Hybrid Parallel Solutions of the Black-Scholes PDE with the Truncated Combination Technique

Janos Benk
Technische Universität München
Boltzmannstr. 3, 85748 Garching, Germany
Email: benk@in.tum.de

Dirk Pflüger
IPVS, Universität Stuttgart
Universitätsstr. 38, 70569 Stuttgart, Germany
Email: Dirk.Pflueger@ipvs.uni-stuttgart.de

POSTER PAPER

***Abstract*—This paper presents an efficient approach to parallel pricing of multi-dimensional financial derivatives based on the Black-Scholes Partial Differential Equation (BS-PDE). One of the main challenges for such multi-dimensional problems is the curse of dimensionality, that is tackled in our approach by the combination technique (CT). This technique consists of a combination of several solutions obtained on anisotropic full grids. Hence, it offers the possibility to compute the BS-PDE on each one in an embarrassingly parallel way. Besides parallelizing on the CT level, we have developed a shared memory parallel multigrid solver for the BS-PDE. The parallel efficiency of our hybrid parallel approach is demonstrated by strong scaling results of 5D and 6D pricing problems.**

***Keywords*—Black-Scholes PDE; combination technique; sparse grids; parallelization**

I. INTRODUCTION

Classical full grid discretizations for D-dimensional problems results in N^D computational points, where N represents the number of points per axis. This effect is called *curse of dimensionality* as the number of total grid points increases exponentially with D and makes accurate computations practically impossible in more than 3 or 4 dimensions. In order to cope with the curse of dimensionality, sparse grids [2] are often employed which use considerably less points than equivalent full grids while maintaining a similar accuracy.

Since sparse grids require considerably less degrees of freedom than full grids, the resulting discrete system's size is also considerably smaller. To represent a multi-dimensional BS-PDE solution, hierarchical basis functions are used on sparse grids. The solution function is thus a weighted sum of basis functions. The basis can directly be used for discretizations based on the Finite Element Method (FEM). This approach of for the BS-PDE has been used in various forms. In [6], the system matrix is set up, and an external solver is applied, whereas in [4] and [5] an iterative solver with the so called UpDown scheme is employed, which realized the matrix-vector product, such that the system matrix is not explicitly assembled. These *direct* sparse grid approaches have the advantage that they allow for spacial adaptivity. On the other hand, solving the resulting system turns out to be algorithmically challenging.

An alternative approach to approximate the sparse grid solution is the combination technique [11] (CT) which we make use of in this paper. It consist of the combination of solutions obtained on several, D-dimensional full grids with various spacial resolutions. The CT does not allow local adaptivity, but it enables the usage of conventional full grid solvers rather than having to implement tedious matrix-vector products, which is a major advantage of this approach. For option pricing, the CT has been used several times in previous work. [7] uses the CT in combination with axial grid stretching and coordinate transformations of the BS-PDE and achieves good convergence results for 5D basket options. [9] and [10] also use grid stretching. In addition, they neglect the strongly anisotropic full grids from the CT. This idea has been used and extended in our previous publication [1] where we introduced the truncated combination technique (T-CT). This type of CT is shown in Section II.

In the following, we introduce the modeling framework of our approach and the resulting BS-PDE equation. We assume a general form of the stochastic differential equation (SDE) which models one stochastic process via

$$dS_i = \mu_i^{'}(\mathbf{S},t)dt + \sigma_i(\mathbf{S},t)dW_i \ , \ i = 1, \ldots, D. \qquad (1)$$

In (1), $\mu_i^{'}(\mathbf{S},t)$ is the drift coefficient and $\sigma_i(\mathbf{S},t)$ represents the volatility in a general form. For the case of geometrical Brownian motion, these coefficient have the form of $\mu_i^{'}(\mathbf{S},t) = \mu_{C,i}S_i$ and $\sigma_i(\mathbf{S},t) = \sigma_{C,i}S_i$, where $\mu_{C,i}$ and $\sigma_{C,i}$ are constant values. Further, we assume that the option has D underlying stochastic processes, that influence directly or indirectly the option price V. Processes that directly influence V are tradable assets and are present in the payoff function of the option, whereas processes such as interest rates and underlying volatility impact V only indirectly. The stochastic increments are correlated such that $dW_i dW_j = \rho_{i,j}dt$.

Given (1) and the initial condition $V(\mathbf{S},0) = F(\mathbf{S})$ (terminal payoff function), the resulting BS-PDE has the form:

$$\frac{\partial V}{\partial t}+\sum_{i=1}^{D}\mu_i(\mathbf{S},t)\frac{\partial V}{\partial S_i}-r(\mathbf{S},t)V$$
$$\frac{1}{2}\sum_{i=1}^{D}\sum_{j=1}^{D}\sigma_i(\mathbf{S},t)\sigma_j(\mathbf{S},t)\rho_{i,j}\frac{\partial^2 V}{\partial S_i\partial S_j}=0, \qquad (2)$$

where the relation $\mu_i(\mathbf{S},t) := \mu_i^{'}(\mathbf{S},t) - \lambda_i(\mathbf{S},t)$ holds in general. For non-arbitrage reasons, the tradable assets have a market price of risk $\lambda_i(\mathbf{S},t)$ such that $\mu_i(\mathbf{S},t) := r(\mathbf{S},t)$, whereas for non-tradable assets the market price of risks is usually set to zero. In our computations, we also set $\lambda_i(\mathbf{S},t) = 0$ for non-tradable assets. At the boundary of the full grids, we impose a boundary condition (BC) that requires the second order derivatives to be zero. Hence, we assume that V is linear at the boundary. We favor this choice of BC here, as it does not need the payoff function and approximates the exact BC well in general.

In the next sections, we present in more detail the components that play a special role in our hybrid parallelization approach. The starting point for the parallelization is the combination technique that is explained in Section II. The main task there is to distribute the full grids among the computing nodes in a load-balanced way. In Section III, we present our efficient multigrid solver on stretched full grids, that allows for shared-memory parallelization. Finally, in Section IV, we demonstrate the hybrid parallel efficiency of our implementation by pricing various derivatives in 5D and 6D.

II. The Combination Technique

The combination technique [11] uses a set of full grids with different resolutions for the coordinate axes in order to compute a sparse grid solution. The resolution of a full grid is given by a level vector $\mathbf{l} = \{l_1, \ldots, l_D\}$, denoting the discretization level per dimension, and the corresponding full grid solution is denoted by $f_\mathbf{l}$. In a given dimension i, the full grid has $O(2^{l_i})$ points and in total $O(2^{\sum_{i=1}^{D} l_i})$ points. With this notation, we restate the formula for the standard combination technique (S-CT), that was introduced in [11] as

$$f_n^C(\mathbf{S}) = \sum_{q=0}^{D-1}(-1)^q\binom{D-1}{q}\sum_{|\mathbf{l}|_1=n-q} f_\mathbf{l}(\mathbf{S}). \qquad (3)$$

There, n represents here the homogeneous level of the full grid that we are approximating with f_n^C. The $S_i, i = 1, \ldots, d$ are the D underlyings. Fig. 1 illustrates the S-CT scheme for 2D with $n = 8$.

A. The Truncated Combination Technique (T-CT)

In financial applications, the initial payoff does not satisfy the S-CT's smoothness conditions for the upper error bounds (see [3] and [2]): the mixed second derivatives are not bounded. To cope for this, a modification to the S-CT

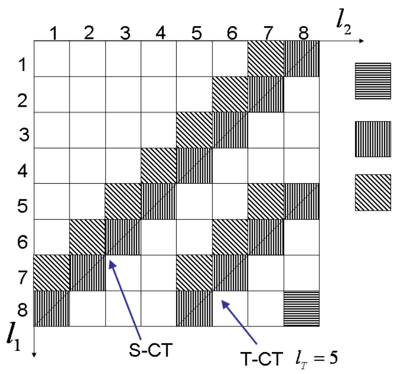

Figure 1. Illustration of the S-CT with $n = 8$ and the T-CT with $l_T = (5,5)$ in 2D. Each square represents an inisotropic full grid with level vector (l_1, l_2). Both S-CT and T-CT approximate the same full grid solution $f_{\{8,8\}}$.

is required. We have used the idea from [9] previously [1], where we introduced a truncated combination technique (T-CT). The idea is to introduce a truncation level vector $\mathbf{l}_T = l_{1,T}, \ldots, l_{D,T}$ that eliminates all the full grids $f_\mathbf{l}$ from the S-CT for which $l_i < l_{i,T}$. We have shown in [1], that the T-CT with $l_{i,T} = n/2$ results in correct and convergent option prices even in higher dimensions. Therefore, we use T-CT in all the numerical examples of this paper. In Fig. 1, the T-CT is illustrated for $n = 8$ and $l_T = 5$. One of the obvious implications of the T-CT are that the anisotropic full grids have potentially more unknowns (increased impact of the curse of dimensionality) and that the number of full grids is smaller in comparison to the S-CT.

B. Aspects of Hybrid Parallel Computations on Distributed Memory Systems

The combination technique offers an embarrassingly parallel concept, where each of the full grid solution is computed independently and without the need for intermediate synchronization. We denote the number of full grids by M and the number of processes by P. In the computations, P represents the number of MPI[a] processes. Since we are using the T-CT, we can not assume $M \gg P$ here, as it was the case in [8], where an efficient parallelization based solely on the distribution of full grids was possible. Due to the truncation effect, we have $M < 100$ in our applications. Therefore, an efficient scheduling algorithm is required that distributes the full grids among the processes in a load-balanced way.

In the first step, we assume that the full grids are sorted in a list according to their descending Manhattan norm $|\mathbf{l}| = \sum_{i=1}^{D} l_i$. Initially, we distribute the first P full grids from the list to the P MPI processes. If one of the P processes finishes the computation of the assigned full grid, it asks the master node for the next grid. This way, the next $M - P$ full grids are distributed in a first-come first-served way. Once all the M full grids have been assigned, we wait for all the P processes

[a]http://www.mcs.anl.gov/research/projects/mpich2/

to finish. Only when all P processes computed their last task, we can evaluate the result. Ideally, the last P full grids that the processes solve should have considerably less unknowns compared to the first $M - P$ full grids from the list in order to avoid unnecessary long waiting times.

We also mention here that the full grids are anisotropically discretized (in general, $l_i \neq l_j$). Hence, the solver times might vary among grids with the same number of grid points. In order to speed up the full grid solver, we developed a shared memory parallel multigrid solver based on OpenMP which we describe later. Thus, our hybrid parallel computation of the BS-PDE consist of the MPI parallelization of the combination technique, and of the shared memory OpenMP parallelization of the multigrid solver which runs on one computing node. This multigrid solver is presented next.

III. MULTIGRID SOLVER ON STRETCHED FULL GRIDS

In each implicit time step of the discretized form of Eq. (2), a linear system needs to be solved. Note that a stable explicit time stepping would result in inefficiently small time steps. We use the second order implicit Crank-Nicolson scheme for the time discretization, whereas the spacial discretization of (2) is done by second order Finite Differences.

Multigrid methods are one of the most effective solvers for linear and non-linear systems [12]. They have the potential to solve a discrete system in only $O(N)$ operations, where N is the number of unknowns. In our applications, the number of unknowns N for the highest resolution on a full grid is of order $10^5 - 10^6$. Therefore, an efficient implementation is crucial. Multigrid methods have been already used for option pricing in [8] and [17]. The main challenge for the multigrid implementation in option pricing is to tackle the anisotropy in the equation and in the discretization. In the combination technique (see Fig. 1), some of the full grids have strong anisotropic discretizations. Additionally, the convection and diffusion terms in Eq. (2) are potentially anisotropic. One way to tackle the anisotropy is to use a more robust smoother [12]. This idea is also used in [7] and [8] as well as in [17].

We, however, employ a different idea. Stretched full grids are employed in order to increase the accuracy in the locality of the SDE's (1) expected value, where also the evaluation point is located. Note that the price of the option is the value at the evaluation point. Such a stretched grid in 2D with a call option's payoff is illustrated in Fig. 2. This approach not just increases the accuracy at the evaluation point, but also counterbalances the anisotropy in the BS-PDE. Hence, a classical correction scheme with a Gauss-Seidel smoother [12] can be applied and good convergence results are achieved. Due to the anisotropic discretization of the T-CT, for a few extreme grids, slow convergence or even divergent behaviour can be detected. For these cases, we increase the number of pre- and post-smooth iterations, such that the smoothing on each level becomes more robust. This way, we achieve convergence for all full grids.

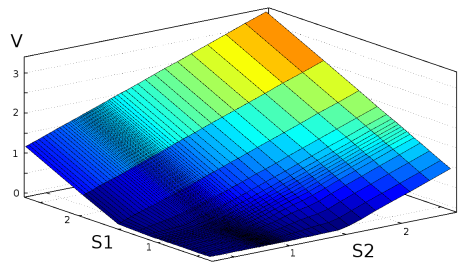

Figure 2. Stretched grid representing the payoff of a two-dimensional call option's payoff. $S1$ and $S2$ represent the two underlyings in the basket.

The initial value for the multigrid's solution is computed by a second order predictor method, which further reduces the computational time. In the following, we focus on the shared memory parallel implementation of the multigrid method.

A. *Shared Memory Parallelization*

The shared memory parallel multigrid solver is an important component of our hybrid parallelization concept. In our implementation, we choose OpenMP [b] for the shared memory parallelization and denote the number of OpenMP threads by O. In Section II, we have described the MPI parallelization that assigns one full grid to one MPI process on the fly. Here, the goal is to solve efficiently one MPI task on a multicore architecture. Several implementational aspects need to be considered at all stages of the multigrid solver:

- **data initialization**: Initially, the solver is called with one single thread which creates the hierarchical data structure for the solver. The main task is to create and initialize these data structures in such a way that each thread uses local data and cache misses are avoided. Therefore, we assign the unknowns on each hierarchical level of the multigrid in a static way and the data structures are initialized by the corresponding OpenMP thread. Thus, according to the *touch first* principle, these data structures will be local on each core.
- **Gauss-Seidel smoother**: In order to avoid synchronization, the shared memory smoother is turned into a block Gauss-Seidel smoother, where the unknowns are assigned statically to the OpenMP threads.
- **prolongation and restriction**: These operations do not require any synchronization and can be parallelized efficiently.

The parallel efficiency of our multigrid implementation is demonstrated in the following section.

[b]http://www.openmp.org/

IV. Numerical Examples

We now show numerical results of the pricing of 5- and 6-dimensional derivatives. Besides the strong scaling results of each example, we also show the convergence result of our numerical approach in terms of relative pointwise, L_2, and L_∞ errors. The L_2 and L_∞ errors are measured on full grid with 9^D grid points that coveres the vicinity of the evaluation point. The measured errors show not only the pointwise convergence of our approach, but also the convergence on this full grid. All the examples were computed with the T-CT, where $l_T = n/2$.

All the presented methods were implemented in the FITOB[c] toolbox, that is developed at TUM. The computations were made on the SGI ICE cluster of the LRZ computing centre[d]. In this cluster, one blade features two Intel Nehalem Xeon processors (Intel Xeon E5540 at 2.53 GHz) and two Infiniband connects. Therefore, the maximum number of OpenMP threads is limited to only eight here.

A. European Option

We start with an European basket call option in 5D and in 6D, where the underlying assets S_i, $i = 1, \ldots, D$ are modeled by geometric Brownian motions and the risk-free interest rate is set to a constant value $r(\mathbf{S}, t) = 5\%$. The terminal payoff function is given by $V = \max(0, \sum_{i=1}^{D} S_i - D)$, and all the initial values are $S_i(0) = 1$, $i = 1, \ldots, D$. The maturity of the option is one year ($T = 1$). We set the model parameters (see Section I) to homogeneous values $\mu_{C,i} = 5\%$ and $\sigma_{C,i} = 40\%$, and the correlations are also set homogeneously to $\rho_{i,j} = 0.1$, $i \neq j$. We emphasize here that we set these parameters homogeneously only for the sake of simplicity. Our approach equally copes with heterogeneous model parameters and correlation matrix as it will be shown in a later example.

The convergence results for 5D and 6D are presented in Tab. I and Tab. II respectively. In these convergence studies, we increase the level n from three to six and all the three error types are measured relative to the $n = 6$ result. The third column (rel. p. error) represents the relative pointwise error in the price, whereas the last two columns represent the error measured in the locality of the evaluation point on a regular full grid. We notice for both 5D and 6D that we achieve convergence with respect to all error measures: in the last refinement step, the price is changed only by 0.4-0.7%.

TABLE I
5D European Call option's convergence results.

n	price	rel. p. error	$\lvert\text{error}\rvert_{L_2}$	$\lvert\text{error}\rvert_{L_\infty}$
3	0.53016	-3.801e-2	5.069e-2	2.906e-1
4	0.54894	-3.942e-3	4.119e-2	3.422e-1
5	0.55068	-7.829e-4	9.953e-4	3.203e-3
6	0.55111			

[c]http://www5.in.tum.de/wiki/index.php/FITOB_Webpage
[d]http://www.lrz.de/services/compute/linux-cluster/overview/

TABLE II
6D European Call option's convergence results.

n	price	rel. p. error	$\lvert error\rvert_{L_2}$	$\lvert error\rvert_{L_\infty}$
3	0.61475	-3.095e-2	2.290e-2	8.979e-2
4	0.62532	-1.428e-2	5.244e-2	6.435e-1
5	0.63413	-4.047e-4	7.499e-4	1.979e-3
6	0.63438			

We consider for a 5D strong scaling study the computation from Tab. I with $n = 6$. In this case, the T-CT consists of 56 full grids, where the maximal number of unknowns per full grid is $2.5 \cdot 10^5$. This results in a total of $\sim 10^7$ unknowns in 56 independent linear systems for one discrete time step. For the maturity of $T = 1$, in average 50 time steps are required. In comparison, a full grid discretization with $n = 6$ in 5D would result in a single computation with 10^9 unknowns. The strong scaling study in Tab. III shows that we reduced the run time from the sequential run of 2.5 hours to only 3.5 minutes with 128 processors. The number of MPI processes is denoted by P and the number of OpenMP threads by O. For a constant processor number, we show the optimal configuration of (P, O), and we note that in the first steps of the scaling study it was more efficient to increase P only. However, for higher processor numbers, a pure MPI-based parallelization would result in a large waiting time at the end of the computations, where all MPI processes wait for the last one to finish (see Section II). Hence, starting from a given processor number it is beneficial to increase O instead of P. In the last step, we also notice a significant decrease in the efficiency. This is due to one full grid from the T-CT that requires considerably longer computing times. This effect could be counterbalanced by a higher number of OpenMP threads, which was not possible on the SGI ICE cluster.

TABLE III
Run times for the 5D European call option with $n = 6$.

# processors	1	2	4	8
(P, O)	$(1,1)$	$(2,1)$	$(4,1)$	$(4,2)$
sol. time [s]	8640	4440	3060	1513
eff. [%]	100	96	71	71
# processors	16	32	64	128
(P, O)	$(4,4)$	$(8,4)$	$(8,8)$	$(16,8)$
sol. time [s]	928	427	221	216
eff. [%]	58	63	61	31

TABLE IV
Run times and parallel efficiency for the 6D European call option with $n = 6$.

# processors	8	16	32	64	128
(P, O)	$(4,2)$	$(4,4)$	$(8,4)$	$(8,8)$	$(16,8)$
sol. time [s]	19741	16200	6023	3628	2025
eff. [%]	100	98	82	94	81

In 6D, the resulting run times for $n = 6$ are shown in Tab. IV. The T-CT is composed of 84 full grids, where the maximal number of unknowns per grid is $\sim 2 \cdot 10^6$. This results in a total of $\sim 10^8$ unknowns in 84 independent linear systems for a discrete time step. In comparison, a full grid discretization with $n = 6$ in 6D would result in $6.8 \cdot 10^{10}$ unknowns. We start the strong scaling study with 8 processors for which the computation takes more than 7 hours. Note that a sequential run could potentially hit the maximum allowed wall-clock-time of the cluster. A run with 128 processors reduces the run time to 55 minutes and results in a parallel efficiency of 81% compared to the run with 8 processors.

B. American Option

In the following, we extend the 5D European option with American features. We use the configurations from the previous example, except for the payoff function. It is given for the put option as $V = \max(0, D - \sum_{i=1}^{D} S_i)$. The American feature is imposed at small discrete time steps by setting the pointwise values to $V = \max(V, D - \sum_{i=1}^{D} S_i)$. Therefore, we limit the maximal time step to 0.01. This way, $130 - 180$ discrete time steps are made until the maturity $T = 1$. (Note that we use an adaptive time step size.) At each of these discrete time steps, the early exercise right of the American option is imposed. The convergence results for the 5D put option is presented in Fig. V. We observe convergence pointwise and on a local domain: in the last refinement step the price is changed only by 0.3%.

TABLE V
5D AMERICAN CALL OPTION'S CONVERGENCE RESULTS.

n	price	rel. p. error	$\lvert error\rvert_{L_2}$	$\lvert error\rvert_{L_\infty}$
3	0.30393	-8.597e-2	6.396e-2	2.842e-1
4	0.33456	6.137e-3	5.301e-2	4.113e-1
5	0.33152	-3.008e-3	1.660e-3	4.515e-2
6	0.33252			

For strong scaling measurements, we consider the scenario with level $n = 6$. Analogous to the European 5D case, the T-CT is composed of 56 full grids. This time, we do not obeserve a drop in the efficiency for 128 processors. The good scaling is due to the larger computational times of each full grid, compared to the European option. With 128 processors and in 5D, we were able to price an American put option accurately in only five minutes and 29 seconds. This nicely shows the practical applicability and the true potential of our implementation.

TABLE VI
RUN TIMES AND PARALLEL EFFICIENCY FOR THE 5D AMERICAN PUT OPTION WITH $n = 6$.

# processors	8	16	32	64	128
(P, O)	$(4, 2)$	$(4, 4)$	$(8, 4)$	$(8, 8)$	$(16, 8)$
sol. time [s]	4260	2160	1310	569	329
eff. [%]	100	98	81	93	81

C. Guaranteed Minimum Withdrawal Benefit

As a last example, we consider a heterogeneous scenario with considerably longer maturity, where different stochastic models are used. The Guaranteed Minimum Withdrawal Benefit (GMWB) is a special type of Variable Annuity (VA) that can be seen as a private retirement investment. The contract starts with the purchase of the product, after which an initial investment of the nominal value into an asset index S is made (the nominal value is 1.0 in Fig. 3). This is followed by two phases; the first one is the inactive phase where no action happens and the nominal value stays invested in S. In our example, this phase is assumed to be 10 years long. In the second phase, the so called deferral phase, the holder receives a yearly payment of $CI = 0.15$. This payment is paid out from the initial investment in S. Even if the asset S performs badly, the issuer must pay the holder yearly CI (guaranteed withdrawal). We assume that this phase is also 10 years long and at the end of this period the holder receives the remaining value of the investment (if this is greater than zero). The overview of the cash-flows is presented in Fig. 3.

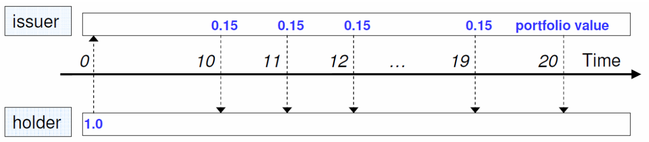

Figure 3. Cash flows of the modeled GMWB.

Furthermore, we extend the product in Fig. 3 with yearly early exercise rights, where the holder can withdraw all the value of the actual investment and terminate the contract. If the holder is not exercising this right and the value of the investment is greater than zero, a fee of 2% of the investment value is paid to the issuer.

The modeling of such complex products requires not just to solve the BS-PDE with a terminal payoff, but also to map values from one grid to another one. This operation is called *Da operation* and was introduced by [1] and [14]. Basically, this implies the initialization of the values of a newly created grid with the evaluation values of an other grid. The described GMWB can be uniquely represented by a script (ThetaML script, see [1], [14]) which is presented in Appendix V.

For the asset price S, we use the Heston model [15], where the volatility is modeled by a mean-reverting process

$$\begin{aligned} dS(t) &= \mu\, S(t)\, dt + \sqrt{v(t)} dW^{(1)} \\ dv(t) &= \kappa(\tilde{v} - v(t))dt + \sigma \sqrt{v(t)} dW^{(2)}. \end{aligned} \tag{4}$$

In addition, we model the interest rate by the CIR model [16],

$$dr(t) = \lambda_r(\theta_r - r(t))dt + \sigma_r \sqrt{r(t)} dW_r^3. \tag{5}$$

We set the parameters of the Heston model to $\mu = r(t)$, $\kappa = 2.0$, $\tilde{v} = 0.06$, and $\sigma = 0.3$. We assume that there is a negative correlation between the volatility and the asset S, namely $\rho_{1,2} = -0.5$, and the other factors do not correlate, i.e., $\rho_{1,3} = \rho_{2,3} = 0.0$. (Note that our approach allows for non-zero correlation, e.g., $\rho_{1,3} = \rho_{2,3} = 0.3$.) The CIR model parameters are $\lambda_r = 3.0$, $\theta_r = 0.05$ and $\sigma_r = 0.4$, whereas the initial values are $v(0) = 0.05$, $S(0) = 1.0$, and $r(0) = 0.01$.

As shown in Appendix V, the three dimensions need to be extended for accurate modelling with the value of the investment PF and the value of the asset at the previous payment date $Sprev$. This way, the resulting problem is five-dimensional. For the interest rate and the underlying volatility dimensions, we use a constant level of 4 in the T-CT. In the convergence study of Tab. VII, n represents the level of T-CT only in the other three remaining dimensions. Even for such complex product with a maturity of 20 years and with additional cash flow operations, the resulting errors show a good convergence, where in the last refinement step the price is changed only by 0.8%. The T-CT for $n = 8$ and $l_T = n/2$ results in only 28 full grids with a maximum of $2.5 \cdot 10^6$ unknowns per grid. Since the mapping of the grid values from one grid to an other is implemented only in an OpenMP context, we use in the strong scaling study of Tab. VIII considerably larger values for O than for P. Due to the grid mappings, the lower number of full grids, and the upper limit of 8 for O as given by the hardware, we achieved a reasonable efficiency only up to 64 processors. The run time was reduced from 24 hours with 16 processors to 11 hours with 64 processors.

TABLE VII
GMWB'S CONVERGENCE RESULTS.

n	price	rel. p. error	$\|error\|_{L_2}$	$\|error\|_{L_\infty}$
4	1.8716	4.832e-1	7.404e-1	1.276e-0
5	1.3245	4.895e-2	7.534e-2	1.275e-1
6	1.2381	-1.880e-2	2.819e-2	4.727e-2
7	1.2514	-8.233e-3	1.086e-2	1.554e-2
8	1.2618			

TABLE VIII
GMWB'S RUN TIMES.

# processors	16	32	64
(P, O)	$(4, 4)$	$(4, 8)$	$(8, 8)$
sol. time [s]	85253	50871	40479
eff. [%]	100	83	53

V. APPENDIX

We finally present the detailed modeling of the described GMWB by a script, called Theta script [1], [14]. There S denotes the asset price, PF the portfolio value (actual value of investment), and Sprev the value of the asset S at the previous time step. The continuous income (guaranteed benefit) is denoted by CI. $Theta\ t$ represents the waiting time t, where the BS-PDE needs to be solved. V is the price of the GMWB and its terminal condition is given by the portfolio value $V = \text{MAX}(PF, 0)$. The line $V = \text{MAX}(V + CI, PF)$ represents the early withdrawal right, where the value of the GMWB is the maximum of the two expressions. In the loop, the value of the actual investment PF is updated. The expression $0.98 * (PF - CI + (PF/Sprev) * (S - Sprev))$ incorporates several effects that influence the investment PF: fee payment (factor 0.98), CI payment, and the change in the asset price $(S - Sprev)$.

```
CI = 0.15;
Theta 10;
PF=S;
Sprev = S;
loop (10)
   V=MAX(V+CI,PF);
   PF = 0.98*(PF-CI+(PF/Sprev)*(S-Sprev));
   Sprev = S;
   Theta 1;
end;
V=MAX(PF, 0.0 );
```

ACKNOWLEDGMENT

We thank for the support from the German Federal Ministry of Education and Research (BMBF).

REFERENCES

[1] J. Benk, H.-J. Bungartz, A.-E.Nagy, and S. Schraufstetter, "Variants of the combination technique for multi-dimensional option pricing." In Progress in Industrial Mathematics at ECMI 2010, 2010.

[2] H.-J. Bungartz and M. Griebel. "Sparse grids." Acta Numerica, 13:1-123, 2004.

[3] H.-J. Bungartz: "Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung." Dissertation, Fakultät fr Informatik, Technische Universität München, November 1992

[4] H.-J. Bungartz, A. Heinecke, D. Plüger, and Stefanie Schraufstetter. "Option pricing with a direct adaptive sparse grid approach." Journal of Computational and Applied Mathematics, 2011 doi 10.1016/j.cam.2011.09.024

[5] H.-J. Bungartz, A. Heinecke, D. Pflğer and S. Schraufstetter. "Parallelizing a Black-Scholes solver based on finite elements and sparse grids." In Concurrency and Computation: Practice and Experience. John Wiley & Sons, Ltd, März 2012.

[6] T. Mertens. Optionspreisbewertung mit dünnen Gittern. Diplomarbeit, Institut für Numerische Simulation, Universität Bonn, 2005.

[7] C. Reisinger. "Numerische Methoden für hochdimensionale parabolische Gleichungen am Beispiel von Optionspreisaufgaben." PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2004.

[8] C. Reisinger, G. Wittum: "Efficient Hierarchical Approximation of High-Dimensional Option Pricing Problems." SIAM Journal on Scientific Computing, 29(1), 2007.

[9] C.C.W. Leentvaar. "Pricing multi-asset options with sparse grids." PhD thesis, Tu Delft, 2008.

[10] C.C.W. Leentvaar and C.W. Oosterlee. "On coordinate transformation and grid stretching for sparse grid pricing of basket options." Journal of Computational and Applied Mathematics, 222(1):193-209, 2008.

[11] M. Griebel, M. Schneider, and C. Zenger. "A combination technique for the solution of sparse grid problems." In P. de Groen and R. Beauwens, editors, Iterative Methods in Linear Algebra, pages 263-281. IMACS, 1992.

[12] P. Wesseling. "An introduction to multigrid methods." J. Wiley, 1992.

[13] C.W. Oosterlee. "On multigrid for linear complementarity problems with application to american-style options." Electr. Trans. on Num. Anal., 15:165-185, 2003.

[14] S. Schraufstetter and J. Benk. "A general pricing technique based on thetacalculus and sparse grids." In Proceedings of the ENUMATH Conference, Uppsala, 2010.

[15] S. L. Heston. "A closed-form solution for options with stochastic volatility with applications to bond and currency options." Review of Financial Studies, 6:327-343, 1993.

[16] J. C. Cox, J. E. Ingersoll, and S.A. Ross. "A theory of the term structure of interest rates." Econometrica, 53(2):385-407, March 1985.

[17] C.W. Oosterlee. "On multigrid for linear complementarity problems with application to american-style options." Electr. Trans. on Num. Anal., 15:165-185, 2003.