Implementing Parallelism into an Unsaturated Soil Zone Simulation Model

Martin Meyer[1], Jana Sallwey[2], René Blankenburg[3], Peter-Wolfgang Graeber[4],
[1, 2, 4] *Technische Universitaet Dresden, Institute of Waste Management and Contaminated Site Treatment,*
[3] *IBGW® Leipzig*

Abstract **- PCSiWaPro® is a software tool developed by the TU Dresden which simulates water flow and contaminant transport in the unsaturated soil zone. Due to the increasing demand in computational efficiency for large numerical meshes, simulation runs needed to be accelerated. This acceleration was carried out by manual parallelization of parts of the software's simulation kernel. After identifying the most time-consuming sections of the code several numerical equation solver libraries were implemented and tested. Different concepts of software parallelization as well as different types of mathematical solving strategies were tested for their suitability for PCSiWaPro®. With the SAMG software, a multigrid solver developed by the Fraunhofer Institute for Algorithms and Scientific Computing, the largest reduction of simulation time could be achieved.**

Keywords **- SAMG, parallelization, simulation, unsaturated soil**

I. Introduction

Models of the unsaturated soil zones have an increasing importance for the simulation of environmental issues. Their transient characteristics can help determine processes that influence the behavior of groundwater bodies and plants with special regard to climate change. Recent developments introduce more exact and physically reasonable features into the models. Another trend adds the prediction of climate change with the help of synthetic time series generation. All of these features increase the applicability of the model but also enlarge the simulation time. Model codes grow and become more complex and the runtime of the model increases with every new feature.

In order to enable reasonable runtimes for the new and complex models, the hardware of the underlying computer system as well as the codes of the software can be accelerated by parallelization. Parallelism is the computing technique of simultaneously running different calculations on one or more machines. Although this approach has already been implemented into the design of new processing units, programmers can additionally introduce parallelism into their software codes. This gives them more control over the complete calculation process and enables to them to apply computation techniques that are not or only hardly possible to implement in sequential programming. Although there are many codes that are not easily parallelizable, it still holds great potential in further accelerating software code.

Parallelism not only helps accelerate more complex models but it also enables the use of models to simulate larger model areas. In case of the unsaturated soil zone an average model width will be several meters. More accurate models can have sizes of up to 1 km, which significantly increases the number of nodes in the finite element mesh, and therefore the size of the system of linear equations to be solved. With the help of parallelism simulations of this size can be solved more effectively by distributing computations to individual computing units.

II. Modeling Unsaturated Soil Zone Processes

Water flow through variably saturated porous media can be modeled by the Richards equation (1) in the following form:

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial x_i}\left[K\left(K_{ij}^{A}\frac{\partial h}{\partial x_j} + K_{iz}^{A}\right)\right] - S \qquad (1)$$

with volumetric water content θ, spatial coordinates x_i (x,z), capillary pressure head h, time t, K_{ij}^{A} and K_{iz}^{A} as components of an anisotropy tensor K, unsaturated hydraulic conductivity K, and a source/sink term S. Since both the hydraulic conductivity and the dependent variable θ are expressed as functions of the pressure head h, the Richards equation becomes a highly non-linear partial differential equation which needs to be solved iteratively.

The interdependence of the water content θ and the pressure head h are expressed by the Van Genuchten / Luckner model (2) [1]

$$\theta = A + \frac{\phi - A - B}{\left[1 + \left(\alpha \cdot p_{c,w-nw}\right)^{n}\right]^{1-1/n}} \qquad (2)$$

with the function of the residual water content A, the function of the residual air content B, soil porosity φ, and the difference between the pressure heads of the wetting and the non-wetting phase $p_{c,w-nw}$. The Van Genuchten parameters α and n are empirical.

Due to the heterogeneity of the soil pore size distribution, the processes of drainage and wetting are subject to hysteresis. This hysteresis can be expressed by defining different α's for the two process directions. As can be seen in Fig. 1, depending on the process direction, the state functions of the soil develop along different paths.

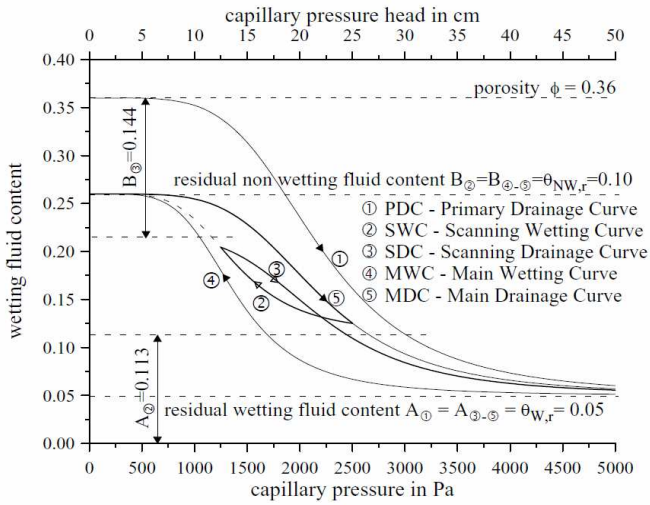

Fig. 1. Hysteresis of soil state functions

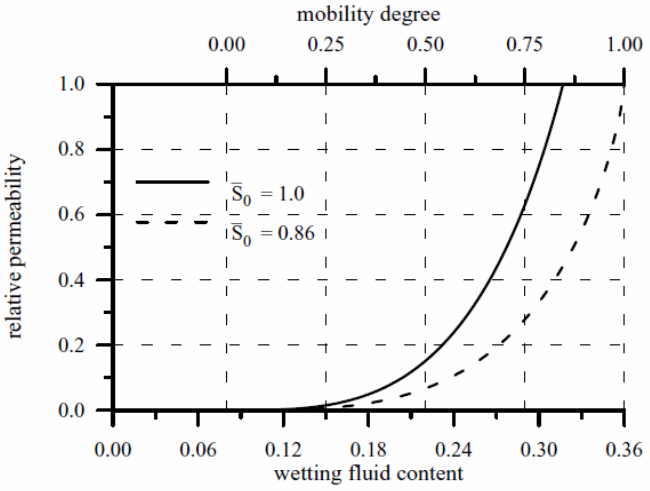

Fig. 2. Unsaturated conductivity function

In contrast to groundwater flow processes, in the unsaturated soil zone the hydraulic conductivity is not constant, but it varies depending on the pressure head. The following formula (3) is taken from [1]

$$k(\theta) = k_0 \cdot \left(\frac{S}{S_0}\right)^{\lambda} \cdot \left[\frac{1-\left(1-S^{\frac{1}{m}}\right)^{m}}{1-\left(1-S_0^{\frac{1}{m}}\right)^{m}}\right]^{2} \qquad (3)$$

with hydraulic conductivity $k_0(\theta_0)$ at a defined point 0 with a known degree of wetting fluid mobility $S_0 = (\theta_0 - \theta_{w,r})/(\varphi - \theta_{w,r})$, a connectivity parameter λ and a transformation parameter m. A typical course of $k(\theta)$ is shown in Fig. 2.

III. The Software PCSIWAPRO®

In the course of the BMBF-funded project "Sickerwasserprognose" ("leachate prognosis") the Institute of Waste Management and Contaminated Site Treatment together with the Ingenieurgesellschaft für Wasser und Boden mbH developed the simulation software SiWaPro DSS [2].

The latest version of this software, PCSiWaPro®, implements the numerical solution of the Richards equation (1) in two spatial dimensions using a finite element approach.

The linear system of equations resulting from the discretized Richards equation is solved using the conjugate gradient method with an incomplete lower-upper-decomposition as preconditioner [3]. Due to the highly nonlinear character of (1), a sequence of linear systems of equations has to be solved for each simulation time step until the difference between two successive solution falls below a predefined limit. The number of such iterations increases with the parametric complexity of the relationship between the soil state functions. Furthermore, in many cases very small time step sizes have to be used, for example when dealing with coarse-grained soil layers. These two factors can lead to very large simulation runtimes, especially when very fine spatial distributions are included in the model.

A graphical user interface was developed to help the user create and manage models, create model geometries and input soil parameters and initial and boundary conditions (Fig. 3).

Software parallelization includes the partitioning of parts of the software's code into several individual execution streams which run in parallel. This is only applicable for parts which do not include internal data dependencies. While there are already several possibilities for automatically parallelizing software codes on the compiler or the hardware level (integrated compiler optimizations, processor vectorization), great potential in terms of runtime efficiency lies in "manual" parallelization by extending or adapting an existing sequential code.

In the field of non-GPU-based software parallelization, two basic concepts are widely used: OpenMP [4] and MPI [5].

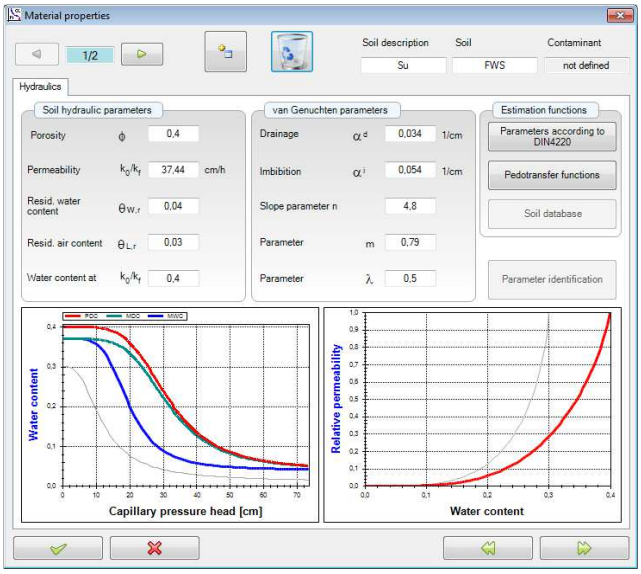

Fig. 3. Graphical user interface of PCSiWaPro®

3.1. Open Multi-Processing (OpenMP)

OpenMP parallelization is based on the fork-join-model (Fig. 4). In this model iterations of a loop are distributed from the previous one to an arbitrary number of threads. The threads execute their assigned operations in parallel and finally rejoin at the end of the loop to a single master thread. Implementation of this code partitioning is carried out solely by compiler directives. This method does not only limit the necessary alterations of the original sequential loop code, but also guarantees that the loop can still be executed if parallelization is not available on a particular machine. Another advantage is that by using OpenMP a code can be parallelized loop by loop, so that with each new parallelization an additional gain in execution speed can be achieved.

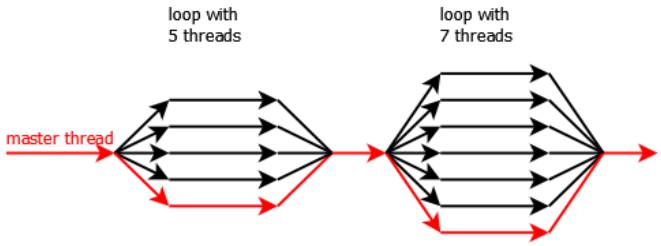

Figure 4. Fork-join-model in OpenMP

The OpenMP execution model works best on shared memory systems, such as machines with several individual cores per processor. Ideally each core can run the operations of one software thread which leads to a measure for the maximum number of threads to be used in parallelizing a code on a given machine. The best runtimes for a code can be achieved if the operations of the loop are distributed to the threads in such a way that all threads or processor cores have the same number of operations to process. For this purpose OpenMP offers a set of specialized directives.

The usage of OpenMP in PCSiWaPro® is especially relevant for loops which iterate over the nodes or elements of the discretized model, and in which single iterations are independent of one another.

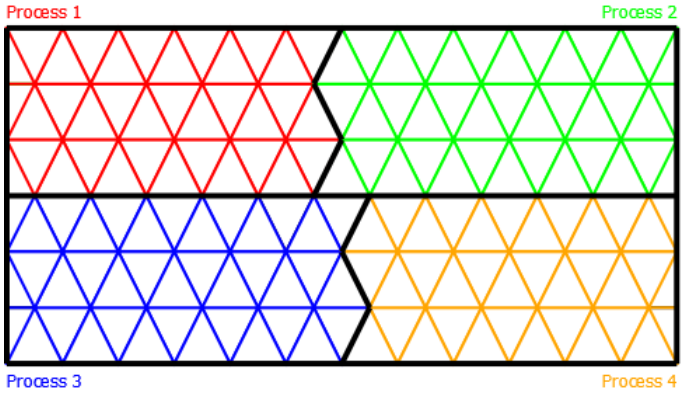

Fig. 5. Idealized distribution of a finite element mesh to four MPI processes

3.2. Message Passing Interface (MPI)

Another concept for software parallelization is MPI. In comparison to OpenMP MPI introduces partitioning in both work and data. MPI-parallelized software is executed with a given number of processes. Each of these processes works independent of the others with its own data and operations. At the startup of a program all data of the task to be parallelized have to be distributed to the processes, after which each process operates on its own dataset and with its own set of operations. In the application scenario of numerically solving the discretized Richards equation two ways of data partitioning are possible:

(a) partitioning of the resulting linear system of equations by rows or columns

(b) partitioning of the finite element mesh of the discretized geometrical model by nodes or elements (Fig. 5)

At the borders of mesh parts that belong to different processes complications can arise, since in these regions data exchange between two generally independent processes has to be implemented. This is carried out by using MPI messages for process intercommunication. Synchronization of the parallel processes as well as process intercommunication has to be implemented manually by the user with the help of special MPI commands. In comparison to OpenMP MPI parallelization is much more expensive. In most cases a reimplementation of many parts of the software code is inevitable.

MPI parallelization produces best results on distributed memory systems, such as massively parallel computer clusters. As with OpenMP, a uniform load balancing between all involved computing components has to be ensured. Since the above-mentioned systems can consist of thousands of individual components, this is rather complicated. Additionally, it is a more influential factor than for OpenMP, since the possible idling of processes as a result of bad load balancing can add up to a much bigger amount of "wasted" computing time.

The speed gain from MPI parallelization can be estimated using Amdahl's law (4). It states that the speedup S of a parallelized code depends on the number of processes p and on the fraction s of the code which was not parallelized.

According to (4), a completely parallelized code using p processes would lead to a runtime p-times faster than the sequential version. On the other hand, using an infinite number of processes would not lead to an infinite speedup, since S is always limited by 1/s.

3.3. Implementation and Results

In order to speed up simulation runs in PCSiWaPro® the most expensive code parts were determined at first by individual runtime measurements. It was detected that the assembly of the matrix of the discretized Richards system of equations together with the numerical solution of the system could take up to 80% of the total simulation runtime.

The assembly of the Richards matrix is implemented by a number of loops which iterate over the elements and nodes of the model area and successively generate matrix entries. These

loops could partially be parallelized with OpenMP. However, because of their inner structure internal synchronizations between the threads had to be implemented. Nevertheless, some speedup could already be accomplished.

PCSiWaPro®'s existing numerical solving method could not be parallelized, especially the integrated LU decomposition as a matrix preconditioner proved to be problematic. For this reason different parallel solver libraries were implemented into the software's simulation kernel in order to replace the original solver, and to test their functionality, scalability and possible speedups.

Solver libraries based on MPI were not able to reduce simulation runtimes in PCSiWaPro®. Tests were carried out with the freely available libraries PETSc [6] and Lis [7]. There are several reasons that induce this behavior. MPI parallelization scales best for software without a graphical user interface run on massively parallel computing clusters solving systems of hundred thousands of equations. In contrast to that, PCSiWaPro® is software developed for shared-memory systems, whose field of application is rather an engineering office than a scientific high-performance computer. Furthermore, due to internal memory handling the software can only simulate finite element meshes with an upper limit of about 150,000 nodes. For these reasons, only OpenMP parallelization was used to accelerate internal operations in PCSiWaPro®.

iterative method	**direct method**
$r = A \cdot \mathbf{x} - B$	$A = L \cdot U$
precon(A)	$A \cdot \mathbf{x} = L \cdot U \cdot \mathbf{x} = B$
while(norm(r)>tolerance)	$U \cdot \mathbf{x} = y$
{ calculate(x)	$L \cdot (U \cdot \mathbf{x}) = L \cdot y = B \rightarrow y$
$r = A \cdot \mathbf{x} - B$ }	$U \cdot \mathbf{x} = y \rightarrow \mathbf{x}$

Fig. 6. Mathematical principles of iterative and direct linear systems solving methods for $A \cdot x = B$

In addition to the type of parallelization, the available libraries can also be divided into direct and iterative solvers. Figure 6 shows how these two techniques differ in their mathematical approaches to solve a linear system of equations. For testing purposes the PARDISO [8] and the MUMPS [9] libraries were implemented into PCSiWaPro® simulation kernel. Time measurements on the individual routines of the solvers showed that the process of factorizing the system matrix A takes a vast amount of time and memory. This could not be significantly decreased by the available solver options or measures carried out in the calling PCSiWaPro® code. Direct solvers thus were not able to decrease simulation runtime in the program.

The best simulation acceleration could be achieved by implementing the SAMG solver library [10], which was developed by the Fraunhofer Institute for Algorithms and Scientific Computing, into the simulation kernel.

TABLE I
COMPARISON OF SIMULATION RUNTIMES WITH SAMG AND THE PREVIOUSLY IMPLEMENTED SEQUENTIAL EQUATION SOLVER FOR SIX MODELS OF DIFFERENT DISCRETIZATION SIZE

Models	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**
Number of nodes	952	3.138	8.689	18.531	18.531	21.221	135.019
Number of non-zero nodes	6.164	21.402	60.382	127.911	127.911	147.211	942.081
Runtime SAMG	16,1s	1min36s	1min15s	1min15s	1min28s	2min25s	0h22min21s
Runtime sequential	11,6s	1min53s	1min34s	1min33s	2min09s	12min21s	3h40min35s
Runtime gain	-39%	+15%	+20%	+19%	+32%	+81%	+89%

This library uses an OpenMP-parallelized multigrid solution technique to solve the Richards equation system. By adjusting the parameter set of the solver to the conditions that best fit the type of problem to be solved, significant gains in simulation runtime have been achieved (Table I). Special emphasis was put into the importance of reusing already computed multigrid structures for several successive solutions of the linear system of equations, which led to better efficiency in memory usage and computation time.

Of particular interest is the increase in runtime gain with the growing number of nodes in the models. This effect can partially be explained by the nature of the underlying multigrid algorithm, whose number of iterations necessary to solve one system of equation is virtually independent of the size of the system matrix. Although the time needed for the solution of an individual iteration is higher than for the sequential solver, the whole time needed to solve one time step is much smaller with SAMG, partly due to its excellent convergence behavior.

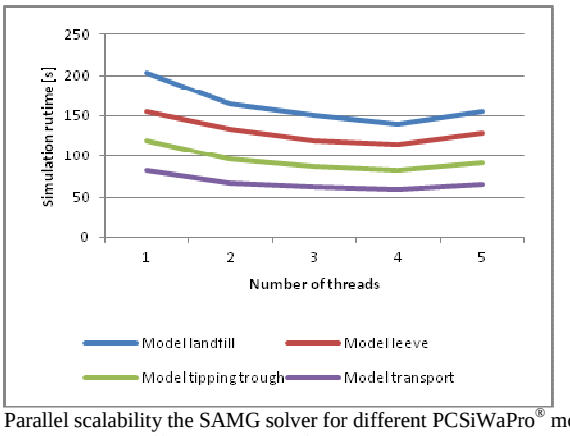

Fig. 7. Parallel scalability the SAMG solver for different PCSiWaPro® models and different numbers of OpenMP threads

SAMG showed a good parallel scalability. All the above-mentioned timing results were measured on a personal computer with an Intel Xeon 3470 processor (8M Cache, 2.93GHz) and 8GB RAM. Since this is a quadcore processor, four OpenMP threads have been used for parallelization. Figure 7 shows the use of different numbers of threads on three example models. It can be seen that with an increasing number of threads the total simulation runtime decreases, until the number of processor cores has been reached. The effect of an increasing simulation runtime with a fifth OpenMP thread can be attributed to the fact that in this case the processor switches into an inefficient time-sharing mode, with one processor core having to take care of the operations of more than one OpenMP thread.

IV. Summary and Outlook

A successful acceleration of the simulation process in PCSiWaPro® has been achieved by replacing its original linear system solver with the parallelized multigrid solver SAMG. Due to the characteristics of the PCSiWaPro® software OpenMP parallelization appeared to be a better choice for implementation instead of MPI. The new solver performed very well in the test case studies, leading to significant savings in simulation runtime for numerical meshes with a big number of nodes.

Future efforts will be made in parallelizing other parts of the software, which are not directly related to the simulation itself. A plugin weather generator, which is used to create synthetic upper boundary conditions for the PCSiWaPro® flow model, will be tested for its suitability to be parallelized. This shall not only lead to an accelerated calculation of such time series, but also to a simultaneous computation of a large amount of time series in a Monte-Carlo-approach. They will be needed in order to carry out statistical analyses on the data.

In addition to that work will be put into increasing the current approximate upper boundary of about 150,000 nodes. Since the SAMG solver has shown a better performance with an increasing number of nodes, the foundation is now set to simulate very large models with the PCSiWaPro® software. The ways which will lead to better internal memory efficiency however still have to be evaluated.

References

[1] Gräber, P.-W. et al., "SiWaPro DSS - Beratungssystem zur Simulation von Prozessen der unterirdischen Zonen", Water Resour. Res. 25(10), pp 2187-2193, 1989.

[2] Luckner et al., "A Consistent Set of Parametric Models for the Two-Phase Flow of Immiscible Fluids in the Subsurface", Simulation in Umwelt- und Geowissenschaften, Shaker Verlag 2006.

[3] Mendoza, C. A., Therrien, R. and Sudicky, E. A., "ORTHOFEM User's Guide", University of Waterloo, Ontario, Canada : Waterloo Centre for Groundwater Research, 1991

[4] "OpenMP Application Program Interface", www.openmp.org

[5] "MPI: A Message Passing Reference Standard Version 2.2", Message Passing Interface Forum, 2009

[6] Balay S. et al., "PETSc User's Manual ANL-95/11 - Revision 3.2.", Argonne National Laboratory, 2011

[7] "Lis User Manual 1.2.62", Research Institute for Information Technology, Kyushu University, 2011

[8] Schenk O. et al., "PARDISO: a high-performance serial and parallel sparse linear solver in semiconductor device simulation", Future Generation Computer Systems 18, Issue 1, pp 69-78, 2001

[9] Amestoy P. R. et al., "A fully asynchronous multifrontal solver using distributed dynamic scheduling", SIAM Journal of Matrix Analysis and Applications 23, No 1, pp 15-41, 2001.

[10] Stüben, K., Delaney, P., Chmakov, S., "Algebraic Multigrid (AMG) for Ground Water Flow and Oil Reservoir Simulation". MODFLOW and MORE, Colorado School of Mines, Golden, Colorado, 2003.

Martin Meyer. He holds a Master Degree in Geoinformation Sciences (2011) from the Technische Universitaet Freiberg, Germany. He is a research associate at the Institute of Waste Management and Contaminated Site Treatment of the TU Dresden in Germany. His research topics there are unsaturated zone modeling and coupling the institute's model PCSiWaPro® with other simulation software for groundwater flow and reactive solute transport. Further research interests are software programming, integrated water balance modeling and contaminant transport simulation.
E-mail: martin.meyer@tu-dresden.de

Jana Sallwey. She studied hydrology at the Technische Universitaet Dresden, Germany, and obtained her Diploma in 2011. Since 2011 she has been working as a research associate at the Institute of Waste Management and Contaminated Site Treatment of the TU Dresden in Germany. Her research topics include unsaturated soil zone modeling, model evaluation and groundwater recharge.
E-mail: jana.sallwey@tu-dresden.de

René Blankenburg. He studied geodesy at the Technische Universitaet Dresden, Germany, and obtained his degree in 2004. Until 2011 he worked at the Institute of Waste Management and Contaminated Site Treatment, dealing with the development of numerical simulation software for transient water flow and solute transport in variably saturated porous media. He is currently employed at IBGW® Leipzig as a groundwater modeler and software developer.
E-mail: r.blankenburg@ibgw-leipzig.de

Peter-Wolfgang Graeber. He studied electronic and control engineering and obtained his degree in 1970. He worked for the research institute of water management before switching to the Technische Universitaet Dresden in 1980, where he is now the head of the research group of modeling and simulation in water engineering. Since 1996 he has worked as a professor at the Faculty of Forest, Geo and Hydro Sciences in the field of systems analysis and environmental computing.
E-mail: peter-wolfgang.graeber@tu-dresden.de