

References

[1] J. Dongarra, A. Geist, R. Manchek, and V. Sunderam. Integrated PVM framework supports heterogeneous network computing. *Computers in Physics*, 7(2):166–175, April 1993.

[2] Message Passing Interface Forum. Document for a standard message-passing interface. Technical Report CS-93-214, University of Tennessee, November 1993. Available on **netlib**.

[3] G.A. Geist, J.A. Kohl, and P.M. Papadopoulos. PVM amd MPI: a comparison of features. *Calculateurs Paralleles*, 8(2):137–150, June 1996.

[4] U. Kuhlmann. *Inverse Modellierung in geklüfteten Grundwasserträgern*. PhD thesis, ETH Zürich, 1992.

[5] D.W. Marquardt. An algorithm for least squares estimation of nonlinear parameters. *SIAM Journal for Applied Mathematics*, 11:431–441, 1963.

[6] G. De Marsily. *Quantitative Hydrogeology*. Academic Press, 1986.

[7] W.W.-G. Yeh. Review of parameter identification procedures in groundwater hydrology: The inverse problem. *Water Resources Research*, 22(2):95–108, 1986.