Hydrogeology Journal
Official Journal of the International Association of Hydrogeologists

10.1007/s10040-003-0252-x

Technical Note

A RAM-economizing domain decomposition technique for regional high-resolution groundwater simulation

W. Gräsle[1] and **W. Kessels**[1]

(1) Leibniz Institute for Applied Geosciences (GGA), Stilleweg 2, 30655 Hannover, Germany

W. Gräsle
Email: w.graesle@gga-hannover.de
Phone: +49-511-6433467
Fax: +49-511-6433665

Received: 19 November 2001 **Accepted:** 21 January 2003 **Published online:** 5 March 2003

Abstract A major limitation of the use of high-resolution groundwater models on a regional scale for resource management by water companies is the excessive RAM requirements of such models which surpass the capacity of today's PCs. A strategy is presented to overcome this problem by using overlapping domain decomposition techniques. Furthermore, because very long computing time is the bottleneck for the practical use of this technique for large groundwater models, an analysis is also presented of a number of methods implemented to increase calculation performance. The approach presented here is characterized by a fairly simple structure that represents a generalized relaxation algorithm. It can be adapted for use with finite element as well as with finite difference methods.

Keywords Groundwater hydraulics - Groundwater management - Numerical modeling - Domain decomposition - Convergence enhancement

---

Resumen La necesidad excesiva de memoria RAM es una de las mayores limitaciones en el uso de modelos de aguas subterráneas de alta resolución a escala regional para las compañías de gestión de recursos hídricos, ya que se sobrepasa la capacidad de las computadoras personales actuales. Se presenta una estrategia para vencer este problema mediante el uso de técnicas de descomposición de dominios superpuestos. Más aún, debido a que los elevados tiempos de cálculo suponen un cuello de botella para la aplicación práctica de esta técnica a modelos extensos, se presenta también un análisis de los métodos implementados para mejorar su rendimiento. El enfoque descrito aquí se caracteriza por una estructura relativamente sencilla que representa un algoritmo generalizado de relajación, que puede ser adaptado a su uso tanto en modelos de elementos finitos como de diferencias finitas.

Résumé Une limitation majeure de la mise en oeuvre de modèles de nappe à haute résolution à l'échelle régionale pour la gestion des ressources en eau par les sociétés d'exploitation de l'eau est la grande gourmandise en mémoire vive de ces modèles, qui dépasse la capacité des micro-ordinateurs actuels. Une stratégie est présentée afin de s'affranchir de ce problème en utilisant des techniques de décomposition de domaine par superposition. De plus, parce que le très long temps de calcul est le problème de fond de l'utilisation pratique de cette technique pour les grands modèles de nappes, une analyse est également présentée d'un certain nombre de méthodes développées pour accroître les performances de calcul. L'approche présentée ici est caractérisée par une structure assez simple qui représente un algorithme généralisé de relaxation. Il peut être adapté à la mise en œuvre des méthodes aux éléments finis, comme de celles aux différences finies.

Introduction

There is an increasing need for reliable instruments for groundwater management against the background of the more intensive use of groundwater resources. Therefore, numerical models of groundwater flow and solute transport become very important in planning sustainable resource utilization and the protection of groundwater quality. In particular, a type of high-resolution regional-scale model with hardware requirements that do not surpass the financial scope of regional water companies is required that can be used in combination with databases and geographical information systems (GIS).

High-resolution models for groundwater flow and solute transport on a regional scale are characterized by an extremely large number of degrees of freedom (typically 10^6 to 10^8). Therefore, they often require more random access memory (RAM) than available on present-day PCs. Domain decomposition techniques provide a possible way of reducing the RAM requirements of such models. These techniques are widely used for solving a great variety of model problems on parallel supercomputers (e.g. Dou and Phan-Thien *1998*; Byrde et al. *1999*; Farhat *2000*). In contrast, the use of domain decomposition methods to reduce RAM requirements on smaller computers plays a minor role (e.g. Beckie et al. *1993*).

MovingWindow – An Overlapping Domain Decomposition Algorithm

MovingWindow is mainly conceived for high-resolution models for groundwater flow and solute transport on a regional scale and for use in combination with geographical databases. Since geographical data are mostly stored on a regular grid, a program specialized in structured grids, and thereby taking advantage of this relatively simple situation, is a good choice. The primary objective is versatile applicability for the usual tasks of water companies but not the optimization of flexibility and performance for highly specialized problems with complex model geometries requiring unstructured meshes. Although MovingWindow is essentially compatible with finite element and finite difference (FD) models, the main field of application will therefore be with FD models.

The use of domain decomposition techniques to reduce RAM requirements differs from the use with parallel computers using a linear calculation sequence of the subdomains. There are no parallel processes that produce results that cannot be used in the other calculations. Therefore, even overlapping domain decomposition algorithms are suitable for this purpose. In parallel computing, overlapping algorithms are less suitable because they cause undesirable restrictions, e.g. the avoidance of the simultaneous calculation of overlapping subdomains. Nevertheless, MovingWindow can be used for parallel computations when taking these restrictions into consideration.

MovingWindow is an overlapping domain decomposition method where the model region is completely covered by a number of approximately equal-sized subdomains ("windows") overlapping each other. The solution of a steady-state boundary value problem or an unsteady initial/boundary condition problem is realized by iteratively solving the problem on the set of windows as shown in Fig. 1. The overlapping algorithm enables a simple and straightforward treatment of the boundary conditions on the generated window boundaries inside the original model domain. The values of status variables resulting from previous calculations of the overlapping neighboring windows are used as Dirichlet-type boundary conditions for the actual window.

Therefore, MovingWindow can be interpreted as a type of generalized relaxation solver. It is in fact identical with the normal relaxation method for a minimum window size of three nodes in any grid direction (resulting in windows with only a single inner node). The convergence behavior is similar to that of a relaxation solver asymptotically showing an exponential decrease in the residual with an increasing number of iterations. Numerical tests comparing calculations with and without domain decomposition confirm that for hydraulic problems, MovingWindow does not cause any numerical errors—with the exception of termination errors. The latter are unavoidable for any iterative method but can be minimized to any desired level by increasing the number of iterations.

The reduction in RAM requirements is mainly attributable to the fact that global matrices are neither generated nor stored in RAM for the complete model region, but only for one subdomain (window) at a time. The program offers an option to reduce RAM requirements even further: the values of status variables not required for the window being calculated can be stored in a file instead of keeping them in RAM. Of course this inevitably causes a substantial deterioration in program performance. Because the RAM requirements for storing the status variables are small compared to storing global matrices, this time-consuming type of RAM saving is only recommended for extremely large models that cannot be calculated by any other means or that already enforce automatic swapping.

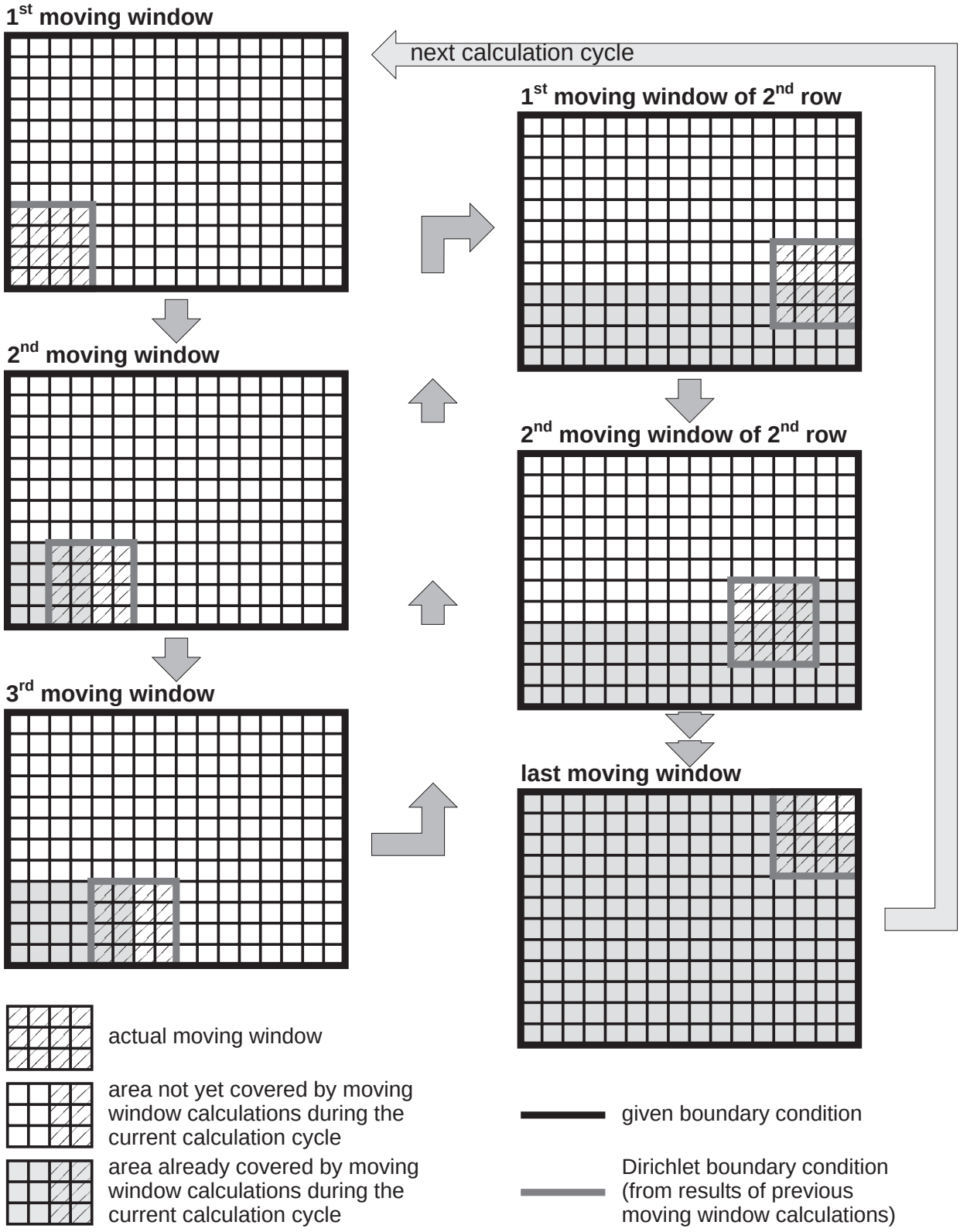

Fig. 1. The principle of MovingWindow: during a calculation cycle, the window is moved over the complete model area while overlapping with neighboring windows. It takes several calculation cycles to approximate the model solution

A promising field of application for MovingWindow is overcoming the common problem of an insufficient knowledge of hydraulic boundary conditions. The RAM-saving capacities of MovingWindow enable model boundaries to be chosen in such a way that the hydraulic boundary conditions are relatively well determined (e.g. no-flow conditions on catchment boundaries) or that the uncertainties of boundary conditions are located far enough from the region of interest to sufficiently reduce their influence. In particular, this is possible for steady-state hydraulic problems that have comparatively moderate calculation time requirements. In many cases, the results of large-scale steady-state calculations can provide a reasonable approximation for the boundary conditions in models of small-scale unsteady or transport problems.

Enhancement of Convergence Speed

In addition to the problem of huge RAM requirements, very long calculation times are a critical problem associated with high-resolution regional groundwater models. Therefore, several methods to enhance convergence speed were implemented and tested. To be able to test MovingWindow and the implemented convergence accelerators with calculations without domain decomposition, the analysis was performed using a simplified steady-state two-dimensional hydraulic problem. Also, notwithstanding its poor performance, a RAM-saving relaxation solver was used for the calculations to facilitate the solution on a PC (600 MHz, 256 MB), even without domain decomposition, of a model of the "coastal aquifer test field" (Fulda et al. *2000*; Kessels et al. *2001*) between Bremerhaven and Cuxhaven (northern Germany) which consists of about 7.5×10^5 nodes. To preserve comparability, the MovingWindow calculations had to use the same relaxation solver inside the windows, although this is not necessary for RAM-saving reasons. Naturally, the three-dimensional version of MovingWindow does not use a relaxation solver but utilizes a BCG (bi-conjugate gradient) solver instead.

To test the convergence acceleration techniques, an adequate number of calculations were carried out on the convergence acceleration methods to analyze the dependency of calculation time on the optimization parameters and on the achieved calculation accuracy.

In total, five methods to optimize the calculation time requirements were tested:

1. Overrelaxation, characterized by the overrelaxation factor (ORF).
2. Optimization of calculation sequence (CS).
3. Hierarchical grid coarsening (multigrid technique), characterized by the grid coarsening factor (GCF) and the number of grid coarsening levels (GCL).
4. Optimization of window size, characterized by the number of nodes per window (NPW).
5. Stepwise improvement of the convergence criteria, characterized by the accuracy improvement factor (AIF).

Overrelaxation

Overrelaxation is a technique specific for the relaxation solver used. This common method results in a strong reduction of calculation time up to approximately one order of magnitude. Figure 2 illustrates that for the investigated model problem, a pronounced optimum was found for overrelaxation factors (ORF) slightly above 1.95. A steep increase in calculation time occurs for ORF>ORF_{opt} (for ORF >2 the iteration does not converge at all), whereas the slope of the t_{calc}(ORF)-relationship is comparatively moderate for ORF<ORF_{opt}. Since the optimum value ORF_{opt} depends on the model problem and cannot be predicted exactly, an overrelaxation factor between 1.8 and 1.95 –which most probably is a slight underestimation of ORF_{opt} – should be a conservative and reasonable guess for this optimization parameter.

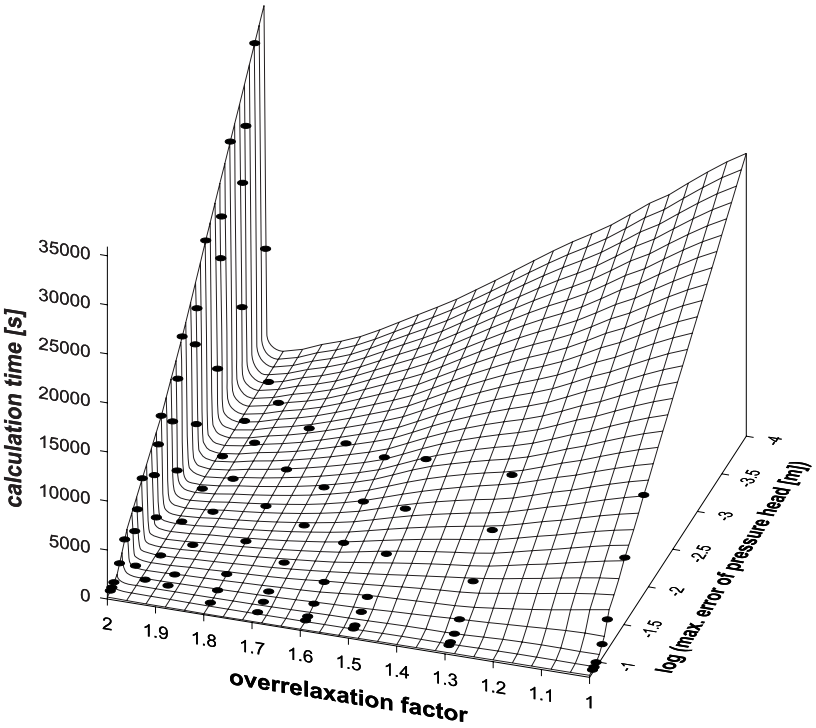

Fig. 2. Influence of overrelaxation on calculation time. Fixed calculation parameters: CS=regular, GCF=4, GCL=2, NPW=10,197, AIF=0.316

Optimization of the Calculation Sequence

Obviously, the sequence of the MovingWindow calculations will influence the convergence speed of this method. The most straightforward approach termed "regular sequence" is to move the window over the model domain in rows or columns with alternating directions similar to the traditional IADI (iterative alternating direction implicit) scheme (Peaceman and Rachford *1955*). This rigid scheme does not consider that the iterative process converges much faster in relatively homogeneous parts of the model region compared to parts showing a greater heterogeneity on the window scale with respect to material properties or boundary conditions.

The idea of calculation sequence optimization is to concentrate the calculation effort on those parts of the model domain where it is really needed, i.e. where the state of the calculation is far from fulfilling the chosen convergence criteria. As illustrated in Fig. 3, the optimization concept implemented by MovingWindow always chooses that window for the next calculation which

shows the largest change in Dirichlet boundary conditions since the preceding calculation. In doing so the size of the boundary condition change is always considered relative to the chosen convergence criteria. The additional RAM requirements of this technique are small because it only requires two-fold data storage of the window boundaries. The calculation time for administrating the changes in boundary conditions on the window boundaries and evaluating the decision criterion is negligible as long as the selected window size is not too small. To initiate this method, all windows have to be calculated once in a regular sequence before changing to the optimized process.

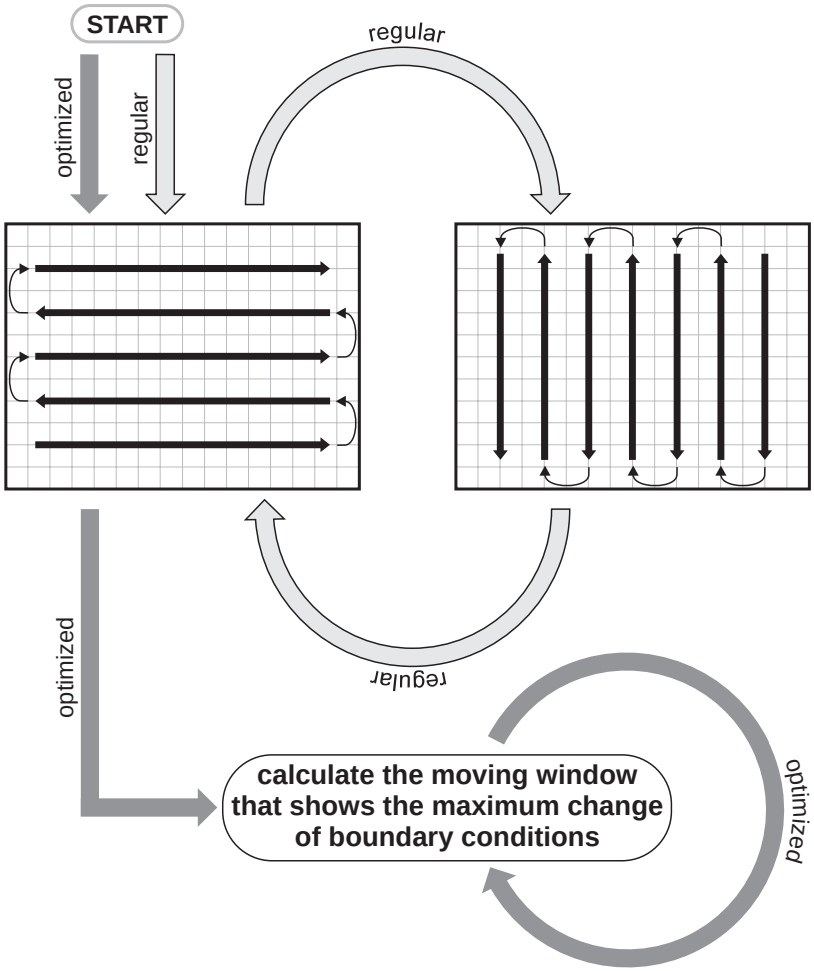

Fig. 3. Comparison of regular and optimized calculation sequences

A comparison of Figs. 2 and 4 shows the reduction in calculation time gained by the optimization of the calculation sequence. Convergence speed is typically increased by 50 to 100%.

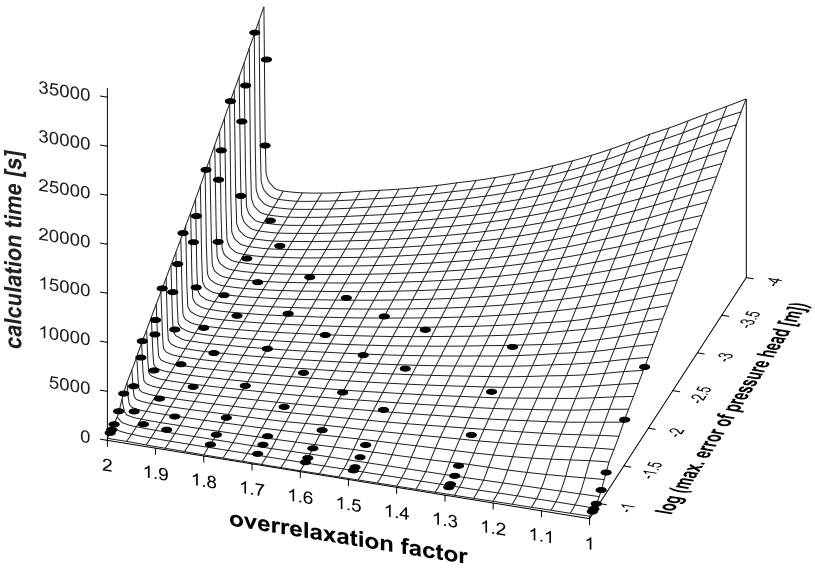

Fig. 4. Influence of overrelaxation on calculation time. Fixed calculation parameters: CS=optimized, GCF=4, GCL=2, NPW=10,197, AIF=0.316. These calculations only differ from those displayed in Fig. 2 by the optimized calculation sequence

Hierarchical Grid Coarsening

Long-wave errors (this means wavelengths significantly longer than the typical window dimension) in the current approximation of the solution decline relatively slowly in the iterative MovingWindow procedure. This also reflects the close relationship between relaxation solvers and the MovingWindow domain decomposition technique because relaxation solvers perform very badly with respect to eliminating smooth long-wave errors (Stephen and McCormick *1987*). Typical approaches to overcoming this problem are multigrid methods, of which an enormous variety were developed during the last decades (e.g. Kornhuber *1997*; Hackbusch *1998*; LeBorne *1999*).

The technique used here is a rather simple nested iteration type approach that can be described as a hierarchical coarsening of the grid. The MovingWindow calculation on the original high-resolution grid is carried out after a calculation on a coarsened grid where a chosen number of original nodes per grid direction (the grid coarsening factor, GCF) are represented by a single node of the coarsened grid. This procedure can be nested hierarchically in several grid coarsening levels.

The MovingWindow principle is applied to any level of grid coarsening. In doing so the number of nodes per window is kept approximately independent of the grid coarsening level. This generates windows with strongly increased linear dimensions for higher grid coarsening levels. The benefit of this technique can also be described in this way: before a particular grid spacing is used, all error components with wavelengths that are too long to be economically eliminated with this grid spacing are eliminated in advance by preliminary comparatively fast calculations on coarser grids. This allows the number of iterations for the time-consuming calculations on finer grids to be significantly reduced. This technique is well suited for more or less homogeneous or smooth distributions of material properties but is not very efficient when there are many pronounced material property contrasts within the model.

In the simple model of the coastal aquifer test field, there was no benefit in employing more than two levels of grid coarsening. It was revealed that the dependency of calculation time on the GCF

shows a typical optimum characteristic, as shown in Fig. 5, with the position of the optimum (GCF_{opt}=4) nearly independent of the achieved accuracy.

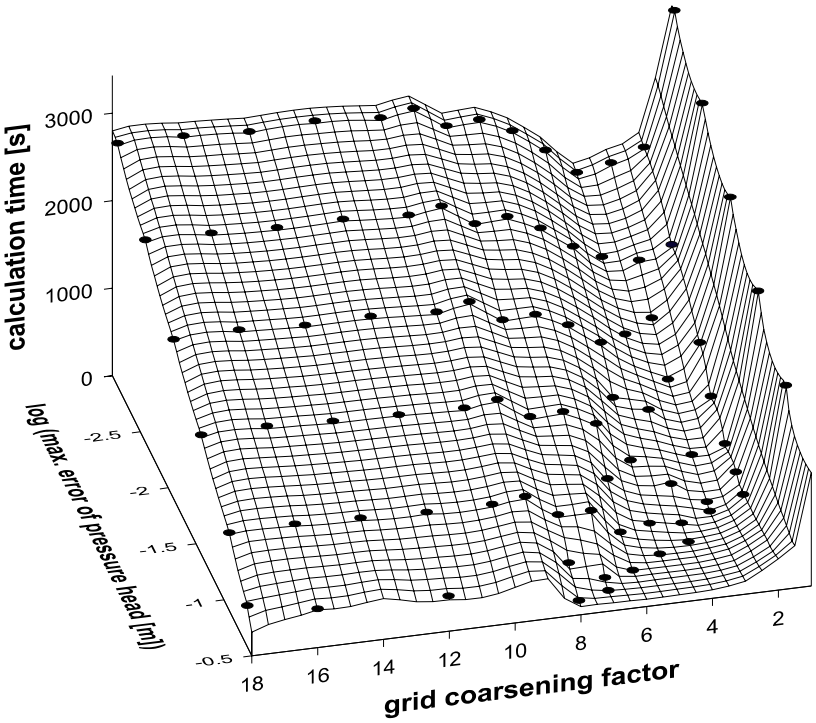

Fig. 5. Influence of hierarchical grid coarsening on calculation time. A grid coarsening factor of 1 denotes that the multigrid technique was not used. Fixed calculation parameters: ORF=1.8, CS=optimized, GCL=2, NPW=10,197, AIF=0.316

If the selected GCF>GCF_{opt} is too high, some error components of intermediate wavelength cannot be adequately reduced by a coarse grid calculation. However, the same error components have wavelengths that are too long to be efficiently eliminated by the finer grid calculation. As a result, there is an increase in the number of fine grid iterations and associated calculation time. For optimum parameter values, the calculation time could be reduced by a factor of 2.5 compared to calculations without multigrid algorithms.

Optimization of Window Size

The size of the windows will strongly affect the calculation time necessary for the solution of a model problem. On the other hand, the window size has to meet the demands of RAM saving. Therefore, the question of achieving a good compromise between calculation performance and a reduction in RAM requirements is very important with respect to window size.

Because the use of a domain decomposition technique like MovingWindow increases the complexity of a problem, one might expect that a calculation without domain decomposition (which is equivalent to a window size equal to the size of the complete model) will be the fastest approach. Nevertheless, the tests revealed a different type of behavior: according to Fig. 6, calculation time shows a broad minimum for an intermediate window size and increases three to four times for very small or very large windows. This means that MovingWindow not only is a tool to reduce the RAM requirements of very large models, but also can be used as a means of convergence enhancement. It is the advantage of a window size significantly smaller than the model domain that all

parts of the model where the calculation converges quickly can be omitted from subsequent computations within the iterative process. On the other hand, when the selected window size is too small, the increasing expense for the administration of window boundary conditions and calculation sequence, as well as the restriction of efficient convergence to very short-wave errors, overcompensate this benefit.

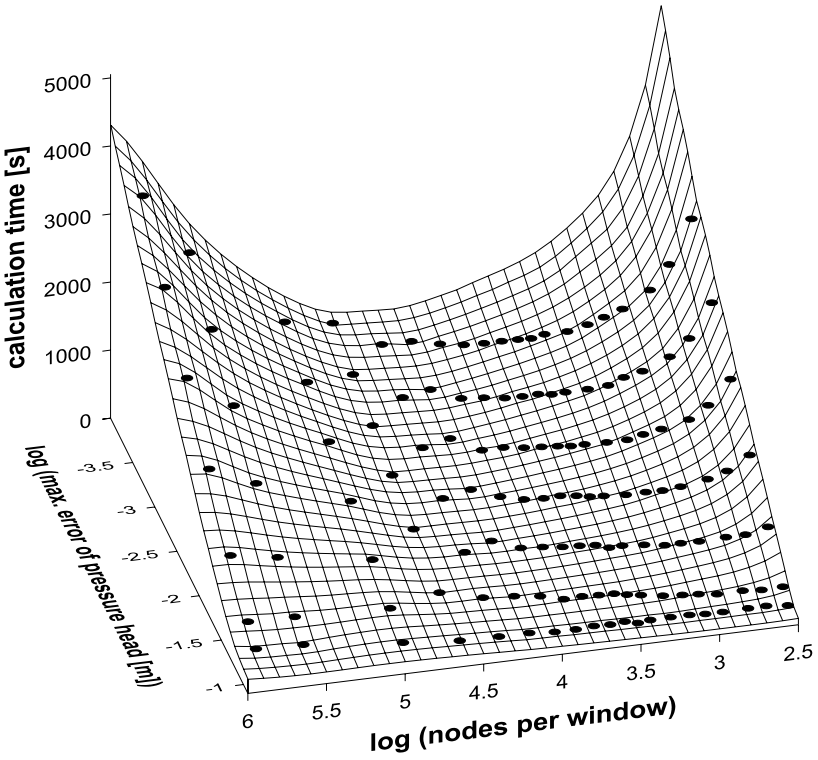

Fig. 6. Influence of window size on calculation time. The maximum window size used is 745,841 nodes and represents a calculation without domain decomposition. Fixed calculation parameters: ORF=1.95, CS=optimized, GCF=4, GCL=2, AIF=0.316. Note the logarithmic scaling of the window size axis

It is important to point out that the influence of window size on calculation time strongly depends on the equation solver used inside the windows because each solver has a different time requirement versus window size characteristic. Furthermore, the hardware limitations and settings of the operating system can have a large effect on the analyzed behavior because the need to use virtual memory by swapping will reduce calculation speed significantly when the window size exceeds a machine-dependent upper limit. The smooth shape of the surface shown in Fig. 6 proves that no swapping occurs in the investigated example for any tested window size.

Stepwise Improvement of the Convergence Criteria

Because the use of direct solvers is inappropriate for larger model problems—even for linear problems—two iterative processes will interact in virtually any MovingWindow application: namely, the iterative equation solver inside any window and the MovingWindow procedure itself. In this case, calculation time requirements not only depend on the intended accuracy of the solution which corresponds to the chosen final convergence criteria, but also are affected by the way the convergence criteria for both iterative processes are administered.

It is not surprising that the simple approach that uses the final convergence criteria right from the beginning is rather inefficient. In this case, time is wasted in achieving the finally desired accuracy inside the windows at an early stage of the MovingWindow process when the boundary conditions of the windows are still very inaccurate. Therefore, it is useful to improve the convergence criteria inside the windows gradually during the progress of the MovingWindow procedure. Directly coupling the convergence criteria inside the windows to the absolute or relative changes actually occurring in a MovingWindow step would lead to an over-sensitive response because, even in an optimized calculation sequence, it is possible that some calculated windows will show rather small changes of status variables followed by window calculations generating more pronounced changes. If this is the case, an over-fast response by the convergence criteria adaptation procedure to the small changes of status in some windows would lead to an unnecessary improvement in convergence criteria for the subsequent steps, and thereby cause an increase in calculation time. Therefore, a somewhat retarded response of the convergence criteria adaptation inside the windows to the behavior of the MovingWindow calculation is required. A natural way of doing this is to link the actual convergence criteria inside the windows to the maximum changes of status variables that occurred in the last complete MovingWindow cycle for a regular calculation sequence, or in a particular number of calculated windows (e.g. equal to the total number of windows in the model) of an optimized calculation sequence.

To provide a means of easily controlling this adaptation process, the implemented procedure uses an "accuracy improvement factor" (AIF) ($0 \leq AIF \leq 1$) and allows the improvement of convergence criteria inside the windows only if the status changes in the preceding MovingWindow cycle have at least decreased by this factor. An AIF=0 denotes no stepwise improvement of convergence criteria (use of the final criteria right from the start); the other extreme, AIF=1, characterizes an over-sensitive adaptation. Figure 7 shows an optimum for an AIF of about 0.3. The observed benefit of this method is a reduction in calculation time by approximately half.

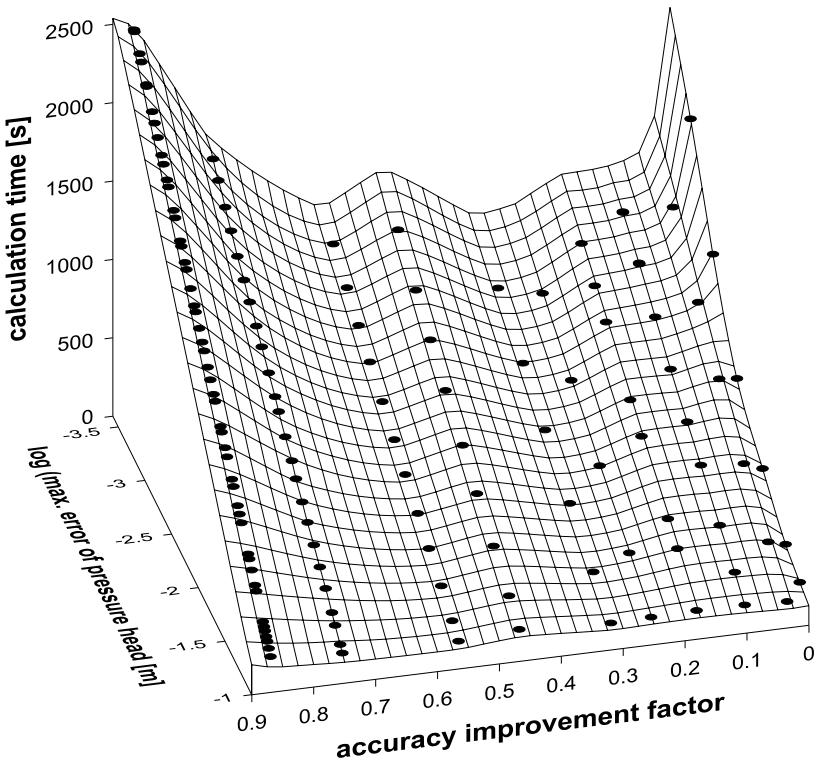

Fig. 7. Influence on the calculation time of stepwise improvement of the convergence criteria. Fixed calculation parameters: ORF=1.95, CS=optimized, GCF=4, GCL=2, NPW=10,197

Overview of Methods for Convergence Improvement

All methods discussed here proved to be useful for an acceleration of convergence speed. With the exception of overrelaxation, all techniques can be employed in any MovingWindow application. An overview of their efficiency and the optimum parameter values found for the analyzed two-dimensional flow model of the coastal aquifer test field is given in Table 1. The efficiency of any technique depends on the particular model problem and on the settings of the other optimization parameters. For the investigated example, the combination of all convergence acceleration methods could reduce the calculation time requirements by approximately two orders of magnitude.

Table 1. Comparison of the tested techniques for reduction of calculation time of MovingWindow applications. Acceleration factors and optimum parameter values are given for the two-dimensional flow model of the coastal aquifer test field

Method	Maximum acceleration factor	Approximate optimum parameter values	
Overrelaxation	11.3	Overrelaxation factor	1.95
Optimization of calculation sequence	2.0	Optimized sequence	
Hierarchical grid coarsening	2.5	Grid coarsening factor	4
		Grid coarsening levels	2
Optimization of window size	3.8	Nodes per window	12,000
Stepwise improvement of convergence criteria	2.0	Accuracy improvement factor	0.3

Summary

The overlapping domain decomposition algorithm MovingWindow was found to be an efficient approach to large-scale groundwater modeling with structured grids on PCs by reducing the RAM requirements of such models to a desired quantity. It represents a type of generalized relaxation method and shows a similarly robust convergence behavior, causing no other additional numerical errors with the exception of termination errors.

Because excessively long calculation times are the second most important limitation (in addition to excessive RAM requirements) for high-resolution regional-scale groundwater models, a number of convergence acceleration techniques were implemented and tested. Whereas overrelaxation is a technique specific for the relaxation solver used with the tests, the other four methods—optimization of the calculation sequence, a nested iteration type hierarchical grid coarsening, optimized window sizes, and the stepwise improvement of the convergence criteria—can be used in combination with virtually any equation solver inside the windows. All these techniques are fully mutually compatible but may strongly interact with respect to their acceleration efficiency. The cumulative benefit of the investigated convergence acceleration techniques achieves a reduction of calculation time of approximately two orders of magnitude. Because the optimum window size was found to be much smaller than the dimension of the complete model, it appears that performance optimization by selecting an appropriate window size does not interfere with the restrictions of window size arising from RAM saving.

References

Beckie R, Wood EF, Aldama AA (1993) Mixed finite element simulation of saturated groundwater flow using a multigrid accelerated domain decomposition technique. Water Resour Res 29:3145-3157

Byrde O, Couzy W, Deville MO, Sawley ML (1999) High-performance parallel computing for incompressible flow simulations. Comput Mech 23:98–107

Dou HS, Phan-Thien N (1998) On the scalability of parallel computations on a network of workstations. Comput Mech 22:344–354

Farhat C (ed) (2000) Vistas in domain decomposition and parallel processing in computational mechanics. Computer methods in applied mechanics and engineering. 184,2/4 Spec Issue. Elsevier, Amsterdam

Fulda C, Gräsle W, Kessels W, Willert T, Zoth G (2000) Regional modeling of flow and transport in a coastal aquifer in northern Germany. In: Bjerg PL, Engesgaard P, Krom TD (eds) Proc Conf Groundwater 2000, Copenhagen, AA Balkema, Rotterdam, pp 85-86

Hackbusch W (ed) (1998) Multigrid methods V. In: Proc 5th European Multigrid Conf, Stuttgart, 1–4 Oct1996, Springer, Berlin Heidelberg New York

Kessels W, Fulda C, Binot F, Dörhöfer G, Fritz J (2001) Monitoring and modeling in the coastal aquifer test field (CAT-Field) between Bremerhaven and Cuxhaven in the northern part of Germany. In: Proc Conf Salt Water Intrusion and Coastal Aquifers—Monitoring, Modeling and Management (SWICA-M^3), 23-25 April, Essaouira, Morocco

Kornhuber R (1997) Globally convergent multigrid methods for porous medium type problems. Preprint from the Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), SC 97-45, ftp://ftp.zib.de/pub/zib-

publications/reports/SC-97-45.ps

LeBorne S (1999) Multigrid methods for convection dominated problems. PhD Thesis, University of Kiel, Germany, http://www.dissertation.de/PDF/sl90.pdf

Peaceman DW, Rachford HH (1955) The numerical solution of parabolic and elliptical differential equations. J Soc. Ind Appl Math 3:28–41

Stephen F, McCormick S (eds) (1987) Multigrid methods. Society for Industrial and Applied Mathematics, Philadelphia, 282 pp