International Journal of Computer Mathematics
Vol. 84, No. 8, August 2007, 1129–1147

Taylor & Francis
Taylor & Francis Group

Efficient *d*-multigrid preconditioners for sparse-grid solution of high-dimensional partial differential equations

H. BIN ZUBAIR*, C. C. W. LEENTVAAR and C. W. OOSTERLEE

Numerical Analysis Group, Delft Institute of Applied Mathematics Delft University of Technology, The Netherlands

(Received 29 December 2006; revised version received 31 January 2007; accepted 04 February 2007)

Fast and efficient solution techniques are developed for *high-dimensional* parabolic partial differential equations (PDEs). In this paper we present a robust solver based on the Krylov subspace method Bi-CGSTAB combined with a powerful, and efficient, multigrid preconditioner. Instead of developing the perfect multigrid method, as a stand-alone solver for a single problem discretized on a certain grid, we aim for a method that converges well for a wide class of discrete problems arising from discretization on various anisotropic grids. This is exactly what we encounter during a sparse grid computation of a high-dimensional problem. Different multigrid components are discussed and presented with operator construction formulae. An option-pricing application is focused and presented with results computed with this method.

Keywords: Anisotropic diffusion equation; Coarsening strategies; High-dimensional PDE; Sparse grids; Multigrid preconditioned Bi-CGSTAB

AMS Subject Classifications: 65M55; 65Y20; 91B28

1. Introduction

Multidimensional PDE have applications in a variety of applied sciences. Mentioning a few would include financial engineering [1, 2], quantum mechanics [3, 4] and molecular life-sciences [5]. In the context of their numerical approximation there are two main limiting factors on computing speed; one is the phenomenon of *temporal-march* of successive solutions and the other is the *high-spatial dimensionality* of the problem which renders an exponential computational complexity on regular tensor product grids. Traditionally, this exponential growth in the number of discrete unknowns is known as the *curse of dimensionality* [6], and it has continually eluded and marred solution techniques. The *sparse-grid* solution method [7–9] relieves this so-called curse to some extent; it is based on discretizing the problem on many grids, each with lesser number of nodes (sparse grids), solving these subproblems, and then combining the solutions to obtain a *mimic* of the solution on the original *dense* grid. Different combination techniques can be employed; we use the technique proposed in [8].

*Corresponding author. Email: h.binzubair@tudelft.nl

International Journal of Computer Mathematics
ISSN 0020-7160 print/ISSN 1029-0265 online © 2007 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00207160701356365

Efficiency of the sparse grid solution method depends on the efficient solution of the underlying subproblems, each of which has the same spatial dimensionality as the original problem and which therefore requires an efficient and robust solution approach. The method that we develop and implement in this work is d-multigrid preconditioned Bi-CGSTAB. In what follows, d-multigrid refers to multigrid for an arbitrary number of space dimensions, which in turn is abbreviated by the letter d.

Bi-CGSTAB [10] is a well known iterative solver. It is generally accepted that all iterative solvers based on Krylov subspaces require preconditioning for faster convergence. Preconditioning is a process aimed at clustering the scattered eigenvalues of the coefficient matrix. It is important to point out that the performance of stand-alone multigrid is dependent significantly on the choice of optimal parameters and components, especially for high-dimensional problems; this is not quite the case when multigrid is used as a preconditioner. By choosing multigrid as a preconditioner we do not need to search for the ideal under-relaxation, which is grid-anisotropy dependent, for example, but we can stay with a fixed parameter. Important theoretical and experimental insights into multigrid preconditioning of Krylov subspace solvers can be gained from earlier work in this context [11–13].

In section 2 we point out that the Black–Scholes multi-asset option-pricing PDE can be reduced to a standard d-dimensional diffusion equation, which underscores the need of an efficient solver for discrete diffusion systems. Moreover, we give the discretization of the continuous model problem with second order finite differences along with implementation in d-dimensions through Kronecker tensor products. Next, in section 3 we present the sparse grid technique along with the computation of accuracy bounds. Section 4 deals with the preconditioner and its components which include point smoothing and grid transfer strategies. These strategies are based on the idea of repeated partial coarsening in the direction(s) of strong coupling. In section 5 we present experimental results based on the full-grid solution method. This includes results for d-multigrid employed both as a stand-alone solver as well as a preconditioner. Section 6 contains results from numerical experiments (on the model problem) based on the sparse grid technique with d-multigrid preconditioned Bi-CGSTAB. There we demonstrate in tables and figures the results that we obtained; and finally, in section 7 we solve the same transformed Black–Scholes PDE of section 2, exhibiting the convergence of the proposed method in a real application. We draw some conclusions from this work in section 8.

2. The application, model problem and discretization

The multi-asset Black–Scholes option pricing (parabolic) PDE [2] is defined as:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{d}\rho_{ij}\sigma_i\sigma_j S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} + \sum_{i=1}^{d}(r-\delta_i)S_i\frac{\partial V}{\partial S_i} - rV = 0,$$
$$(0 < S_1, \ldots, S_d < \infty, \quad 0 \leqslant t < T). \tag{1}$$

V stands for the option price; S_i are the d underlying asset-prices; t, the current time; ρ_{ij}, the correlation coefficients between the ith and the jth asset-prices; σ_i, the volatility of the ith asset-price; r, the risk free interest-rate and δ_i the continuous dividend yield for asset i. The equation comes with a final condition,

$$V(\mathbf{S}, T) = \max\left\{\sum_{k=1}^{d} w_k S_k - K, 0\right\}, \tag{2}$$

where K is the exercise price and $0 \leq w_k \leq 1$ are the weights of the assets in the basket. The PDE (1) is a second order partial differential equation in d dimensions and $2d$ boundary conditions are mandatory. As the asset price domain is truncated $S_k \in [0, S_k^{\max}]$, we first of all need a boundary condition at $S_k = 0$. When using the reduced form of equation (1), where every coefficient belonging to a derivative with respect to S_k vanishes at $S_k = 0$, a $(d-1)$-dimensional partial differential equation remains at the boundary. This is sometimes called the *natural boundary condition*. In particular, the boundary condition at $S_1 = 0$ or $S_2 = 0$ for a two-asset option is represented by the well-known 1D Black–Scholes equation for a vanilla option.

Also for $S_k = S_k^{\max}$ a boundary condition must be prescribed. If $S_k^{\max}$ is large enough, i.e. $w_k S_k^{\max} \gg K$, a linearity condition can be applied, which means that the option price can be assumed to show a linear growth in that coordinate direction. In this case we set the second derivative with respect to S_k equal to zero at that boundary. All other derivatives remain present. An appropriate size of the truncated domain is important for this boundary condition not to have a negative effect on the option prices at the spot price and/or at the exercise price K.

It is known [1, 2] that a simple log transform can convert (1) into the following d-dimensional diffusion equation:

$$\frac{\partial V}{\partial \tau} = \frac{1}{2} \sum_{i=1}^{d} \frac{\partial^2 V}{\partial {x_i}^2}, \quad -\infty < x_i < \infty, \quad 0 < \tau \leqslant T. \tag{3}$$

Thus we choose the d-dimensional diffusion equation, with transformed initial conditions and Dirichlet boundary conditions, to serve as our model problem.

In what follows $\boldsymbol{x}$ is a d-tuple $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$. For $\{a_i, b_i, c_i, \tau_1, \tau_2\} \in \mathbb{R}$ and $c_i > 0$ the model problem reads:

$$\begin{aligned} \frac{\partial}{\partial t} u(\boldsymbol{x}, t) &= \sum_{i=1}^{d} c_i \frac{\partial^2}{\partial {x_i}^2} u(\boldsymbol{x}, t); \quad \boldsymbol{x} \in \left(\Omega = \prod_{i=1}^{d} [a_i, b_i] \right) \subset \mathbb{R}^d; \quad t \in [\tau_1, \tau_2]; \\ u(\boldsymbol{x}, t) &= f^{\Gamma}(\boldsymbol{x}, t); \quad \boldsymbol{x} \in \Gamma = \partial\Omega; \quad x_i \in \{a_i, b_i\}. \end{aligned} \tag{4}$$

The spatial discretization of the model problem (4) is $O(\Sigma_{i=1}^{d}(h_i^2))$ finite difference $(2d+1)$ stencil, h_i is the mesh size along the ith space dimension. We chose the implicit (second order) Crank–Nicolson time stepping scheme for the temporal discretization. For k being the size of the time-step with $d = 2$ this yields the following iterative representation in stencil notation:

$$\begin{aligned} &\begin{bmatrix} & -\dfrac{c_2}{2h_2^2} & \\ -\dfrac{c_1}{2h_1^2} & \left\{ \dfrac{c_1}{h_1^2} + \dfrac{c_2}{h_2^2} + \dfrac{1}{k} \right\} & -\dfrac{c_1}{2h_1^2} \\ & -\dfrac{c_2}{2h_2^2} & \end{bmatrix} u_{\boldsymbol{h},k}(x_1, x_2, t+k) \\ &= \begin{bmatrix} & \dfrac{c_2}{2h_2^2} & \\ \dfrac{c_1}{2h_1^2} & \left\{ -\dfrac{c_1}{h_1^2} - \dfrac{c_2}{h_2^2} + \dfrac{1}{k} \right\} & \dfrac{c_1}{2h_1^2} \\ & \dfrac{c_2}{2h_2^2} & \end{bmatrix} u_{\boldsymbol{h},k}(x_1, x_2, t). \end{aligned} \tag{5}$$

We have Dirichlet boundary conditions prescribed at all boundaries of the spatial *hyper* domain. Inclusion of the boundary grid coordinates in the grid-point enumeration scheme (the non-eliminated boundary scheme) results in M grid points. The spatial discretization grid is given by $\boldsymbol{N} = [N_1, N_2, \ldots, N_d]$, N_i represents the number of divisions along the ith dimension, and the total number of points in the (finest) grid is $M = \prod_{i=1}^{d}(N_i + 1)$. We represent the index of a grid-point by a d-tuple $(j_d, j_{(d-1)}, \ldots, j_1)$, which is the d-dimensional extension of the two-dimensional lexicographic enumeration scheme. Written in terms of matrix operators, the *d-dimensional representation* of (5) is:

$$(\boldsymbol{L}_h + \boldsymbol{D}_k)u_{\boldsymbol{h},k}(\boldsymbol{x}, t + k) = (-\boldsymbol{L}_h + \boldsymbol{D}_k)u_{\boldsymbol{h},k}(\boldsymbol{x}, t). \tag{6}$$

The matrix-operators have the order $(M \times M)$. $\boldsymbol{D}_k$ is a diagonal matrix containing $1/k$ on the main diagonal, and $\boldsymbol{L}_h$ is the iteration matrix obtained in the following way:

$$\boldsymbol{L}_h = \boldsymbol{X}_h + \boldsymbol{B}_h \tag{7}$$

$$\boldsymbol{X}_h = \sum_{i=1}^{d} \left\{ \bigotimes_{j=i}^{d-1} \mathbf{I}_{(d+i-j)} \otimes \boldsymbol{X}_i \otimes \bigotimes_{j=1}^{i-1} \mathbf{I}_{(i-j)} \right\}. \tag{8}$$

The operator $\boldsymbol{L}_h$ is the spatial operator consisting of the *interior point operator* $\boldsymbol{X}_h$ and the *boundary point operator* $\boldsymbol{B}_h$. $\boldsymbol{X}_i$ is the difference operator matrix (of order $N_i + 1$) obtained by substituting 0 for the boundary grid-points and applying the following Crank–Nicolson difference stencil to all the interior grid-points along the ith dimension:

$$\frac{c_i}{h_i^2}\left[-\frac{1}{2} \quad \underline{1} \quad -\frac{1}{2}\right]. \tag{9}$$

$\mathbf{I}_i$ is a diagonal matrix of order $(N_i + 1)$ containing 1 on the main diagonal, except at the first and the last positions (boundary positions) where it is 0.

Finally, we represent the Crank–Nicolson iteration matrix as $\boldsymbol{A} = \boldsymbol{L}_h + \boldsymbol{D}_k$. Different grid realizations of this matrix are used at every grid level during the multigrid process. Kronecker tensor products are employed in this work for defining the d-dimensional operators. They are non-commutative and associative operations (see [14]). In the formulae presented above $\otimes$ is the Kronecker tensor product of matrices and $\bigotimes$ is the cumulative Kronecker tensor product in the same sense as the cumulative sum Σ, or the cumulative product Π. The commutative order is determined by the subscripts and the associative hierarchy is immaterial. This completes the discussion of the discretization issues that arise from the arbitrary spatial dimensionality of the model problem.

3. The sparse grid method

Consider the task of the numerical approximation of a parabolic d-dimensional problem discretized with $N = 2^n$ points per spatial coordinate. The grid thus formed is termed as a *full-grid* and a full-grid based solution process involves (at the minimal), vectors of the size $2^{n.d}$. For six space dimensions and only 32 divisions along each axis, the storage cost is around 9 gigabytes per vector, and grows worse for increasing d. The sparse grid approach, developed by Zenger and co-workers [7, 9] is a technique that splits the full grid problem of N^d points up into layers of subgrids. Each subgrid represents a coarsening in several coordinates up to a minimal required number of points. In the so-called *sparse grid combination technique*, the

partial solutions that are computed on these grids, are combined *a posteriori* by interpolation to a certain point or region.

DEFINITION 3.1 *A multi-index $\mathcal{I}_d$ belonging to a d-dimensional grid is a collection of numbers n_i, $i = 1, \ldots, d$, which represents a d-dimensional grid with N_i grid points in coordinate i, with $N_i = 2^{n_i}$. The sum of a multi-index $|\mathcal{I}_d|$ is defined by*:

$$|\mathcal{I}_d| = \sum_{i=1}^{d} n_i. \tag{10}$$

According to Definition 3.1 the multi-index $\mathcal{I}_d$ of a full grid with $N = 2^n$ points per coordinate reads $\mathcal{I}_d = \{n, n, \ldots, n\}$, with $|\mathcal{I}_d|$ being the layer number.

The full grid solution will be denoted by u_n^f; the sparse grid solution after the combination will be denoted by u_n^c and the exact solution by u_E. Now, we can define the following [8].

DEFINITION 3.2 *The combined sparse grid solution u_n^c corresponding to a full grid solution u_n^f reads*

$$u_n^c = \sum_{k=n}^{n+d-1} (-1)^{k+1} \binom{d-1}{k-n} \sum_{|\mathcal{I}_d|=k} u_{\mathcal{I}_d}^f, \tag{11}$$

with $u_{\mathcal{I}_d}^f$ being the solution of the problem on a grid with multi-index $\mathcal{I}_d$ such that $|\mathcal{I}_d|$ equals k. For sufficiently smooth functions, the sparse grid solution (for most practical purposes) can be used instead of the full grid solution. For a simple two-dimensional case, the subgrids (as constructed by the sparse-grid scheme) are depicted in figure 1(a)–(g). Note that the *shape of the stretch* in all these grids is different, which implies that in each of these subproblems we have a different grid induced anisotropy. If the subgrids are simply combined without any interpolation, which means that all the evaluated points in every subgrid are added with the binomial coefficients (11).

The number of points in the full grid with $n_i = n$ reads $N_f = (2^n)^d$.

From equation (11) it follows that the number of problems to be solved in the sparse grid technique reads

$$Z_{n,d} = \sum_{k=n}^{n+d-1} \binom{k-1}{d-1} = \frac{n}{d} \binom{n+d-1}{d-1} - \frac{n-d}{d} \binom{n-1}{d-1}. \tag{12}$$

Furthermore the number of points N_n employed in a grid with $|\mathcal{I}_d| = n$ reads

$$N_n = 2^n. \tag{13}$$

Combining (12) and (13) results in the total number of points employed in the sparse grid technique

$$N_n^c = \sum_{k=n}^{n+d-1} N_k \binom{k-1}{d-1} = \sum_{k=n}^{n+d-1} \binom{k-1}{d-1} 2^k. \tag{14}$$

It is known that the error of the discrete solution from a second order finite difference discretization of the 2D Laplacian can be split [15] as

$$u_n^f - u_E = C_1(x_1, h_1)h_1^2 + C_1(x_2, h_2)h_2^2 + D(x_1, h_1, x_2, h_2)h_1^2 h_2^2. \tag{15}$$

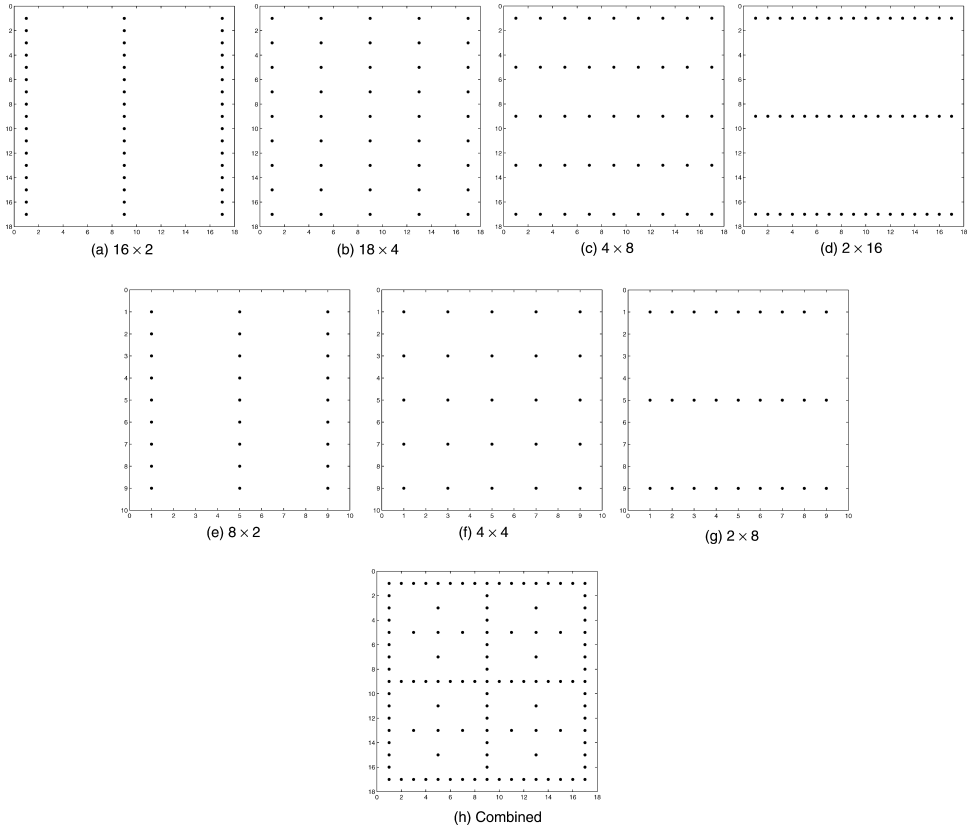

Figure 1. Construction of a 2D sparse grid; (a)–(d): grids on layer 5, (e)–(g): grids on layer 4; (h) combined sparse grid solution.

With the combination technique as in Definition 3.2 and the splitting in (15), the dimension-dependent absolute error (for the Laplacian), reads, [16]:

$$\epsilon_n = |u_n^c - u_E| = \mathbf{O}(h_n^2(\log_2 h_n^{-1})^{d-1}), \tag{16}$$

with h_n the finest mesh-size along one dimension.

4. The *d*-multigrid preconditioner

We employ multigrid as a preconditioner for Bi-CGSTAB due to the robustness that the resulting solver possesses. In this section we explore the preconditioner and browse through its various components. *Coarse grid correction* and *error smoothing* are two essential components of any multigrid algorithm. For anisotropic PDE it is well known that full coarsening along with point smoothing does not work, as it does not sufficiently smooth the errors that have to be approximated on the coarse grid. To address anisotropy, the standard way in two dimensions is either to coarsen along only one dimension (the one where error components are strongly coupled) or else to do a full coarsening but to resort to *line-relaxation* along the strongly coupled dimension [17].

These techniques can be extended to arbitrary higher d. A relaxation method based on *hyperplane* relaxation has been proposed in [18], which is analogous to *line-relaxation* in

two dimensions. Contrary to that approach, we proposed a method based on *point smoothing* and *partial coarsening* schemes in [19]. There we treat discrete anisotropies induced by non-equidistant grids. The proposal is to employ partial coarsening so that coarsening only takes place along those directions that have a strong coupling. We extend this technique to the general situation, where anisotropies may stem from two different sources, viz; the mesh size and the presence of constant anisotropic coefficients in the continuous problem.

An advantage of the point-wise smoothing method combined with the partial coarsening technique adopted here is that each problem in the sparse grids setting gets an individual treatment. The coarsening is different for each type of anisotropy considered. This may not be the case for a multigrid method with a fixed coarsening and hyperplane smoothing strategy. Furthermore, a multidimensional multigrid method based on a parallel point-wise Gauss–Seidel smoothing method can be parallelized. Although the paralellization of a multidimensional algorithm is not at all a trivial exercise.

4.1 *The relaxation method*

Point-wise *Red–Black Gauss–Seidel* (GSRB) is a standard method used for relaxation in the context of multigrid [17, 20]. We employ this in our multigrid preconditioner due to its excellent smoothing properties for elliptic equations. The first step in this process is the partitioning of the grid $\boldsymbol{G}$ into the *red* part ($\boldsymbol{G}_R$), and the *black* part ($\boldsymbol{G}_B$). Once this partition has been obtained, GSRB for a d-dimensional setting is no different from its two-dimensional counterpart. Let the index set containing the grid-points in $\boldsymbol{G}_R$ be represented by $\mathcal{I}_R$ and that containing the grid-points in $\boldsymbol{G}_B$ be represented by $\mathcal{I}_B$. In the standard literature [17, 20], the term *Red* refers to odd and *Black* to even points. The distinction of even and odd for a grid-point $(j_d, j_{(d-1)}, \ldots, j_1)$ is based on the following rule:

$$\text{Even if : } \mod\left(\sum_{i=1}^{d} j_i, 2\right) = 0; \quad \text{Odd if : } \mod\left(\sum_{i=1}^{d} j_i, 2\right) = 1;$$

From an implementational aspect, it is more convenient to adjust this definition, fixing *Red* as the category of the first unknown in the grid, which toggles between even and odd with the increase in d (for Dirichlet type of boundaries). Thus, we assert that we carry out a *Red-Black Gauss-Seidel* for all d, even if from the point of view of the above definition, we do an *even–odd* in two dimensions, *odd–even* in three, and so on.

4.2 *Coarsening strategies to handle anisotropies*

We use two grid coarsening strategies based on partial grid transfers to handle the anisotropies in the discretized system. Anisotropies can appear in the discrete system either through the presence of constant anisotropic coefficients in the continuous problem or due to unequal mesh sizes along different dimensions of the domain. These causative factors lead to a single coefficient ϵ_i for each dimension, viz $\epsilon_i = c_i/h_i^2$ with $h_i = (b_i - a_i)/N_i$. See (4). Fourier analysis suggests that all coefficients within a factor $< \pm\, 1.3$ of the maximum coefficient be considered equivalent for the purpose of partial doubling, i.e. the grid may be halved all along such dimensions *together*. However, as we experiment both with *doubling* ($h \to 2h$) as well as with *quadrupling* ($h \to 4h$) partial transfers; we would like to point out that for partial quadrupling the rule is much more strict. There we pick the maximum coefficient and only do a quadrupling transfer along the dimensions having a coefficient equal to this max. Whenever we require full coarsening, we always resort to full doubling as full quadrupling hampers multigrid convergence.

The application problem under the sparse grid solution method gives – in particular – subproblems where anisotropies originate from the unequal mesh-sizes along the dimensions. For clarity, we illustrate both the doubling and the quadrupling based coarsening strategies through the following simple example; where the anisotropy is only grid-based.

Example In the particular case when anisotropy sprouts only from discretization on non-equidistant grids (say, during the sparse grid solution of a high-dimensional problem), the mesh-aspect-ratios between various dimensions are powers of 2. This reduces the coarsening strategy (explained above) to the more simple case illustrated here for a five-dimensional problem discretized on the non-equidistant grid given by $\boldsymbol{N} = \begin{bmatrix}128 & 4 & 16 & 16 & 64\end{bmatrix}$. $c_i = c$ for $i = \{1, 2, \ldots, d\}$ and $\Omega = [a, b]^d$. Then the grid is coarsened in the following way;

Strategy 1, doubling (h $\longrightarrow$ 2h)	Strategy 2, quadrupling (h $\longrightarrow$ 4h)
$\Omega_6 = [128 \quad 4 \quad 16 \quad 16 \quad 64]$	
$\Omega_5 = [64 \quad 4 \quad 16 \quad 16 \quad 64]$	$\Omega_4 = [128 \quad 4 \quad 16 \quad 16 \quad 64]$
$\Omega_4 = [32 \quad 4 \quad 16 \quad 16 \quad 32]$	$\Omega_3 = [64 \quad 4 \quad 16 \quad 16 \quad 64]$
$\Omega_3 = [16 \quad 4 \quad 16 \quad 16 \quad 16]$	$\Omega_2 = [16 \quad 4 \quad 16 \quad 16 \quad 16]$
$\Omega_2 = [8 \quad 4 \quad 8 \quad 8 \quad 8]$	$\Omega_1 = [4 \quad 4 \quad 4 \quad 4 \quad 4]$
$\Omega_1 = [4 \quad 4 \quad 4 \quad 4 \quad 4]$	$\Omega_0 = [2 \quad 2 \quad 2 \quad 2 \quad 2]$
$\Omega_0 = [2 \quad 2 \quad 2 \quad 2 \quad 2]$	

(17)

4.3 *Coarse-grid discretization*

An important component in the coarse grid correction process is the choice of the coarse-grid operator L_H. We use the coarse-grid analog of the discrete operator on the fine-grid. Once the next coarser-grid is decided we build the operator using the same scheme as in section 2.

Another option is to use the Galerkin operator. Some especial transfer operators (in one and two dimensions) can be employed to generate a relatively sparse Galerkin operator [20] but as of yet it is unknown how this kind of transfer might be extended to abstract d dimensions. A significant disadvantage of employing the Galerkin operator (constructed with the usual transfer operators) is its being much more dense than the coarse-grid analog of the fine-grid operator. This issue becomes more serious with increasing d.

4.4 *The transfer operators*

We employ the d-dimensional analogues of the full-weighting (FW) restriction operator and of the bilinear interpolation operator in two dimensions for the intergrid transfers of the grid functions. In this section we present a tensor formulation to generate the restriction and prolongation operator matrices. For completeness we first mention [17] that a $2d$ FW restriction operator is the Kronecker tensor product of the following x_1 and x_2 directional one-dimensional FW operators:

$$(I_h^{2h})_{x_1} \triangleq \frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix}, \quad (I_h^{2h})_{x_2} \triangleq \frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix}.$$

The following formula – based on Kronecker tensor products – gives a FW restriction operator matrix $\mathbf{R}$ (for the non-eliminated boundary scheme). It unifies doubling and quadrupling

transfers into the following compact form:

$$\mathbf{R} = \prod_{i=1}^{d} (\mathbf{R}_i)^{t_i},$$

$$(\mathbf{R}_i)^{t_i} = \prod_{l=0}^{t_i-1} \left[\bigotimes_{j=i}^{d-1} \mathbf{I}_{N_{(d+i-j)}} \otimes \mathbf{O}_{[N_i/2^{(t_i-l-1)}]} \otimes \bigotimes_{j=1}^{i-1} \mathbf{I}_{[N_{(i-j)}/2^{t_{(i-j)}}]} \right]. \tag{18}$$

We now define the quantities involved in (18) for the dummy subscript a. $\mathbf{I}_a$ is a diagonal matrix of order $(a+1) \times (a+1)$. The diagonal entry is 1, except at the first and the last positions where it is 0. $\mathbf{O}_a$ is the $1d$ FW restriction operator matrix, order = $(a/2+1) \times (a+1)$, obtained by applying the $1d$ transfer stencil at the interior points and taking 0 at the boundaries. $\boldsymbol{N} = [N_1, N_2, \ldots, N_d]$, is the grid description. $\boldsymbol{T} = [t_1, t_2, \ldots, t_d]$ is the coarsening request, t_i is the count of $(h \to 2h)$ transfers along the ith dimension. *We say that quadrupling takes place along the ith dimension if $t_i = 2$.* It is trivial to verify this formula with a matrix manipulation software package. Once the FW restriction operator matrix in d dimensions is set, the prolongation (d-linear interpolation) operator matrix can be obtained by the following relation:

$$\mathbf{P} = 2^{\sum_{i=1}^{d} t_i} (\mathbf{R}^T). \tag{19}$$

$\mathbf{R}^T$ is the transpose of the restriction operator matrix $\mathbf{R}$. With this general transfer operator we can experiment with different transfers based on doubling and quadrupling, depending on the anisotropy of the discrete system. This FW restriction operator provides the required matrix for any number of coarsenings along any number of dimensions for an abstract d-dimensional problem. The stage is all set now to experiment with the preconditioner and check out its utility and efficiency.

4.5 *The multigrid algorithm*

In the set-up phase a coarsening strategy is determined based on the coefficients ϵ_i, $i = 1, \ldots, d$ (see section 4.3). With the coarsening strategy, the discrete coarse grid problems are defined, as in Example (17). The set-up stage ends with the construction of the transfer operators between the various coarse grids, as in (18), (19). They are also dictated by the coarse grids. After this the multigrid iteration starts.

We present the algorithm slightly different from the basic multigrid literature [17]. The V, W and F cycle-types are included in a unified algorithm. In the following, *cycle* is the cycle-type, and the basic pseudo-instructions appear underscored. $\frac{1}{2}$ in the superscript implies *half way through the cycle* or the state *just after coarse-grid-correction*; l is the grid level indicator. A lower value of l implies a coarser-grid.

Multigrid pseudocode $u_l^{m+1} = \mathbf{MG}(l, \gamma, cycle, u_l^m, \boldsymbol{A}_l, b_l, \nu_1, \nu_2)$

(0) Initialization
- If $l = 1$, $u_l^{m+1} =$ **exact** $(\boldsymbol{A}_l, b_l)$; Bail out; endif
- Build the coarse-grid operator $\boldsymbol{A}_{l-1}$, and the transfer operators $\boldsymbol{R}$, and $\boldsymbol{P}$

(1) Pre–smoothing
- Compute $\bar{u}_l^m$ by applying $\nu_1 (\geq 0)$ smoothing steps to $u_l^m : \bar{u}_l^m =$ **smooth**$^{\nu_1}$ $(u_l^m, \boldsymbol{A}_l, b_l)$.

(2) Coarse grid correction
- Compute the defect $\bar{r}_l^m = b_l - \boldsymbol{A}_l \bar{u}_l^m$
- Restrict the defect $\bar{r}_{l-1}^m = \boldsymbol{R} \bar{r}_l^m$

– Compute the approximate error $\hat{e}_{l-1}^{m}$ from the defect equation. $\boldsymbol{A}_{l-1}\ \hat{e}_{l-1}^{m} = \bar{r}_{l-1}^{m}$

by the following mini algorithm

If $l = 1$, $\hat{e}_{l-1}^{m} =$ ***exact*** $(\boldsymbol{A}_{l-1}, \bar{r}_{l-1}^{m})$; *endif*

If $l > 1$, *solve for* $\hat{e}_{l-1}^{m}$ *approximately by the recursion:*

$\hat{e}_{l-1}^{m,1} = \bar{0}$;

do $i = 1$ *to* γ

If $cycle = f$ *and* $i \neq 1$,

$\hat{e}_{l-1}^{m,i+1} = \boldsymbol{MG}(l-1, 1, cycle, \hat{e}_{l-1}^{m,i}, \boldsymbol{A}_{l-1}, \bar{r}_{l-1}^{m}, \nu_1, \nu_2)$

else

$\hat{e}_{l-1}^{m,i+1} = \boldsymbol{MG}(l-1, \gamma, cycle, \hat{e}_{l-1}^{m,i}, \boldsymbol{A}_{l-1}, \bar{r}_{l-1}^{m}, \nu_1, \nu_2)$

endif

continue i

endif

– Interpolate the correction $\hat{e}_{l}^{m} = \boldsymbol{P}\ \ \hat{e}_{l-1}^{m}$.

– Compute the corrected approximation on $\Omega_l\ u_l^{m+\frac{1}{2}} = \bar{u}_l^{m} + \hat{e}_l^{m}$.

(3) Post–smoothing

– Compute u_l^{m+1} by applying $\nu_2\ (\geq 0)$ smoothing steps to $u_l^{m+\frac{1}{2}}$: $u_l^{m+1} = \mathbf{smooth}^{\nu_2}(u_l^{m+\frac{1}{2}}, \boldsymbol{A}_l, b_l)$.

5. Numerical experiments based on the *full-grid solution* method

In this work *full-grid solution* refers to a solution on a regular tensor-product grid (where the sparse grid technique is not used). As described in the previous section, we have quite a strong and robust multigrid preconditioner. Before we actually use it in the sparse-grid setting, we would like to test its performance as a stand-alone solver versus as a preconditioner for Bi-CGSTAB in a full-grid solution process.

5.1 *d-multigrid performance in stationary cases*

A useful numerical insight for time-marching solution processes (our ultimate aim in this work) comes from an insight into the stationary process per time-step. Through the numerical

solution of the PDE we approximate the following test function:

$$u(\boldsymbol{x}) = \frac{\sum_{i=1}^{d} \sin(d\pi^2 x_i)}{d\pi + \sum_{i=1}^{d} x_i}. \tag{20}$$

We have conducted a number of numerical experiments -isotropic and anisotropic- and have included the convergence graphs for them. These graphs show the residual reduction against iteration and cpu time for d-multigrid used in these two contexts (solver and preconditioner). A word of caution while examining these graphs is just in place. Multigrid is an $O(M)$ solver (where M is the number of unknowns on the finest grid) when optimal relaxation and ideal coarse grid correction are available. In such a situation multigrid is extremely efficient. Some of the graphs here show a tough competition between multigrid as a solver against multigrid as a Bi-CGSTAB preconditioner. This happens due to the fact that for the model-problem the employed relaxation method and the coarse grid correction form near-optimal d-multigrid attributes. Evidently, in any situation where tuning multigrid with optimal attributes is not a choice, multigrid works better as a Krylov preconditioner than as a stand-alone solver.

First of all, we check out d-multigrid performance for a five-dimensional isotropic case, with 32 divisions along all dimensions of the domain. The number of unknowns in the system is 39, 135, 393. In this case the V(1, 1) multigrid (multigrid method based on V cycles with

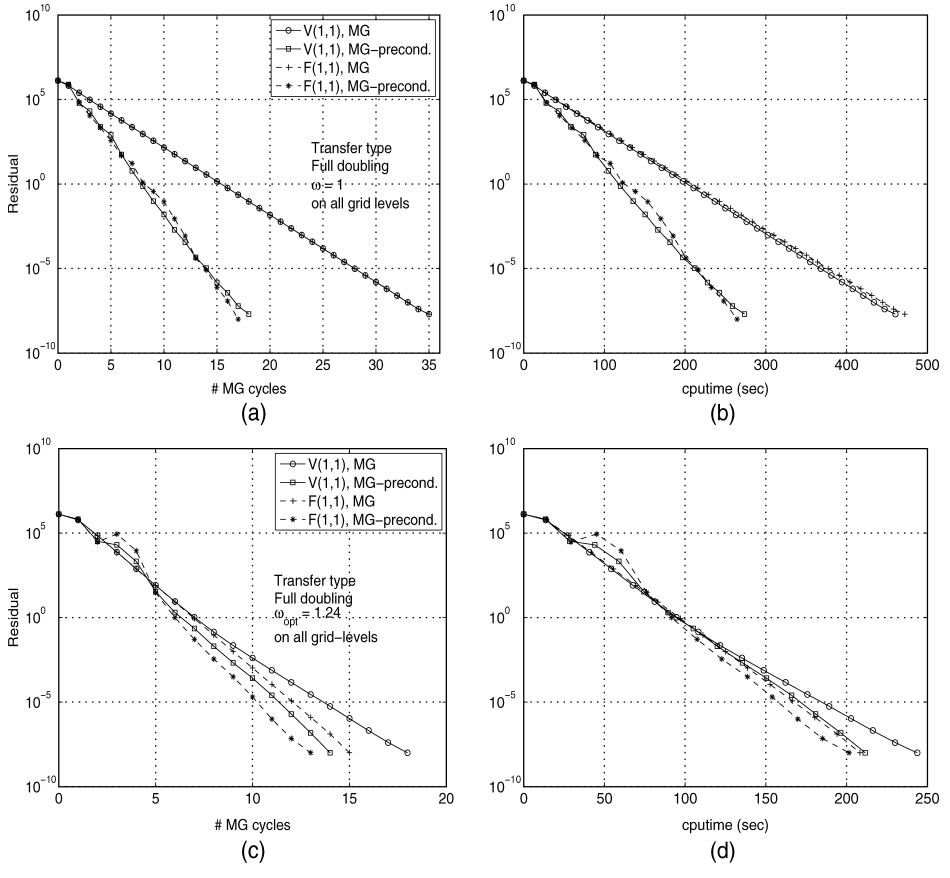

Figure 2. Convergence diagram for a five-dimensional isotropic problem, 32 divisions along each dimension.

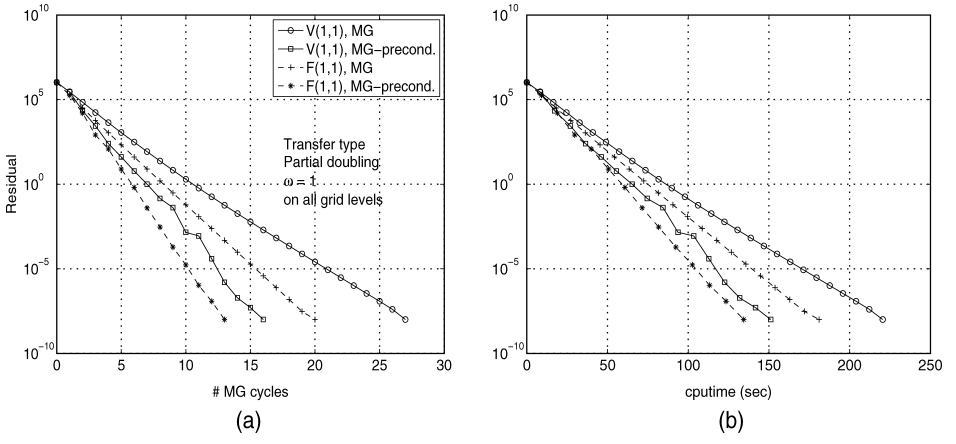

Figure 3. Convergence diagram for a six-dimensional anisotropic problem, grid stretched along $d/2$ dimensions and given by $\boldsymbol{N} = [32, 8, 32, 8, 32, 8]$. Number of unknowns is 26,198,073. (a) shows a comparison between multigrid as a solver and multigrid as a preconditioner, on the iteration scale. (b) on the cpu time scale.

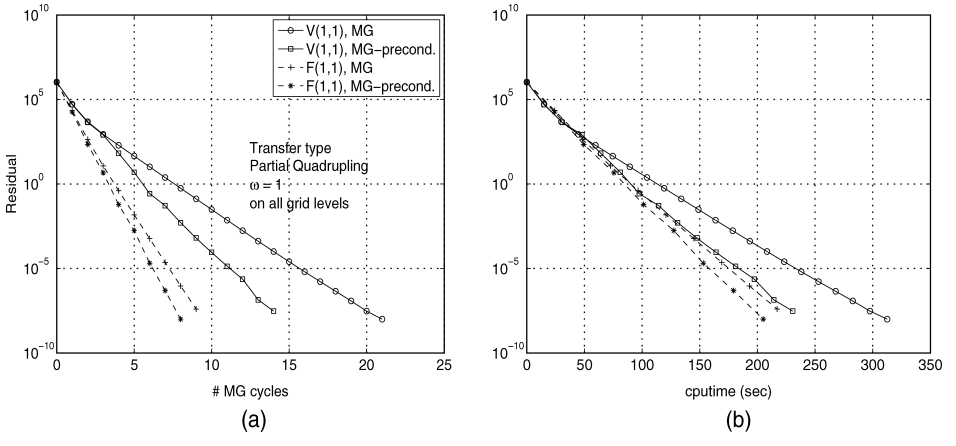

Figure 4. Convergence diagram for a seven-dimensional anisotropic problem, grid stretched along one dimension, and given by $\boldsymbol{N} = [8, 8, 8, 64, 8, 8, 8]$, Number of unknowns is 34,543,665.

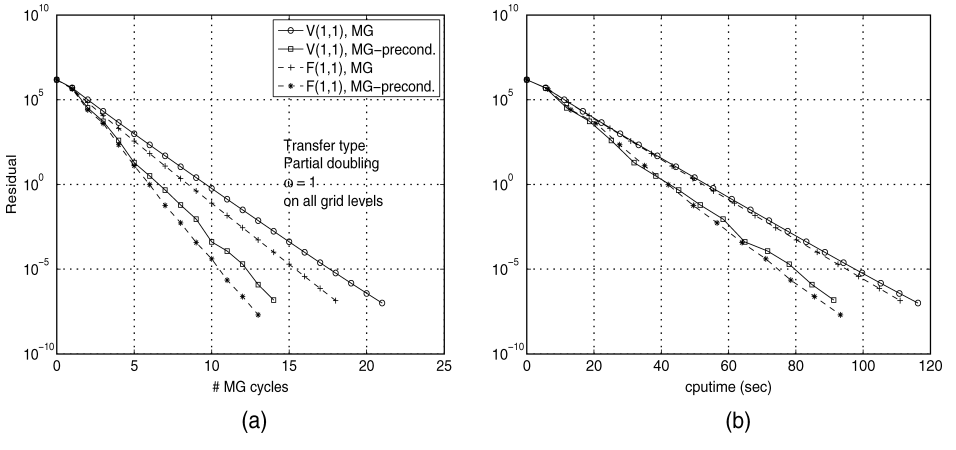

Figure 5. Convergence diagram for a four-dimensional anisotropic problem, grid stretched along $(d-1)$ dimensions, and given by $\boldsymbol{N} = [128, 128, 128, 8]$, Number of unknowns is 19,320,201.

one pre- and one post-smoothing steps) preconditioned Bi-CGSTAB far out-performs the V(1,1) multigrid solver; figure 2(a) and (b) (here we choose $\omega = 1$ in ω-GSRB). However, with $\omega_{\text{opt}} = 1.24$ included in the game (a possibility for the model problem) the comparison is not as bright; figure 2(c) and (d). This confirms that no great Krylov induced enhancement should be expected when multigrid (as a solver) approaches optimality.

Next we present some experiments based on problems with *discrete anisotropies* that result from discretization on a non-equidistant grid, i.e. a grid where the number of divisions is different along different dimensions of the hyper domain. We have selected three high-dimensional problems, each with a different discrete anisotropy. The problems have been chosen with the aim of harvesting experimental results for grids highly stretched along one dimension as well as grids highly stretched along multiple dimensions. The anisotropies are handled with the partial coarsening schemes as illustrated by the example in section 4.2. The results appear in figures 3–5. Here, we find that the F(1, 1) MG cycles are more suited than V(1, 1), both for the stand-alone multigrid solver as well as a preconditioner for Bi-CGSTAB, if the coarsening strategy is based on doubling. Quadrupling suits the situation more when the grid is stretched along only a few dimensions, (preferably $< d/2$) and when optimal relaxation parameters are available. However, with quadrupling, V(2, 2) and F(1, 1) cycles seem to yield better results than V(1, 1).

In [19] we pointed out that for grids stretched along a single dimension, a method based on partial quadrupling as the coarsening strategy has an $O(M)$ complexity even with W cycles; achieving this is not possible with methods based on partial doubling along one dimension. This fact makes it a strategy of choice for such grids, figure 6. We have only chosen the cpu time scale (for presenting results) because with different transfer operators, the number of cycles are not comparable. Quadrupling -in contrast with doubling- relies on optimal relaxation to quite some extent; in fact, the better the relaxation process the shorter the cpu time. This points us to the fact that if optimization in the relaxation process is an impossibility we might be better off with doubling for all kinds of grid-based discrete anisotropies.

The multigrid convergence factors in all these experiments are quite low (around an average of 0.1), implying that a *full multigrid algorithm* starting on the coarsest grid is expected to reach an approximate solution up to the discretization accuracy in just one or two cycles.

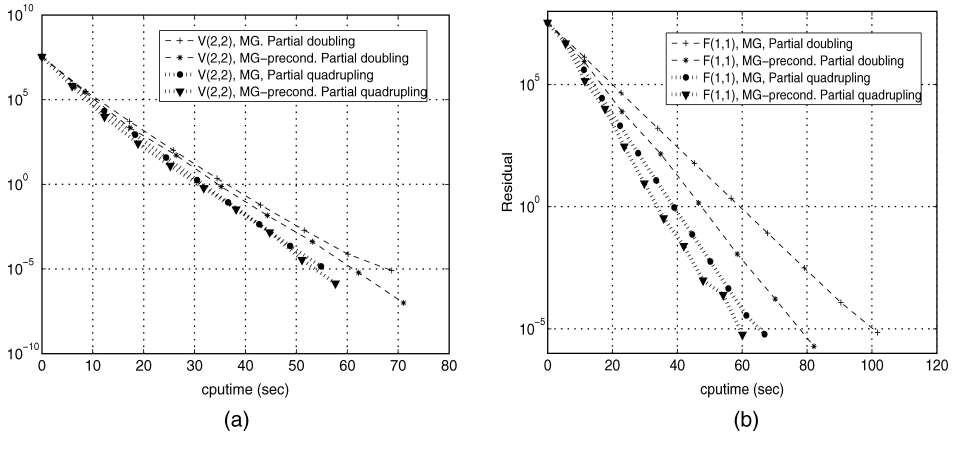

Figure 6. Convergence diagrams comparing d-multigrid based on doubling transfers against quadrupling transfers for (a) V(2,2), and (b) F(1,1) multigrid cycles. This five-dimensional problem is discretized on $\boldsymbol{N} = [8, 8, 2048, 8, 8]$, Number of unknowns is 13,443,489.

6. Numerical experiments based on the *sparse-grid solution*

6.1 *d-multigrid as a preconditioner in the time-independent case*

As the solver works for every kind of grid that might arise within the sparse grid setting (as described in section 4), the aim is now to reach a reasonable number of dimensions. The test function for the sparse grid stationary experiment reads:

$$u(\boldsymbol{x}) = \prod_{i=1}^{d} e^{x_i^2} = \exp\left(\sum_{i=1}^{d} x_i^2\right), \tag{21}$$

with $\Omega = [0, 1]^d$, $c_i = 1$. The approximation is done for $2 \leqslant d \leqslant 8$ with a mimic of the grid with 1024 cells per coordinate. In a full grid setting the maximum problem would have a size of $1024^8 = 2^{80}$, i.e., 2^{53} GB of memory, which is – of course – immensely huge. The maximum size of an 8D problem in the sparse grid setting chosen here is $1024 \times 2^7 = 2^{17}$, which is only 1 MB. The solution at the central point in the domain $x_i = 0.5$ is computed here. The results are described in detail for $d = 2$ and $d = 8$ in table 1 and the error and convergence for all values of d are plotted in figure 7. In the table, the time indicated is the total computational time for the sparse grid solution including the interpolation. The number of problems #*probl* is as in equation (12) and the theoretical convergence *Th. Conv* from equation (16).

The table and figures show the dependence of the number of dimensions in the convergence according to the theoretical convergence ratio in equation (16). Although the theoretical convergence of the sparse grid method is low when d is high at small numbers of $n_{\max}$ (the largest number of cells in one direction), the convergence in this test experiment is reasonable. A possible reason may be the smoothness of the analytic solution.

Table 2 presents a comparison of the total number of multigrid cycles when multigrid is employed both as a solver as well as a preconditioner for Bi-CGSTAB for solving all sparse grid subproblems on a layer of a d-dimensional problem, with d increasing. Presented are the maximum, the minimum and the average numbers of iterations, as well as the total number of subproblems solved on which these numbers are based. The stopping criterion is the residual

Table 1. Time-independent experiments of problem (21) using sparse grids. *Top*: two-dimensional case. *Bottom*: eight-dimensional case. Column one gives $n_{\max}$, the largest number of cells in one coordinate.

$n_{\max}$	Value	Error	Conv.	Time	#probl.	Th. Conv.
			$d = 2$			
16	1.65	$5.52 \cdot 10^{-3}$	3.05	0.04	7	3.00
32	1.65	$1.72 \cdot 10^{-3}$	3.21	0.07	9	3.20
64	1.65	$5.16 \cdot 10^{-4}$	3.34	0.10	11	3.33
128	1.65	$1.50 \cdot 10^{-4}$	3.43	0.12	13	3.43
256	1.65	$4.30 \cdot 10^{-5}$	3.50	0.16	15	3.50
512	1.65	$1.21 \cdot 10^{-5}$	3.55	0.25	17	3.56
1024	1.65	$3.36 \cdot 10^{-6}$	3.60	0.33	19	3.60
			$d = 8$			
16	7.63	$2.43 \cdot 10^{-1}$	1.50	18.51	165	0.53
32	7.54	$1.48 \cdot 10^{-1}$	1.64	111.07	495	0.84
64	7.47	$8.33 \cdot 10^{-2}$	1.77	578.55	1287	1.12
128	7.43	$4.40 \cdot 10^{-2}$	1.89	2404.47	3003	1.36
256	7.41	$2.20 \cdot 10^{-2}$	2.00	9067.30	6435	1.57
512	7.40	$1.04 \cdot 10^{-2}$	2.10	32925.66	12869	1.75
1024	7.39	$4.76 \cdot 10^{-3}$	2.19	106826.59	24301	1.91

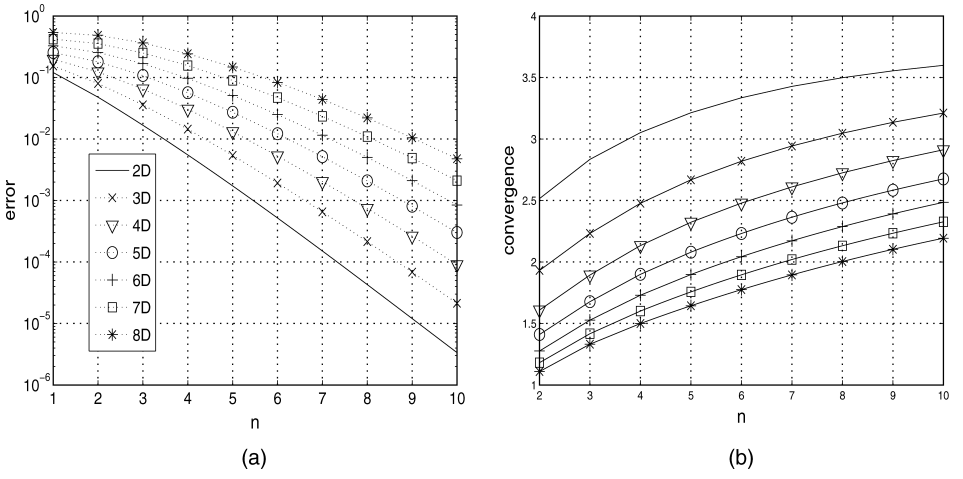

Figure 7. (a) Decay of the error $|u_n^c - u_E|$, with u_E the exact solution (21). (b) Convergence of the error in the left picture.

Table 2. Comparison of the total number of pure multigrid and multigrid preconditioned Bi-CGSTAB iterations (maximum, minimum and average number), for all the subgrids on the finest layer of a d-dimensional problem, for increasing d, # grids is the number of subgrids solved.

d	Max	Min	Average	# grids
		Bi-CGSTAB with MG		
2	12.0	4.0	9.400	10
3	14.0	6.0	10.691	55
4	14.0	6.0	10.982	220
5	16.0	6.0	10.999	715
6	16.0	6.0	10.928	2002
7	16.0	6.0	10.791	5005
8	14.0	6.0	10.488	11440
		Pure multigrid		
2	13.0	5.0	10.000	10
3	20.0	6.0	12.436	55
4	21.0	7.0	13.155	220
5	24.0	8.0	13.221	715
6	22.0	8.0	13.185	2002
7	21.0	8.0	12.997	5005
8	20.0	8.0	12.762	11440

being smaller than 10^{-14}, which is severe but gives a good insight in the comparison. For both solvers we choose smoothing relaxation parameter $\omega = 1$. We see that the average number of multigrid cycles – when multigrid is used as a preconditioner instead of being used as a solver – does not reduce significantly for low values of d. However the cycle difference in the two usages do become significant for higher d because then the total number of subproblems also increase binomially. As the number of subproblems to be solved is more than 5000 for the 7D problem a gain in average of about two multigrid iterations is still interesting.

The time for the highest level in the case $d = 8$ over 10^5 seconds which is 28 hours. However, the number of subproblems is 24300, so the average computational time per grid is only 5 seconds. If the sparse grid method is parallelized over 10 machines, the time would be

2.8 hours in total, because the sparse grid is only a combination technique of subproblems. Parallelization is a future task in our project.

6.2 *d-multigrid as a preconditioner in the time-dependent case*

For the time-dependent case, we choose as the test problem solution,

$$u(\boldsymbol{x}, t) = e^t \prod_{i=1}^{d} \exp\left(\frac{x_i}{\sqrt{d}}\right) = \exp\left(t + \frac{1}{\sqrt{d}} \sum_{i=1}^{d} x_i\right), \tag{22}$$

with $\Omega = [0, 1]^d$, $t \geqslant 0$ and $c_i = 1$. The approximation is done for $2 \leqslant d \leqslant 5$ with 1024 cells as the maximum number per coordinate in the sparse grid technique. The solution of the central point in the domain $x_i = 0.5$ is computed at time $t = 0.1$. The number of time steps used is fixed at 400.

The *grid convergence* results are described in detail for $d = 2$ and $d = 5$ in table 3 and the errors and convergence for all values of d are plotted in figure 8. In the table, again 'time' is the total computation time for the complete sparse grid solution including the interpolation and time integration. The number of problems is as in (12) and the theoretical convergence from equation (16).

Again, the results in the table and figures shows the dependence of the number of dimensions in the error. The total time for the 5D computation is again relatively small per grid and the accuracy results are satisfactory for the time-dependent case. The multigrid convergence remains excellent, as in the stationary test case above.

Table 3. Time-dependent experiments with solution (22) using sparse grids. *Top*: two-dimensional case. *Bottom*: five-dimensional case. Column one gives the maximum number of cells per coordinate.

	$X = 0.5^d$			Control		
Grid	Value	Error	Conv.	Time	#probl.	Th. Conv.
			$d = 2$			
8	2.24	$1.37 \cdot 10^{-4}$	2.96	2.14	5	2.67
16	2.24	$4.36 \cdot 10^{-5}$	3.14	5.67	7	3.00
32	2.24	$1.33 \cdot 10^{-5}$	3.28	10.87	9	3.20
64	2.24	$3.93 \cdot 10^{-6}$	3.38	19.12	11	3.33
128	2.24	$1.14 \cdot 10^{-6}$	3.46	31.53	13	3.43
256	2.24	$3.23 \cdot 10^{-7}$	3.52	49.01	15	3.50
512	2.24	$9.07 \cdot 10^{-8}$	3.56	70.35	17	3.56
1024	2.24	$2.56 \cdot 10^{-8}$	3.54	99.82	19	3.60
			$d = 5$			
8	3.38	$9.67 \cdot 10^{-5}$	1.98	9.02	21	0.79
16	3.38	$4.45 \cdot 10^{-5}$	2.17	36.99	56	1.27
32	3.38	$1.92 \cdot 10^{-5}$	2.32	129.77	126	1.64
64	3.38	$7.83 \cdot 10^{-6}$	2.45	436.85	251	1.93
128	3.38	$3.06 \cdot 10^{-6}$	2.56	1288.44	456	2.16
256	3.38	$1.15 \cdot 10^{-6}$	2.65	3658.62	771	2.34
512	3.38	$4.24 \cdot 10^{-7}$	2.72	11297.58	1231	2.50
1024	3.38	$1.50 \cdot 10^{-7}$	2.83	32266.03	1876	2.62

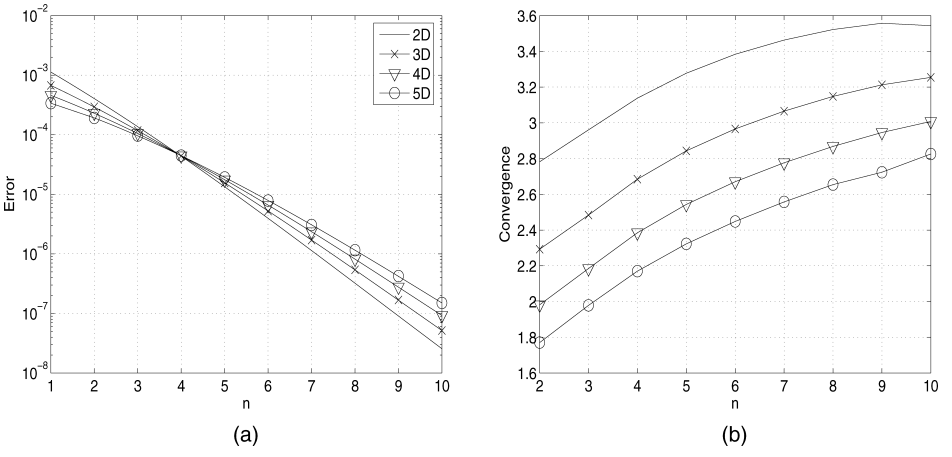

Figure 8. (a) Decay of the error $|u_n^c - u_E|$, u_E from (22). (b) Convergence of the error in the left picture.

7. Multi-asset option

The last experiment is the computation of the price of a basket option by solving equation (3) with the sparse grid technique. The corresponding initial condition reads:

$$u(\boldsymbol{x}, 0) = \max\left(\sum_{i=1}^{d} w_i e^{\sigma_i x_i} - K, 0\right), \tag{23}$$

with w_i percentages of the assets in the underlying basket, σ_i the volatility in asset i and K the exercise price. This payoff function has a non-differentiability in the hyperplane when the basket sum equals the strike price. This will be problematic for the sparse grid solution of this problem, as one requirement for sparse grid convergence is that numerical solutions have bounded mixed derivatives [7]. Still we are interested in the convergence of sparse grids for this option pricing problem.

The remaining problem parameters are set for each asset as

- $w_i = 1/d$, $1 \leq i \leq d$
- $K =$euros 40
- $r = 6\%$
- $T =$ one year
- $\sigma_i = 20\%$, $1 \leq i \leq d$
- $\delta_i = 4\%$, $1 \leq i \leq d$
- $\rho_{ij} = 0.25$, $i \neq j$.

The price of the option is computed for $2 \leqslant d \leqslant 5$ where d represents the number of assets in the basket. The outer domain boundaries are placed at $S = 5K$ to mimic infinity in (1). In the x-domain, this means that $\Omega = [-\sigma_i^{-1} \log 5, \sigma_i^{-1} \log 5]^d$. The sparse grid approximation contains grids with at maximum 1024 cells in a coordinate and with 128 time steps. The results of the experiments are summarized in table 4.

In the table, satisfactory grid convergence is observed for the lower dimensional cases, but it is no longer regular. In particular when d is increasing, the convergence becomes irregular. The reason may lie in the fact that in higher dimensions a large number of subgrids is included with only a very small number of grid points in many dimensions ($n_i = 2$ in this test). An alternative is to use the sparse grid technique based on a larger number of points in each

Table 4. Option prices of basket calls. *Top*: Sparse grid option prices for $d = 2$ and $d = 3$. *Bottom*: Option prices for the higher dimensions. n represents the maximum number of point in one dimension.

	$d = 2$		$d = 3$	
n	Value	Error	Value	Error
8	2.291	$5.11 \cdot 10^{-1}$	2.102	$4.51 \cdot 10^{-1}$
16	2.727	$7.60 \cdot 10^{-2}$	2.498	$5.46 \cdot 10^{-2}$
32	2.801	$1.10 \cdot 10^{-3}$	2.562	$1.00 \cdot 10^{-2}$
64	2.807	$4.23 \cdot 10^{-3}$	2.563	$9.04 \cdot 10^{-3}$
128	2.811	$8.56 \cdot 10^{-3}$	2.562	$9.87 \cdot 10^{-3}$
256	2.807	$4.65 \cdot 10^{-3}$	2.555	$3.28 \cdot 10^{-3}$
512	2.803	$7.72 \cdot 10^{-4}$	2.554	$1.39 \cdot 10^{-3}$
1024	2.803	$6.52 \cdot 10^{-4}$	2.553	$3.61 \cdot 10^{-4}$
	$d = 4$		$d = 5$	
8	1.983	$4.32 \cdot 10^{-1}$	1.896	$4.32 \cdot 10^{-1}$
16	2.364	$5.05 \cdot 10^{-2}$	2.273	$5.42 \cdot 10^{-2}$
32	2.429	$1.45 \cdot 10^{-2}$	2.343	$1.57 \cdot 10^{-2}$
64	2.427	$1.19 \cdot 10^{-2}$	2.341	$1.36 \cdot 10^{-2}$
128	2.422	$7.02 \cdot 10^{-3}$	2.331	$3.97 \cdot 10^{-3}$
256	2.420	$5.35 \cdot 10^{-3}$	2.335	$7.88 \cdot 10^{-3}$
512	2.417	$2.69 \cdot 10^{-3}$	2.330	$2.28 \cdot 10^{-3}$
1024	2.413	$1.29 \cdot 10^{-3}$	2.326	$1.77 \cdot 10^{-3}$

dimension (n_i at least 4 or 8) see, for example, [21]. Furthermore, the accuracy is hampered by the fact that the initial condition is non-differentiable.

The multigrid convergence, however, remains excellent, of course.

8. Conclusions

A promising technique to handle high-dimensional PDE problems numerically is the sparse grid method, which gives rise to an abundant number of smaller sized problems on equidistant and non-equidistant grids. The non-equidistant grids generate a discrete anisotropy in the system. In the context of developing a d-multigrid method for these problems we have shown in this paper how these anisotropies can be treated by a suitable combination of partial coarsening strategies and point based relaxation. The partial coarsening schemes are based on doubling and quadrupling transfers. It turns out that for isotropic systems, an F(1, 1) cycle with/without an optimal relaxation parameters is a very nice combination for a multigrid method. For anisotropic problems, a V(1, 1) method without an optimal parameter proves excellent. A speed-up can be had by employing partial quadrupling as the coarsening strategy for anisotropic problems provided that optimal relaxation parameters are accessible. For partial quadrupling, the V(2, 2) and F(1, 1) cycles with optimal relaxation parameters seem good.

Although it is well known that multigrid methods are amongst the fastest solvers for elliptic equations, the throughput of a multigrid solver usually depends on how best it could be tuned with optimal attributes which include optimal relaxation and an ideal coarse grid correction. It is often difficult to reach this optimality in a practical situation. To some extent this could be substituted by having multigrid as a preconditioner of a suitable Krylov subspace method, such as Bi-CGSTAB. We have shown in this paper how such a d-multigrid method may be set up and employed as a preconditioner, and supplemented the development with numerical experiments and convergence diagrams.

The resulting solver is quite robust and generally applicable to a wide class of discrete parabolic and elliptic problems. We demonstrate its utility by solving the Black–Scholes equation of pricing options dependent on multiple underlying assets. However, we also show that the *grid convergence* of the sparse grid solution is irregular due to a non-differentiable payoff. This needs to be analysed in more detail. Further, we also plan to exploit the parallel features of this solver by automating this parallelism.

Acknowledgements

This research has been partially supported by the Dutch government through the national program BSIK: knowledge and research capacity, in the ICT project BRICKS (http://www.bsik-bricks.nl), theme MSV1, partially by the Government of Pakistan through The HEC-Pakistan research grant, contract Ref: 1-3/PM-OVER/Neth/SPMU/2004 and partially by the Dutch Technology foundation STW. We would like to express our thanks to these sponsors.

References

[1] You-lan, Z., Xiaonan, W. and I-Liang, C., 2004, *Derivative Securities and Difference Methods* (New York: Springer-Verlag).
[2] Kwok, Y.K., 1998, *Mathematical Models of Financial Derivatives*, 2nd edn (Singapore: Springer-Verlag).
[3] Beylkin, G. and Martin, J.M., 2005, Algorithms for numerical analysis in high dimensions. *SIAM Journal on Scientific Computing*, **26**, 2133–2159.
[4] Yserentant, H., 2005, Sparse grid spaces for the numerical solution of the electronic Schrödinger equation. *Numerische Mathematik*, **101**, 381–389.
[5] Elf, J., Lötstedt, P. and Sjöberg, P., 2003, Problems of high dimension in molecular biology. In: W. Hackbusch (Ed), *Proceedings of the 19th GAMM Seminar*, Leipzig, pp. 21–30.
[6] Bellman, R., 1961, *Adaptive Control Processes: A Guided Tour* (Princeton: Princeton University Press).
[7] Bungartz, H.J. and Griebel, M., 2004, Sparse grids. *Acta Numerica*, May, pp. 147–269.
[8] Griebel, M., Schneider, M. and Zenger, C., 1992, A combination technique for the solution of sparse grid problems. In: P. de Groen and R. Beauwens (Eds), *Proceedings of the IMACS International Symposium on Iterative Methods in Linear Algebra*, pp. 263–281 (Amsterdam: Elsevier).
[9] Zenger, C., 1990, Sparse grids. In: W. Hackbusch (Ed), *Proceedings of the 6th GAMM Seminar; Notes on Numerical Fluid Mechanics*, volume 31, 241–251.
[10] van der Vorst, H.A., 1992, A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, **13**, 631–644.
[11] Oosterlee, C.W. and Washio, T., 1998, An evaluation of parallel multigrid as a solver and as a preconditioner for singularly perturbed problems. *SIAM Journal on Scientific Computing*, **19**, 87–110.
[12] Wienands, R., Oosterlee, C.W. and Washio, T., 2000, Fourier analysis of gmres(m) preconditioned by multigrid. *SIAM Journal on Scientific Computing*, **22**, 582–603.
[13] Erlangga, Y.A., Oosterlee, C.W. and Vuik, C., 2006, A novel multigrid based preconditioner for heterogeneous helmholtz problems. *SIAM Journal on Scientific Computing*, **27**, 1471–1492.
[14] Steeb, H.W., 1991, *Kronecker Product of Matrices and Applications* (Mannheim: Wissenschaftsverlag).
[15] Hofmann, P., 1967, Asymptotic expansions of the discretization error of boundary value problems of the laplace equation in rectangular domains. *Numerische Mathematik*, **9**, 302–322.
[16] Bungartz, H.J., Griebel, M., Röschke, D. and Zenger, C., 1994, Pointwise convergence of the combination technique for the Laplace equation. *East-West Journal of Numerical Mathematics*, **2**, 21–45.
[17] Trottenberg, U., Oosterlee, C.W. and Schüller, A., 2001, *Multigrid* (New York: Academic Press).
[18] Reisinger, C. and Wittum, G., 2006, Efficient hierarchical approximation of high-dimensional option pricing problems. Technical Report, University of Oxford.
[19] bin Zubair, H., Oosterlee, C.W. and Wienands, R., 2006, Multigrid for high dimensional elliptic partial differential equations on non-equidistant grids. Technical Report, Delft University of Technology.
[20] Stüben, K. and Trottenberg, U., 1982, *Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications* (Berlin: Springer).
[21] Leentvaar, C.C.W. and Oosterlee, C.W., 2006, On coordinate transformation and grid stretching for sparse grid pricing of basket options. Technical Report 06-13, Delft University of Technology.