INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 10, Number 2, Pages 374–388

APPLICATION OF AN ENERGY-MINIMIZING ALGEBRAIC MULTIGRID METHOD FOR SUBSURFACE WATER SIMULATIONS

JING-RU C. CHENG, XUE-HAI HUANG, SHI SHU, JINCHAO XU, CHEN-SONG ZHANG, SHUO ZHANG, AND ZHIYANG ZHOU

Abstract. Efficient methods for solving linear algebraic equations are crucial to creating fast and accurate numerical simulations in many applications. In this paper, an algebraic multigrid (AMG) method, which combines the classical coarsening scheme by [19] with an energy-minimizing interpolation algorithm by [26], is employed and tested for subsurface water simulations. Based on numerical tests using real field data, our results suggest that the energy-minimizing algebraic multigrid method is efficient and, more importantly, very robust.

Key words. subsurface water simulation, multigrid, algebraic multigrid, energy-minimizing interpolation.

1. Introduction

Various mathematical models have been proposed to simulate the flow of water, sediment, chemicals, nutrients, and microbial organisms within watersheds, as well as to quantify the impact of human activities in the course. In these models, systems of partial differential equations (PDEs) are used to describe one or several physical processes of the subsurface water flow. As it is often impossible to obtain exact solutions, we turn to numerical simulations to help us understand the complicated physics underlying such processes.

Various discretization methods for PDEs (see [17, 7]) can be applied to reduce the continuous differential equations to finite dimensional sparse linear systems. There are many possible algorithms for solving sparse linear systems (see [20]) that arise from discretizations of PDEs, among which is the geometric multigrid (GMG) method. When applicable, GMG is generally considered one of the most efficient techniques (e.g., [1, 6, 25, 22] and [11]).

However, since GMG requires an explicit hierarchy of the underlying grids and its implementation is problem dependent, its applicability is limited in practice. The

Received by the editors January 31 of 2011 and, in revised form, February 28, 2012.

2000 *Mathematics Subject Classification.* 65F10, 65N22.

Cheng's research is funded by Army 6.1 Basic Research project in Civil Work category entitled "An intelligent linear solver system for scalable parallel solution of large scale surface and subsurface flow and transport problems."

Huang's research was partially supported by Zhejiang Provincial Natural Science Foundation of China (Y6110240) and NSFC11171257.

Shu's research was partially supported by NSF Key Project (Grant NO. 11031006) and NSFC11171281.

Xu's research was partially supported by NSF DMS0915153 and U.S. Army Corps of Engineers BAA094478.

C. Zhang's research was supported by the Deans Startup Fund, Academy of Mathematics and System Sciences and NSFC-91130011.

S. Zhang's research was partially supported by NSFC11101415 and the National Centre for Mathematics and Interdisciplinary Sciences, Chinese Academy of Sciences.

Zhou's research was partially supported by the Key Project of Scientific Research Fund of Hunan Provincial Science and Technology Department (Grant No. 2011FJ2011).

algebraic multigrid (AMG) method, on the other hand, requires minimal geometric information about the underlying problem and can sometimes be employed as a "black-box" solver for a range of problems. The classical AMG method ([3] and [18]) has been shown to be effective for a range of problems (cf. [19, 21] and [10]).

Since the early 1980s, considerable effort has been devoted to enlarging the applicability of AMG. Many different types of algebraic multigrid methods have been developed. In the interest of making implementation easier, [23, 13] and [16] all proposed aggregation-type AMG methods. The energy-minimizing interpolation approach ([24, 26]) constructs coarse-grid spaces by computation to enhance stability and approximation. Other examples of AMG methods include element interpolation based AMG (or AMGe) by [5, 12] and [8], element agglomeration based AMG by [14], and compatible relaxation based techniques (see [2, 9, 15] and [4]). Each has advantages for solving certain problems.

There are several well-established free and commercial software packages for algebraic multigrid methods. However, we have found that the problems inhering in subsurface water simulations are still extremely challenging in practice; see Section 2 for details. The main difficulties lie in heterogeneity and anisotropic coefficients with large jumps. For some of the packages we tested, it was necessary to tune their parameters for every single test problem in order to make the iterative solvers converge to a satisfactory tolerance.

In this paper, we employ an energy-minimizing AMG method to address a subsurface water simulation problem. This method utilizes the Ruge–Stüben strategy to choose coarse-grid variables and to establish the interpolation operator on each level, it uses an energy-minimizing (EM) approach. The main idea of EM interpolation was proposed by [24] and later explored by [26]. We use the algorithm given by [26]. Two essential components for the convergence of multigrid methods, namely stability and approximation, are taken into account when constructing coarse-grid spaces; and, numerical experiments show that this AMG method, referred to as EMAMG, is robust with respect to the parameters of the subsurface water simulation test problems under consideration here.

We compare EMAMG with the well-known classical AMG method—using the Ruge–Stüben strategy to choose coarse-grid variables and the direct interpolation for constructing grid transfer operators. The classical AMG method can be found in references including [18, 21], and [22]. Numerical experiments in the present paper show that, in our implementation, the EMAMG method outperforms the AMG based on the classical interpolation (or RSAMG in short) for discrete problems arising from subsurface flow modeling. Hence, EMAMG is a competitive alternative to RSAMG.

The remainder of the paper is organized as follows. A model for simulating subsurface water problems is described in Section 2. The two types of AMG methods (RSAMG and EMAMG) under consideration are introduced in Section 3. Numerical experiments for RSAMG and EMAMG are presented in Section 4. Finally, some concluding remarks and observations based on our experiments are given in Section 5.

2. A model problem for subsurface water systems

Many subsurface models have been established to describe the contaminant transportation through saturated–unsaturated porous media. WASH123D is one such model based on the first principles (see [27]). We take an important equation from WASH123D as the test problem and it is labeled as SD-5. SD-5 is a

three-dimensional problem on a subdomain of the first draft of the South Florida Regional Engineering Model for Ecosystem Restoration (REMER). It covers most of the land area south of the Tamiami Trail in South Florida, which is bounded by the Tamiami canal and the C4 canal in the north, and by the shores of the Gulf of Mexico, Florida Bay, and Biscayne Bay (Fig. 1).

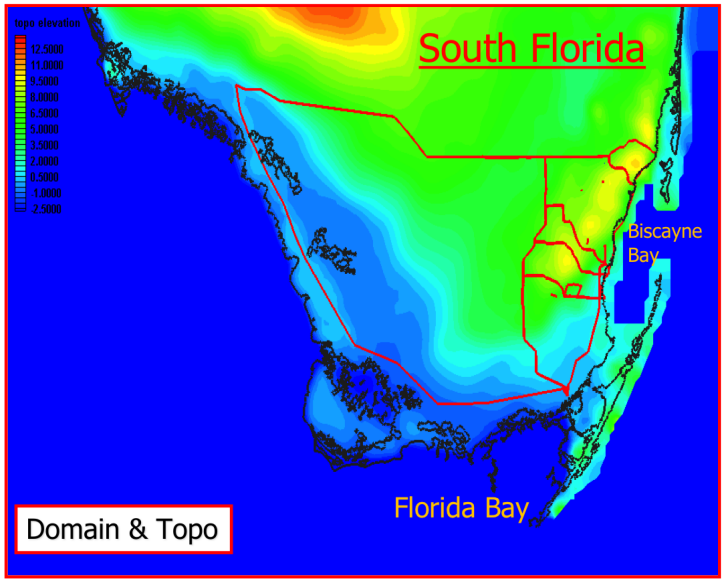

FIGURE 1. Illustration of the domain of SD-5.

In WASH123D, the governing equation of subsurface density dependent flow through saturated–unsaturated porous media can be derived based on the mass conservation of water; i.e.,

$$\text{(1)} \qquad \begin{cases} \frac{\rho}{\rho_0}\mathcal{F}\frac{\partial h}{\partial t} &= -\nabla\cdot(\frac{\rho}{\rho_0}\mathbf{V}) + \frac{\rho^*}{\rho_0}q \\ \mathbf{V} &= -\mathbf{K}\cdot(\frac{\rho_0}{\rho}\nabla h + \nabla z), \end{cases}$$

where the first equation is from the mass conservation and the second is Darcy's law for porous media. Here is a list of the most important variables:

- ρ is the density of the water [M/L^3];
- ρ_0 is the referenced density of the water [M/L^3];
- h is the referenced pressure head [L];
- ρ^* is the density of the source water;
- q is the source and/or sink [L^3/t/L^3];
- $\mathbf{V}$ is the Darcy velocity [L/t];
- $\mathbf{K}$ is the hydraulic conductivity tensor [L/t];
- z is the potential head [L];
- $\mathcal{F} = \alpha'\frac{\theta_e}{n_e} + \beta'\theta_e + n_e\frac{d\mathcal{S}}{dh}$ is the water capacity [1/L];
- α' is the modified compressibility of the medium [1/L];
- β' is the modified compressibility of the water [1/L];
- θ_e is the effective moisture content [L^3/L^3];
- n_e is the effective porosity [L^3/L^3];
- $\mathcal{S}$ is the degree of saturation.

The initial condition of h is equipped on the region of interest R, and different types of boundary conditions on boundary B of R have been considered (see [27] for details).

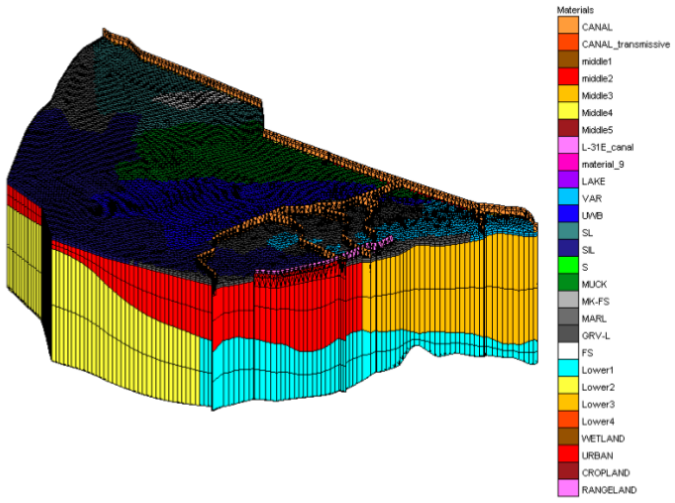

FIGURE 2. A sample mesh of the SD-5 computational domain.

The governing equation is discretized by the finite element method. The nonlinearity of the system is treated using Picard iteration, and the generated set of linearized equations can be solved with the methods described in the next section. We use the backward finite difference method to approximate the temporal derivative term $\frac{\partial h}{\partial t}$ in (1) and the finite element method in space. The referenced head in (1) is approximated by

$$h \approx \tilde{h} = \sum_{j=1}^{n} h_j(t) N_j(x, y, z) \tag{2}$$

where h_j and N_j are the amplitude of h and the base function, respectively, at nodal point j and n is the total number of nodes. Once the residual has been defined and the weighted residual forced to zero, (1) is approximated as

$$\left[\int_R N_i F N_j dR\right] \frac{dh_j}{dt} + \left[\int_R (\nabla N_i) \cdot \mathbf{K} \cdot (\nabla N_j) dR\right] h_j = \int_R N_i q dR - \int_R (\nabla N_i) \cdot \mathbf{K} \cdot \nabla z dR + \int_B \mathbf{n} \cdot \mathbf{K} \cdot \nabla (h + z) N_i dB. \tag{3}$$

Furthermore, (3) can then be written in matrix form:

$$[\mathbf{M}] \cdot \frac{\{h\}^{t+\Delta t} - \{h\}^t}{\Delta t} + [\mathbf{S}] \cdot \{h\}^{t+\Delta t} = \{Q\} + \{G\} + \{B\}. \tag{4}$$

The matrices, $[\mathbf{M}]$ and $[\mathbf{S}]$, and vectors, $\{G\}$, $\{Q\}$, and $\{B\}$, are the mass matrix, the stiffness matrix, the gravity vector, the source/sink vector, and the boundary term vector, respectively, evaluated using the estimate of $\{h\}^t$ from the previous iteration. The mass matrix $[\mathbf{M}]$ and the stiffness matrix $[\mathbf{S}]$ are defined as

$$M_{ij} = \sum_{e \in M_e} \int_{R_e} N_\alpha^e F N_\beta^e dR \quad \text{and} \quad S_{ij} = \sum_{e \in M_e} \int_{R_e} (\nabla N_\alpha^e) \cdot \mathbf{K} \cdot (\nabla N_\beta^e) dR,$$

where R_e is the region of element e, M_e is the set of elements that each has a local side $\alpha - \beta$ coinciding with the global side $i - j$, and N_α^e is the $\alpha - th$ local base function of element e. The three load vectors $\{Q\}$, $\{G\}$, and $\{B\}$ are defined as

$$Q_i = \sum_{e \in M_e} \int_{R_e} N_\alpha^e \, q \, dR, \quad G_i = - \sum_{e \in M_e} \int_{R_e} (\nabla N_\alpha^e) \cdot \mathbf{K} \cdot \nabla z \, dR, \tag{5}$$

$$B_i = - \sum_{e \in N_{se}} \int_{B_e} N_\alpha^e \, \mathbf{n} \cdot [-\mathbf{K} \cdot \nabla (h + z)] dB, \tag{6}$$

where N_{se} is the set of boundary segments that each has a local node coinciding with the global node i and where B_e is the length of boundary segment e.

Then (4) can be reduced to a linear system of equations where the unknown vector is $\{h\}^{t+\Delta t}$, and can be represented as follows:

$$[\mathbf{A}]\{h\} = \{L\} + \{B\} =: \{R\} \tag{7}$$

where $[\mathbf{A}]$ is the assembled coefficient matrix, $\{h\}$ is the unknown vector to be computed and represents the values of the discretized pressure field at the new time level, $\{L\}$ is the load vector resulting from initial conditions and all types of sources/sinks, $\{B\}$ is the load vector contributed by the boundary conditions including the global boundary and the media-interface boundaries, and $\{R\}$ is the final righthand-side vector.

For simplicity, we will write the above linear system (7) as

$$Ax = b. \tag{8}$$

In this paper, a sequence of symmetric positive definite (SPD) linear systems (selected because they are difficult to solve) from SD-5 are used to test the computational efficiency of the solvers. These problems arise from finite element discretization of (1) on different meshes; see Fig. 2 for a sample mesh. Some basic algebraic properties of the test problems are summarized in Table 1, where "n" is the number of unknowns, "nnz" is the number of nonzeros in the coefficient matrix, "DDR" is the ratio between the number of diagonally dominant rows and the total number of rows, and "NMR" is the percentage of rows that contain positive off-diagonal entries.

TABLE 1. Basic properties of the five test matrices.

System	n	nnz	nnz/n	symmetry	DDR	NMR
1	59409	1075411	18.10	Yes	3.71%	96.73%
2	93423	1743045	18.66	Yes	4.52%	95.91%
3	178353	3433171	19.25	Yes	4.12%	95.91%
4	717486	13875608	19.34	Yes	1.63%	98.52%
5	2124108	42568402	20.00	Yes	1.08%	98.92%

None of the five coefficient matrices is an M-matrices, nor are any of them diagonally dominant. If we apply the diagonal preconditioned conjugate gradient method to solving these problems, the iteration numbers will be more than $15,000$ for each of the five problems. As noted, we also chose software packages that contain many different types of AMG methods, such as *hypre*, ML, and SAMG; however, we failed to find a robust solver (or set of parameters) for all the test problems under investigation.

3. An energy-minimizing algebraic multigrid method

Algebraic multigrid methods consist of two phases: the SETUP phase, which generates the multi-level coarse-grid problems and the intergrid transfer operators, and the SOLVE phase, during which the smoothing and coarse-grid correction proceed recursively.

The goal of the SETUP phase is to construct the intergrid transfer operator P, which is a prolongation operator from the coarse level to the fine level, and the coarse grid problem is then constructed. A general process for constructing P from n_c coarse grid points to n fine grid points is described by the following generic two-level algorithm:

- Choose a set of n_c coarse grid points (coarse variables);
- Choose a sparsity pattern of the prolongation $P \in \mathbb{R}^{n\times n_c}$;
- Define the weights of the interpolation (i.e., the entries of P).

The next level operator can be given by $A_c = P^T AP \in \mathbb{R}^{n_c\times n_c}$, where P^T is the transpose of P. For multi-level methods, the process can be repeated recursively, and the coarse grid problems are constructed level by level. In this section, the SETUP phase of the classical and energy-minimizing interpolations is described for completeness.

3.1. Coarsening. The purpose of the coarsening step is to choose the set of coarse degrees of freedom (C/F splitting) and build a sparsity pattern for the prolongation operator. In this paper, for historical reason, we do not distinguish "coarse-grid point" from "coarse degree of freedom." The coarse-grid points are selected using the maximal independent subset idea, thereby ensuring that the problem scale is significantly reduced on the basis of accurate approximation properties. We apply the coarsening strategy from [19] to choose coarse grid points. This is equivalent to splitting the set of degrees of freedom (set of DOFs, denoted here by N) as a union of two non-overlapping sets: a set of coarse-grid DOFs (denoted by C) and a set of fine-grid DOFs (denoted by F). We then have $N = C \cup F$ and $C \cap F = \emptyset$.

Given a full set of points, the concepts of *strong dependence* and *strong influence* are essential to the selection of coarse-grid points. For a given point i, its neighborhood is denoted by $N_i = \{j \in N : j \neq i, a_{ij} \neq 0\}$, and the set of points that i *strongly depends on* or *strongly connected to* is defined as

$$S_i := \begin{cases} \emptyset & \text{if } |\sum_{l=1}^{n} a_{il}| \geq \theta_2 |a_{ii}| \\ \{j \in N_i : -a_{ij} \geqslant \theta_1 \max_{k\neq i}(-a_{ik})\} & \text{otherwise,} \end{cases}$$

for given parameters $0 < \theta_1 \leqslant 1$ and $0 < \theta_2 < 1$. The set of DOFs that are strongly influenced by i is, in turn, denoted by

$$S_i^T := \{j : i \in S_j\}.$$

Remark 1. The two parameters θ_1 and θ_2 are connected to the concept of strong dependence. Different values of θ_1 and θ_2 can imply different effects of the whole multigrid iterator. See the numerical experiments in Section 4 for more discussion.

The criteria given by [19] for C/F splitting can be described as follows:

C1: For each $i \in F$, each point $j \in S_i$ should either be in C, or strongly connected to at least one point in $C_i (= C \cap S_i)$.

C2: C should be a maximal subset of all points wherein no two C-points are strongly connected.

These two goals, **C1** and **C2**, could be contradictory. Hence, we enforce **C1** only and treat **C2** as a guideline. The final algorithm consists of two passes: The

first pass is to divide the DOFs into coarse-grid variables and fine-grid variables under the guidance of **C2**, and the second is to enforce **C1**, by possibly adding more coarse-grid variables. In the rest of this paper, we will use $|S|$ to denote the cardinality or number of the entries of a finite set S.

Algorithm 1. (First Pass) Initialization: $C = \emptyset$, $F = \emptyset$, $U = N$.

1. Compute $\lambda_i = |S_i^T|$ for all $i \in N$.
2. Pick an $i \in U$ with maximal λ_i. Set $C = C \cup \{i\}$ and $U = U \setminus \{i\}$.
3. For all $j \in S_i^T \cap U$, perform (1) and (2):
   (1) Set $F = F \cup \{j\}$ and $U = U \setminus \{j\}$.
   (2) For all $l \in S_j \cap U$, set $\lambda_l = \lambda_l + 1$.
4. For all $j \in S_i \cap U$, set $\lambda_j = \lambda_j - 1$.
5. If $U = \emptyset$, stop; otherwise, go to Step 2.

Algorithm 2. (Second Pass) Initialization: $T = \emptyset$.

1. If $T \supseteq F$, stop; otherwise, pick $i \in F \setminus T$ and $T = T \cup \{i\}$.
2. Set $C_i = S_i \cap C$, $D_i^s = S_i \cap F$, and $\tilde{C}_i = \emptyset$.
3. For each $j \in D_i^s$, do
   If $S_j \cap C_i = \emptyset$, then
   – If $\tilde{C}_i = \emptyset$, set $\tilde{C}_i = \{j\}$, $C_i = C_i \cup \{j\}$.
   – If $\tilde{C}_i \neq \emptyset$, set $C = C \cup \{i\}$, $F = F \setminus \{i\}$ and go to Step 1.
4. Set $C = C \cup \tilde{C}_i$, $F = F \setminus \tilde{C}_i$, and go to Step 1.

By the two algorithms above, for any point $i \in F$, we can define the set of interpolatory variables

$$P_i = C \cap S_i \tag{9}$$

that will be used to form the interpolation. Indeed, such a set P_i is a byproduct of the C/F splitting. The sparsity pattern of P is then determined. $P_{i,j} \neq 0$ if the i^{th} fine-grid variable coincides with the j^{th} coarse-grid variable or the j^{th} coarse-grid variable belongs to the set P_i as a fine-grid variable.

3.2. Direct interpolation. Direct interpolation is one way to define the weights of the interpolation. We take a two-level case as an example to introduce the idea of direct interpolation. Define the full rank interpolation P of the form

$$e_i = (Pe^c)_i := \begin{cases} e_i^c & \text{if } i \in C \\ \sum_{k \in P_i} w_{ik} e_k^c & \text{if } i \in F, \end{cases} \tag{10}$$

where e^c is the coarse-level error and the weights w_{ik} are to be determined.

The basic idea of determining the interpolation weights is straightforward. After the process of relaxation, the algebraically smooth error, e, is assumed to satisfy

$$a_{ii} e_i + \sum_{j \in N_i} a_{ij} e_j \approx 0 \qquad \forall i \in F. \tag{11}$$

For many problems, the largest off-diagonal entries are negative, and the aforementioned method works well. Indeed, the definition of strong dependency above is motivated by the properties of M-matrices. For some algebraically smooth errors, it can be observed that

$$\frac{1}{\sum_{k \in P_i} a_{ik}} \sum_{k \in P_i} a_{ik} e_k \approx \frac{1}{\sum_{j \in N_i} a_{ij}} \sum_{j \in N_i} a_{ij} e_j.$$

Heuristically, we have an approximate local explicit interpolation formula:

$$w_{ik} = -\alpha_i a_{ik}/a_{ii}, \quad \text{with} \quad \alpha_i = \frac{\sum_{j\in N_i} a_{ij}}{\sum_{k\in P_i} a_{ik}}.$$

If there are positive off-diagonal entries, a similar argument can be applied as long as such entries are relatively small—positive couplings are ignored (simply considered weak). However, if the coefficient matrix contains relatively large positive off-diagonal entries, the above approach should be generalized by considering both positive and negative couplings, which leads to interpolation formulas containing both "positive weights" and "negative weights."

In this case, the set P_i contains two parts: $P_i = P_i^+ \cup P_i^-$, such that $a_{ij} > 0$, if $j \in P_i^+$ and $a_{ij} < 0$ if $j \in P_i^-$. For the direct interpolation, the weights are taken to be

$$w_{ik} = \begin{cases} -\alpha_i a_{ik}/a_{ii} & \text{if } k \in P_i^- \\ -\beta_i a_{ik}/a_{ii} & \text{if } k \in P_i^+, \end{cases} \tag{12}$$

with

$$\alpha_i = \frac{\sum_{j\in N_i} a_{ij}^-}{\sum_{k\in P_i} a_{ik}^-} \quad \text{and} \quad \beta_i = \frac{\sum_{j\in N_i} a_{ij}^+}{\sum_{k\in P_i} a_{ik}^+},$$

where $a_{ij}^{\pm}$ are the positive and negative parts of a_{ij}. The sets P_i^+ and P_i^- are generated separately by the same procedures. If $P_i^+ = \emptyset$, (12) is modified by setting $\beta_i = 0$ and adding all positive entries, if any, to the diagonal.

3.3. Energy-minimizing interpolation. In this subsection, we introduce another way to define the weights of the interpolation (cf. [24, 26].) We assume that the coarse-grid selection is conducted as in Section 3.1 and that C/F splitting with n_c coarse-grid points and interpolatory sets are given.

Algorithm 3 (Energy-minimizing interpolation)**.** We define the interpolation by

(1) For $k = 1, \ldots, n_c$, find A_k^{-1}, where

$$A_k := (\mathbf{e}_{i_{k1}}, \ldots, \mathbf{e}_{i_{kn_k}})^T A (\mathbf{e}_{i_{k1}}, \ldots, \mathbf{e}_{i_{kn_k}});$$

(2) For $k = 1, \ldots, n_c$, compute

$$T_k := (\mathbf{e}_{i_{k1}}, \ldots, \mathbf{e}_{i_{kn_k}}) A_k^{-1} (\mathbf{e}_{i_{k1}}, \ldots, \mathbf{e}_{i_{kn_k}})^T;$$

(3) Let $\mathbf{1}$ be the vector with all entries as 1. Calculate

$$g := (\sum_{k=1}^{n_c} T_k)^{-1} \mathbf{1};$$

(4) For $k = 1, \ldots, n_c$, calculate $p_k := T_k g$.

In the algorithm above, we use the following notation:

- $A \in \mathbb{R}^{n\times n}$: the coefficient matrix of the fine grid;
- $S^c = \{1, 2, \ldots, n_c\}$: the label of the nodes in the coarse grid;
- $\mathbf{e}_i$: the i^{th} column of the identity matrix $I \in \mathbb{R}^{n\times n}$;
- $\{i_{k1}, i_{k2}, \ldots, i_{kn_k}\}$: the label number in the fine grid of the nodes associated with the k^{th} node in the coarse grid, where n_k is the number of the fine-grid nodes associated with the k^{th} coarse-grid node.

In the end, the matrix $P := [p_1, p_2, \ldots, p_{n_c}]$ is set to be the interpolation matrix from coarse nodes to fine nodes.

Remark 2. One major concern for the energy-minimizing basis is the computational cost for its construction; more specifically, the question is whether it is possible to economically compute g in Step (3). The answer to the question is positive because the operator $\sum_{k=1}^{n_c} T_k$ is either well conditioned or can easily be preconditioned by local operations. We refer to [26] for details.

3.4. Algebraic multigrid algorithm. As mentioned before, an AMG method consists of a SETUP phase and a SOLVE phase. These two phases are summarized in the two algorithms below.

Algorithm 4 (SETUP phase)**.** Let $l = 1$. Given the finest matrix A_l.

1. Coarsening:
   a) Carry out coarse-grid selection with the Ruge–Stüben strategy;
   b) Generate prolongation matrix P_{l+1}^l;
   c) Generate coarser-level matrix $A_{l+1} = (P_{l+1}^l)^T A_l P_{l+1}^l$;
   d) Let $l = l + 1$.
2. Repeat Step 1 until the DOFs at level l are small enough or l is big enough.

Remark 3. Using Algorithm 4, the SETUP phase generates multi-level matrices

$$A_{l+1} = (P_{l+1}^l)^T A_l P_{l+1}^l, \quad l = 1, \ldots, L-1,$$

where L is the coarsest level. Furthermore, for step b) in Algorithm 4, we can employ either the direct interpolation in Section 3.2, the energy-minimization interpolation in Section 3.3, or another interpolation scheme.

Once the SETUP is complete, the SOLVE phase proceeds, during which iterators B_l are defined level by level. On each level, the action of B_l is a combination of smoothings and coarse-grid corrections. We will use the simplest possible method, i.e., the standard multigrid V-cycle with the pointwise Gauss–Seidel (GS) smoother. We denote the GS smoother on level l by R_l.

Given any vector z_l, $e_l = B_l z_l$ (the action of the MG V-cycle and the W-cycle SOLVE phase) can be defined recursively as follows:

Algorithm 5. (V-cycle SOLVE phase) For a given z_l, e_l is obtained by

1) If $l = L$, then solve the equation $A_L e_L = z_L$ exactly and return.
2) If $l < L$, then
   A) **pre-smoothing**:
   $$e_l = R_l z_l, \quad r_l = z_l - A_l e_l;$$
   B) **coarse-grid correction**
      a) **restriction:**
      $$r_{l+1} = (P_{l+1}^l)^T r_l;$$
      b) **coarse-grid relaxation:**
      $$e_{l+1} = B_{l+1} r_{l+1};$$
      c) **prolongation:**
      $$e_l^c = P_{l+1}^l e_{l+1}, \quad r_l = r_l - A_l e_l^c;$$
   C) **post-smoothing:**
   $$e_l = e_l + e_l^c + R_l r_l, \quad z_l = z_l - A_l e_l.$$
3) $l = l + 1$ and go to 1).

Remark 4. For a W-cycle, we only need to modify Step b) by forming the residual and applying B_{l+1} one more time.

Remark 5. Note that there are many possible choices for the SOLVE phase; however, we will not discuss them here as the main focus of this paper is using the energy-minimizing interpolation to improve the performance of the SETUP phase.

4. Numerical experiments

To make a comparison, we implement six multi-level solvers and test them on the discrete problem (8) using the real data from SD-5:

(1) RSAMG: This solver employs an AMG V-cycle/W-cycle iterator for the SOLVE phase, and the SETUP phase uses the Ruge–Stüben strategy.
(2) EMAMG: This solver employs an AMG V-cycle/W-cycle iterator for the SOLVE phase, and the SETUP phase uses the energy-minimizing strategy as in the previous section.
(3) RSCG: This solver uses a preconditioned conjugate gradient method with one RSAMG V-cycle/W-cycle iterator as a preconditioner.
(4) EMCG: This solver uses a preconditioned conjugate gradient method with one EMAMG V-cycle/W-cycle iterator as a preconditioner.
(5) RSBCGs: This solver uses a preconditioned BiCGstab method with one RSAMG V-cycle/W-cycle iterator as a preconditioner.
(6) EMBCGs: This solver uses a preconditioned BiCGstab method with one EMAMG V-cycle/W-cycle iterator as a preconditioner.

All our numerical experiments are performed on a computer with Intel Xeon E5530 2.4GHz CPU and 24GB 1067MHz DDR3 RAM. All the numerical tests are performed using our in-house linear solver package, FASP (see `http://www.multigrid.org`).

For the SETUP phase, three of the above methods, (1), (3) and (5), employ the classical Ruge–Stüben strategy, and the other three, (2), (4) and (6), employ the energy-minimizing algorithm described in the previous section. The coarsening process will stop if the number of DOFs on the coarse level is smaller than 500. In each V- or W-cycle, one Gauss-Seidel (GS) or Symmetric Gauss-Seidel (SGS) sweep with C/F ordering is employed for both pre- and post-smoothings. The matrix equations on the coarsest grid are solved exactly. The initial guess is always zero and the stopping criteria is that the relative residual in the Euclidean norm is less than 10^{-8}.

We first compare the performance of RSAMG and EMAMG for a small test problem, System 1. The results in Tables 2 and 3 show that, in our implementation, EMAMG is robust for a range of thresholds, $\theta_1 \in [0.4, 0.9]$ and $\theta_2 \in [0.5, 0.9]$, and the EMAMG outperforms the RSAMG. Moreover, the operator complexity, i.e., the ratio of the total number of non-zero entries of matrices on all levels and the number of non-zero entries of the matrix on the finest level, is *robust* with respect to θ_1 and θ_2 as well (Tables 4 and 5). Therefore, the total cost of each cycle of the SOLVE phase is robust with respect to the parameters. We also notice that the number of levels of the hierarchical structure is robust with respect to θ_1 and θ_2 for both the RSAMG and EMAMG SETUPs: both end up with 8 levels independent of θ_1 and θ_2.

In all our subsequent tests, described next, we fix $\theta_1 = 0.6$ and $\theta_2 = 0.9$. The number of levels in the hierarchical structures are recorded in Table 6. The outer iteration of the AMG SOLVE phase can be either the V-cycle or the W-cycle, and the smoother on the fine grid can be the Gauss-Seidel iteration or the symmetric

TABLE 2. Number of iterations using RSAMG (V+GS) on System 1 for different parameters θ_1 and θ_2

θ_2	θ_1						
	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.4	3766	1321	1281	1287	1124	1126	1284
0.5	3745	1222	1198	1202	1220	1202	1202
0.6	3687	1250	1214	1209	1255	1255	1216
0.7	3685	1311	1285	1213	1225	1255	1284
0.8	3673	1302	1214	1285	1284	1224	1284
0.9	3670	1311	1224	1215	1285	1230	1216

TABLE 3. Number of iterations using EMAMG (V+GS) on System 1 for different parameters θ_1 and θ_2

θ_2	θ_1						
	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.4	1704	1688	1708	1690	1709	1709	1709
0.5	188	47	45	39	36	47	58
0.6	352	61	50	30	32	47	56
0.7	364	46	40	34	31	59	66
0.8	491	43	38	29	30	55	62
0.9	412	66	39	36	29	48	60

TABLE 4. Operator complexity using RSAMG on System 1 for different parameters θ_1 and θ_2

θ_2	θ_1						
	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.4	2.04	1.92	1.88	1.84	1.82	1.79	1.75
0.5	2.05	1.93	1.88	1.85	1.82	1.80	1.76
0.6	2.05	1.93	1.88	1.85	1.82	1.80	1.76
0.7	2.06	1.92	1.87	1.84	1.82	1.80	1.76
0.8	2.05	1.92	1.87	1.84	1.82	1.80	1.76
0.9	2.04	1.92	1.87	1.85	1.82	1.80	1.76

Gauss-Seidel iteration. The results are recorded in Tables 7–14. In the tables, the choices of the outer multilevel iteration and smoother are labeled as "V+GS," "W+GS," "V+SGS" and "W+SGS" for V-cylce with GS smoother, W-cycle with GS smoother, V-cycle with SGS smoother, and W-cycle with SGS smoother, respectively. In each case, we record the number of outer iterations and the time used and compare them with respect to RSAMG and EMAMG. According to our experiments, the solvers based on EMAMG perform better than those based on RSAMG; the SETUP phase of EMAMG is more time-consuming than that of RSAMG, but the total CPU time of the EMAMG-based solvers is shorter, and the EMAMG-based solvers are more robust with respect to the problem size.

5. Concluding Remarks

In this paper, we applied an energy-minimizing AMG method to a subsurface water simulation. We presented a brief introduction to an AMG method—the

TABLE 5. Operator complexity using EMAMG on System 1 for different parameters θ_1 and θ_2

θ_2	θ_1						
	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.4	2.05	1.92	1.88	1.85	1.82	1.79	1.75
0.5	2.05	1.93	1.88	1.85	1.82	1.80	1.76
0.6	2.06	1.92	1.88	1.85	1.82	1.80	1.76
0.7	2.06	1.93	1.87	1.84	1.82	1.79	1.76
0.8	2.06	1.92	1.87	1.84	1.82	1.80	1.76
0.9	2.05	1.92	1.87	1.84	1.82	1.80	1.76

TABLE 6. Number of levels for Systems 1—5 (In the coarsening processes, we fix $\theta_1 = 0.6$ and $\theta_2 = 0.9$)

System	1	2	3	4	5
RS coarsening	8	8	9	11	12
EM coarsening	8	7	9	10	12

TABLE 7. Number of iterations using RSAMG and EMAMG (V+GS)

System	Number of iterations					
	RSAMG	EMAMG	RSCG	EMCG	RSBCGs	EMBCGs
1	1215	36	60	15	34	7
2	1771	35	101	14	70	6
3	3287	34	115	15	83	7
4	276	32	52	13	28	6
5	375	61	43	20	23	10

TABLE 8. Total CPU time using RSAMG and EMAMG methods (V+GS)

System	Total CPU time (seconds)					
	RSAMG	EMAMG	RSCG	EMCG	RSBCGs	EMBCGs
1	21.20	0.87	1.11	0.49	1.34	0.52
2	47.36	1.33	2.82	0.77	3.99	0.75
3	169.43	2.56	8.26	1.54	9.51	1.64
4	79.99	9.69	16.19	7.04	13.72	6.27
5	363.36	70.77	41.92	36.44	46.66	39.45

energy-minimizing AMG method, and performed numerical experiments to illustrate the performance of the method. Here are some of our observations:

(1) When applied to the subsurface flow simulation problems from real applications, we found that, as a standalone iterative solver, EMAMG is quite robust with respect to the problem size.
(2) When accelerated by the Krylov methods, the robustness of RSAMG improves; however, it still did not match the performance of EMAMG for the test problems under consideration.
(3) Both the coarsening strategy (Ruge–Stüben method) and the interpolation scheme (energy-minimizing interpolation) play important roles in the convergence of the AMG method.

TABLE 9. Number of iterations using RSAMG and EMAMG with (V+SGS)

System	Number of iterations					
	RSAMG	EMAMG	RSCG	EMCG	RSBCGs	EMBCGs
1	810	33	50	12	28	6
2	3342	30	81	13	60	6
3	5406	29	107	13	68	7
4	458	24	40	11	22	5
5	453	62	44	18	25	9

TABLE 10. Total CPU time using RSAMG and EMAMG methods (V+SGS)

System	Total CPU time (seconds)					
	RSAMG	EMAMG	RSCG	EMCG	RSBCGs	EMBCGs
1	18.60	0.99	1.29	0.51	1.78	0.56
2	119.40	1.45	3.13	0.85	4.84	0.90
3	414.29	2.87	8.43	1.74	10.98	1.99
4	131.03	10.00	13.28	6.52	15.57	6.98
5	610.99	89.25	60.56	34.78	73.13	37.68

TABLE 11. Number of iterations using RSAMG and EMAMG (W+GS)

System	Number of iterations					
	RSAMG	EMAMG	RSCG	EMCG	RSBCGs	EMBCGs
1	794	33	48	13	30	8
2	1715	25	87	12	63	5
3	3210	23	110	10	62	5
4	264	20	41	11	25	5
5	237	66	34	18	17	9

TABLE 12. Total CPU time using RSAMG and EMAMG methods (W+GS)

System	Total CPU time (seconds)					
	RSAMG	EMAMG	RSCG	EMCG	RSBCGs	EMBCGs
1	158.25	6.61	9.61	2.69	12.28	3.67
2	342.65	6.00	17.07	3.02	25.10	3.02
3	1549.94	12.46	50.63	5.77	77.06	6.91
4	540.42	42.50	86.36	25.40	106.58	26.96
5	1881.46	482.31	258.79	132.02	275.10	145.85

TABLE 13. Number of iterations using RSAMG and EMAMG (W+SGS)

System	Number of iterations					
	RSAMG	EMAMG	RSCG	EMCG	RSBCGs	EMBCGs
1	622	30	45	11	28	6
2	2732	22	75	11	53	5
3	5143	21	96	10	67	5
4	370	16	36	9	21	4
5	322	63	37	17	20	9

TABLE 14. Total CPU time using RSAMG and EMAMG methods (W+SGS)

System	Total CPU time (seconds)					
	RSAMG	EMAMG	RSCG	EMCG	RSBCGs	EMBCGs
1	139.03	6.63	10.20	2.58	13.18	3.22
2	678.23	5.92	17.53	3.15	29.46	3.42
3	3015.75	13.08	52.85	6.67	74.93	9.22
4	958.59	47.65	89.49	24.19	109.25	26.62
5	3003.56	638.15	351.29	158.32	400.40	184.58

(4) The AMG method that employs the RS coarsening strategy together with EM interpolation is robust for the test problems. However, we noticed that it would be quite another story if the RS coarsening were replaced by simpler coarsening strategies.
(5) The numerical examples indicate that EMAMG can bring better applicability and robustness with respect to the parameters, θ_1 and θ_2.
(6) The SETUP phase of EMAMG costs more CPU time and memory than the classical AMG. Therefore, EMAMG is preferable only when the linear systems are relatively difficult to solve, i.e., when the SOLVE phase dominates the whole computational time.

Finally, we would like to point out that, when the computational technique presented in [26] is used, the EMAMG method can be processed entirely through pure algebraic operations. The EMAMG method does not require any use of the geometric information related to the PDE or the discretization. Therefore, such a method is automatically applicable to most of the numerical models of subsurface flows.

References

[1] J.H. Bramble, Multigrid Methods, volume 294 of Pitman Research Notes in Mathematics Series, Longman Scientific & Technical, Harlow, 1993.
[2] A. Brandt, General Highly Accurate Algebraic Coarsening, Electron. Trans. Numer. Anal., **10**(2000), 1-20.
[3] A. Brandt, S. McCormick, and J. Ruge, Algebraic Multigrid (AMG) for Automatic Multigrid Solution with Application to Geodetic Computations, Report. Inst. Comp. Studies Colorado State Univ., 109:110 (1982).
[4] J. Brannick, and L. Zikatanov, Algebraic Multigrid Methods Based On Compatible Relaxation and Energy Minimization, Lecture Notes in Comput. Sci. and Eng., **55**(2007), 15-26.
[5] M. Brezina, A. Cleary, R. Falgout, V. Henson, J. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge, Algebraic Multigrid Based on Element Interpolation (AMGe), SIAM J. Sci. Comput., **22**(2000), 1570-1592.
[6] W. L. Briggs, V. E. Henson, and S. F. McCormick, A Multigrid Tutorial Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2000.
[7] W. Cai, Z. C. Shi, C. W. Shu, and J. Xu, Numerical Methods in Applied Sciences, Science Press, Beijing, 1995.
[8] T. Chartier, R. D. Falgout, V. E. Henson, J. Jones, T. Manteuffel, S. McCormick, J. Ruge, and P. S. Vassilevski, Spectral AMGe (ρAMGe), SIAM J. Sci. Comput., **25**(2003), 1-26.
[9] R. D. Falgout, and P. S. Vassilevski, On Generalizing the Algebraic Multigrid Framework, SIAM J. Numer. Anal., **42**(2004), 1669-1693.
[10] R. D. Falgout, An Introduction to Algebraic Multigrid, Comput. Sci. and Engrg., **8**(2006), 24-33.
[11] W. Hackbusch, Multi-Grid Methods and Applications, Springer Verlag, Berlin, 2003.
[12] V. E. Henson, and P. S. Vassilevski, Element-Free AMGe: General Algorithms for Computing Interpolation Weights in AMG, SIAM J. Sci. Comput., **23**(2001), 629-650.
[13] A. Janka, Smoothed Aggregation Multigrid for A Stokes Problem, Comput. Vis. Sci., **11**(2008), 169-180.

[14] T. V. Kolev, and P. S. Vassilevski, AMG by Element Agglomeration and Constrained Energy Minimization Interpolation, Numer. Linear Algebra Appl., **13**(2006), 771-788.
[15] O.E. Livne, Coarsening by Compatible Relaxation, Numer. Linear Algebra Appl., **11**(2004) 205-227.
[16] A. C. Muresan, and Y. Notay, Analysis of Aggregation-Based Multigrid, SIAM J. on Sci. Comput., **30**(2008), 1082-1103.
[17] A. Quarteroni, and A. Valli, Numerical Approximation of Partial Differential Equations, Volume 23 of Springer Series in Computational Mathematics. Springer-Verlag, Berlin, 1994.
[18] J. W. Ruge, and K. Stüben, Efficient Solution of Finite Difference and Finite Element Equations, In D. J. Paddon and H. Holstein, editors, Multigrid Methods for Integral and Differential Equations, **3**(1985), 169-212.
[19] J. W. Ruge and K. Stüben, Algebraic Multigrid, Multigrid methods, **3**(1987), 73-130.
[20] Y. Saad, Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.
[21] K. Stüben, A Review of Algebraic Multigrid, J. Comput. Appl. Math., **128**(2001) 281-309.
[22] U. Trottenberg, C. W. Oosterlee, and A. Schüller, Multigrid, Academic Press, San Diego, CA, 2001. With contributions by A. Brandt, P. Oswald and K. Stüben.
[23] P. Vaněk, J. Mandel, and M. Brezina, Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order Elliptic Problems, Computing, **56**(1996), 179-196.
[24] W. L. Wan, T. F. Chan, and B. Smith, An Energy-Minimizing Interpolation for Robust Multigrid Methods, SIAM J. Sci. Comput., **21**(1999), 1632-1649.
[25] P. Wesseling, An Introduction to Multigrid Methods, Cos Cob, CT, 2001. Reprint of the 1992 edition.
[26] J. Xu, and L. Zikatanov, On An Energy-Minimizing Basis for Algebraic Multigrid Methods, Comput. Vis. Sci., **7**(2004), 121-127.
[27] G. T. Yeh, G. Huang, H. P. Cheng, F. Zhang, H. C. Lin, E. Edris, and D. Richards, A First Principle, Physics Based watershed model: WASH123D. In V. P. Singh and D. K. Frevert, editors, Watershed Models, 211-244. CRC Press, Boca Raton, FL, 2006.

Jing-Ru C. Cheng, Scientific Computing Research Center, Information Technology Laboratory, U.S. Army Engineer Research and Development Center, Vicksburg, MS, USA.

Xue-Hai Huang, College of Mathematics and Information Science, Wenzhou University, Wenzhou, China.

Shi Shu, Hunan Key Laboratory for Computation and Simulation in Science and Engineering, Xiangtan University, China.

Jinchao Xu, Department of Mathematics, Pennsylvania State University, PA, USA.

Chen-Song Zhang, NCMIS & LSEC, Academy of Mathematics and System Sciences, Beijing, China.

Shuo Zhang, SLEC, Institute of Computational Mathematics, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing, China.

Zhiyang Zhou, School of Mathematics and Computational Science, in Xiangtan University, Xiangtan, China.