SIAM J. SCI. COMPUT.
Vol. 38, No. 4, pp. C426–C440

SEGMENTAL REFINEMENT: A MULTIGRID TECHNIQUE FOR DATA LOCALITY*

MARK F. ADAMS†, JED BROWN‡, MATT KNEPLEY§, AND RAVI SAMTANEY¶

Abstract. We investigate a domain decomposed multigrid technique, termed segmental refinement, for solving general nonlinear elliptic boundary value problems. We extend the method first proposed in 1994 by analytically and experimentally investigating its complexity. We confirm that communication of traditional parallel multigrid is eliminated on fine grids, with modest amounts of extra work and storage, while maintaining the asymptotic exactness of full multigrid. We observe an accuracy dependence on the segmental refinement subdomain size, which was not considered in the original analysis. We present a communication complexity analysis that quantifies the communication costs ameliorated by segmental refinement and report performance results with up to 64K cores on a Cray XC30.

Key words. multigrid, parallel multigrid, distributed memory multigrid, segmental refinement

AMS subject classifications. 68Q25, 68R10, 68U05

DOI. 10.1137/140975127

1. Introduction.

Multigrid methods are widely used in practice. They are important methods to adapt to emerging architectures; however, like many algorithms their effective use on emerging architectures is challenging and is likely to become more so in the near future. Full multigrid (FMG) is an asymptotically exact, noniterative algebraic equation solver for discretized elliptic partial differential equations (PDEs) with work complexity of approximately five residual calculations, or what is known as textbook multigrid efficiency, for the constant coefficient Laplacian [3]. While textbook multigrid efficiency is only provable for a small class of elliptic problems, it has been observed experimentally in a range of problems [14, 15, 2] and is applicable to general nonlinear elliptic equations.

Memory movement, in both intranode and internode communication, and global data dependencies are the primary drivers of costs, in power and time, for PDE simulations. Memory movement pressures are not new and have been accumulating for decades, but the recent prominence of energy costs in powering memory and moving data is exacerbating this problem. Segmental refinement (SR) addresses the challenges posed by the deep memory hierarchies of modern architectures at a fundamental, al-

*Submitted to the journal's Software and High-Performance Computing section June 30, 2014; accepted for publication (in revised form) May 16, 2016; published electronically August 4, 2016. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, and performed under the auspices of the U.S. Department of Energy by Lawrence Berkeley National Laboratory under contract DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center, which is a DOE Office of Science User Facility. Authors from Lawrence Berkeley National Laboratory were supported by the U.S. Department of Energy's Advanced Scientific Computing Research Program under contract DEAC02-05CH11231.

http://www.siam.org/journals/sisc/38-4/97512.html

†Scalable Solvers Group, Lawrence Berkeley Laboratory, Berkeley, CA 94720 (mfadams@lbl.gov).

‡Department of Computer Science, University of Colorado, Boulder, CO 80309 (jedbrown@mcs.anl.gov).

§Computational and Applied Mathematics, Rice University, Houston, TX 77005 (knepley@gmail.com).

¶King Abdullah University of Science and Technology, Thuwal, 23955, Saudi Arabia (ravi.samtaney@kaust.edu.sa).

gorithmic level by exploiting the local nature of multigrid processes and a tolerance for finite algebraic error, which vanishes asymptotically. An SR data model or method explicitly decouples subdomain processing at a level of the memory hierarchy to improve data locality, to amortize latency costs, and to reduce data dependencies.

SR was proposed in the 1970s (see [5, section 7.5], [8, section 8.7], and [9]) as a low memory complexity technique for FMG to avoid storage of the entire solution in memory at any one time and was recognized for its attractive properties in distributed memory computing in the 1990s [7]. Indeed, it is inherently asynchronous and highly parallel, with no interprocess communication on fine grids, and it requires only a modest amount of extra storage and work in buffer cells. This paper extends the development of SR by quantifying its complexity both experimentally and analytically. We present multilevel numerical results of a cell-centered version of the original algorithm, present and analyze a new SR method, and report performance results with up to 64K cores on a Cray XC30.

2. Differential and discretized problems. We consider general nonlinear elliptic problems in an open domain $\Omega \subset \mathbb{R}^n$ with boundary $\partial\Omega$ of the form

$$Lu\left(\mathbf{x}\right) = f\left(\mathbf{x}\right), \qquad \left(\mathbf{x} \in \Omega\right), \tag{1}$$

where f is a known function, u is unknown, and L is a uniformly elliptic operator. While the methods described herein are generally applicable, we restrict ourselves to the three-dimensional (3D) Poisson operator: $\mathbf{x} = (x_1, x_2, x_3)$, $L = \left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \frac{\partial^2 u}{\partial x_3^2}\right)$. In addition to this interior equation, a suitable boundary condition on $\partial\Omega$ is assumed.

The discretization of (1) can be general, but we restrict ourselves to a cell-centered finite difference method on isotropic Cartesian grids and rectangular domains. For the grid Ω_h with mesh spacing h covering the domain Ω, the equation is written as

$$L_h u_h(\mathbf{i}) = f_h(\mathbf{i}), \qquad \left(\mathbf{i} \in \Omega_h\right), \tag{2}$$

where $i = \left(x - \frac{h}{2}\right)/h$ is an integer vector, $x = ih + \frac{h}{2}$ is a cell center, and the boundary $\partial\Omega$ lines up with the cell edges. In 3D, $\mathbf{i} = (i_1, i_2, i_3)$ is an index for a cell in grid Ω_h. The indexing in (2) is dropped and field variables (e.g., u_h) are vectors of scalars.

Multigrid requires an accurate solver on a coarse grid Ω_0 and smoothers on a sequence of finer grids Ω_0, Ω_1, Ω_2, ..., Ω_M, where $\Omega_k \equiv \Omega_{h_k}$, $h_k = h_{k-1}/2$, $h_M = h$, $N_k = 2N_{k-1}$, $N_M = N$. We assume Ω_h and subsequent subdomains are isotropic and are expressed as tensor products of one-dimensional (1D) grids. In addition, the length of Ω_h, in each dimension, is an integer vector. We simplify the presentation by using the integer N, since we assume cubical subdomains.

3. Multigrid background. The antecedents of modern multigrid go back at least to the 1930s [13], and the early 1960s [10], and others [15]. It was developed in its modern form in the 1970s [4] as an asymptotically exact solver with work complexity of a few residual calculations—what is known as textbook multigrid efficiency. The method was applied to complex domains, variable coefficients, and nonlinear problems [4], and a substantial body of literature, both theoretical and experimental, demonstrates the efficacy of multigrid on a range of problems [15, 8]. Full approximation scheme (FAS) is also an effective nonlinear solver, with costs similar to those of a linearized multigrid solve (e.g., [15, section 5.3.3] and [2]).

3.1. Multigrid algorithm. Multigrid starts with the observation that errors that are poorly resolved with local processes are often resolved with local processes

on a lower resolution discretization. This lower resolution problem is known as a coarse grid. A multigrid method applies this process recursively until the problem size is small enough to be solved inexpensively and exactly. The coarse grid space is represented algebraically by the columns of the prolongation operator I_H^h or I_{k-1}^k, where h is the fine grid mesh spacing and H is the coarse grid mesh spacing. Residuals are mapped from the fine grid to the coarse grid with the restriction operator I_h^H. The coarse grid operator is then formed in one of two ways (with some exceptions): either algebraically to form Galerkin (or variational) coarse grids, $L_H = I_h^H L_h I_H^h$, or by creating a new operator on each coarse grid if an explicit coarse grid with boundary conditions is available.

The so-called correction scheme (CS) multigrid, which computes corrections to the solution, is appropriate for linear problems, but FAS multigrid is more natural for segmental refinement. FAS is derived by writing the coarse grid residual equation for (2) as

$$r_H = L_H(u_H) - L_H(\hat{u}_H) = L_H(\hat{u}_H + e_H) - L_H(\hat{u}_H), \tag{3}$$

where u_H is the exact solution, $\hat{u}_H$ is an approximation to $I_h^H u_h$ (which is the full solution represented on the coarse grid), and e is the error. With an approximate solution on the fine grid $\tilde{u}_h$, the coarse grid equation is written as

$$L_H\left(I_h^H \tilde{u}_h + e_H\right) = L_H\left(I_h^H \tilde{u}_h\right) + I_h^H\left(f_h - L_h \tilde{u}_h\right) = f_H = I_h^H\left(f_h\right) + \tau_h^H \tag{4}$$

and is solved approximately. Here, τ_h^H is the tau correction, which represents a correction to the coarse grid from the fine grid. After $I_h^H \tilde{u}_h$ is subtracted from the $I_h^H \tilde{u}_h + e_H$ term, the correction is applied to the fine grid with the standard prolongation process. Algorithm 1 is a FAS multigrid $V(\nu 1, \nu 2)$-cycle algorithm with nonlinear local process or smoother $u \leftarrow S(L, u, f)$. A lower order restriction operator, $\hat{I}_h^H$, is used to restrict solution values if a higher order I_h^H is used for the residual, since the approximate coarse grid solution is subtracted from the update to produce an increment and is only needed for the nonlinearity of the operator (i.e., $\hat{I}_h^H = 0$ recovers CS multigrid).

Algorithm 1 FAS multigrid V-cycle.

function $FASMGV(L_k, u_k, f_k)$
  if $k > 0$ **then**
    $u_k \leftarrow S^{\nu 1}(L_k, u_k, f_k)$
    $r_k \leftarrow f_k - L_k u_k$
    $u_{k-1} \leftarrow \hat{I}_k^{k-1}(u_k)$
    $r_{k-1} \leftarrow I_k^{k-1}(r_k)$
    $t_{k-1} \leftarrow u_{k-1}$
    $w_{k-1} \leftarrow FASMGV(L_{k-1}, u_{k-1}, r_{k-1} + L_{k-1} u_{k-1})$
    $u_k \leftarrow u_k + I_{k-1}^k(w_{k-1} - t_{k-1})$
    $u_k \leftarrow S^{\nu 2}(L_k, u_k, f_k)$
  else
    $u_k \leftarrow L_k^{-1} f_k$
  end if
  return u_k

3.2. Full multigrid algorithm. An effective V-cycle reduces the error by a constant fraction and is thus considered an iterative method, but it is also used to

build an asymptotically exact solver that reduces the algebraic error to the order of the discretization error. This property, with $\mathcal{O}(N)$ work complexity, is termed textbook multigrid efficiency. FMG starts on the coarsest grid where an inexpensive accurate solve is available, prolongates the solution to the next finest level, applies a V-cycle, and continues until a desired resolution is reached. Algorithm 2 is the FMG algorithm, with M coarse grids and α steps of the smoother before each V-cycle, in an F(α,ν1,ν2) cycle. A higher order interpolator between the level solves, Π_H^h, is required to achieve optimal efficiency of FMG—e.g., if I_h^H is not of sufficient order (e.g., Π_H^h must be at least linear, for cell-centered 2nd-order accurate discretizations, whereas I_h^H can be constant).

Algorithm 2 Full multigrid with α pre V-cycle smoothing steps.

$u_0 \leftarrow L_0^{-1} f_0$
for $k = 1$ to M **do**
  $u_k \leftarrow \Pi_{k-1}^k u_{k-1}$
  $u_k \leftarrow S^\alpha(L_k, u_k, f_k)$
  $u_k \leftarrow FASMGV\,(L_k, u_k, f_k)$
end for
return u_0

We analyze FMG with an induction hypothesis that the ratio r of the algebraic error to the discretization error is below some value and assume where the discretization error is of the form Ch^p, where p is the order of accuracy of the discretization. Further, we assume that the solver on each level—e.g., one V-cycle—reduces the error by some factor Γ (proven or measured experimentally) to derive the relationship between Γ and r, where $\Gamma = \frac{r}{(4r+3)}$, with $p = 2$ and a refinement ratio of two. It is essential to use a sufficiently powerful solver such that $\Gamma < 0.25$. In the case of compressible resistive magnetohydrodynamics problems, two V(2,2)-cycles have been shown to be sufficient [2].

3.3. Conventional distributed memory multigrid. Domain decomposition is a natural technique for distributed memory processing of many classes of discretized PDEs, where each subdomain is placed on a processor or memory partition and the semantics of the serial algorithm are replicated. This process starts by decomposing Ω_h into P disjoint grids Ω_h^p such that $\Omega_h = \bigcup_{p=1}^P \Omega_h^p$. We use a rectangular array of processes of size (P_1, P_2, P_3) and thus $P = P_1 P_2 P_3$. In addition, boundary conditions in (2) are implemented with ghost cells, i.e., Ω_h is enlarged by one cell in all directions to form Ω_h^{+1}. Boundary ghost cell values are set with appropriate (linear) interpolation of interior values before each operator application.

We define the number of cells on each side of a (cube) subdomain Ω_h^p as the integer N_k^p on level k, again using integers for simplicity. The total number of cells in our problems is thus $n = P_1 P_2 P_3 (N_M^p)^3$, where N_M^p is the number of cells in each dimension on the fine grid. A conventional distributed memory FMG algorithm starts with a small coarse grid on a small number of processes (e.g., one process), and the coarse grid is refined and split into equally sized patches, which populate more processes. This process continues until all processes are used, forming an octree in 3D. We continue with simple refinement once all processes are used.

4. Segmental refinement. This section describes a cell-centered SR data model or method. SR begins with a conventional distributed memory FAS-FMG method,

which is used as a "coarse" grid solver. The finest level of this solver is the "transition" level and is given a grid index $k = 0$; coarser grids have negative indices. The subsequent K fine grids are defined as SR grids. Figure 1 shows a 1D example, with two SR levels and four processes.

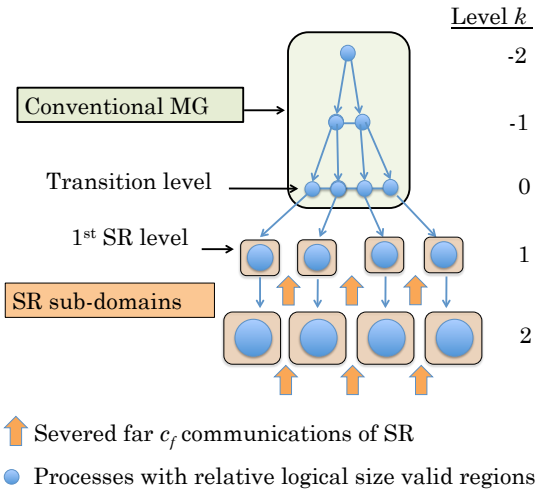

FIG. 1. *1D SR data model.*

Nonghost cells are defined as genuine cells, and the *genuine region* is defined as ${}^{p}\Omega_k^V \equiv \Omega_k^p$, where the process superscript is moved to the left. Then, SR adds buffer cells by growing each local subdomain grid by $2J_k$ cells in each dimension. We define the length or depth, in each dimension, of the SR buffer region J_k to be

$$J_k = J(k) = 2 \cdot \left\lfloor \frac{A + B \cdot (K - k)}{2} \right\rfloor, \tag{5}$$

where A is a constant term and B is a linearly increasing term on coarser grids, as defined originally [7]. J_k is constrained to even integers to simplify restriction. In the end, the union of the genuine cells and the J_k buffer cells defines the *compute region* ${}^{p}\Omega_k^C \equiv {}^{p}\Omega_k^{V+J_k} \cap \Omega_k$; that is, the genuine region grown by $J_k h_k$ in all directions and clipped by the domain. The length of the compute region is generally ${}^{p}N_k^C = {}^{p}N_k^V + 2J_k$, where ${}^{p}N_k^V$ is the length of the compute region in grid k.

We define *process ghost cells* by ${}^{p}\Omega_k^G \equiv {}^{p}\Omega_k^{C+1} \setminus {}^{p}\Omega_k^C$ and subdivide ${}^{p}\Omega_k^G$ into two sets: ${}^{p}\Omega_k^{GBC} \equiv {}^{p}\Omega_k^G \setminus \Omega_k$ and ${}^{p}\Omega_k^{GSR} \equiv {}^{p}\Omega_k^G \cap \Omega_k$. Here, the ${}^{p}\Omega_k^{GBC}$ cell values are computed with the conventional boundary condition algorithm, while ${}^{p}\Omega_k^{GSR}$ cells are set during the I_0^1 prolongation process and are "frozen," in that they are not updated with the neighbor exchanges, during the rest of the multigrid process. This "freezing" is a consequence of the elided communication of SR; ${}^{p}\Omega_k^{GSR}$ cells are set with prolongation only. We define the *support of the compute region*, on grid k, of grid $k + 1$ as ${}^{p}\Omega_k^F \equiv {}^{p}\Omega_{k+1}^C$, which is the region updated with the simple averaging restriction operator. Figure 2 shows a 1D example at the edge of the domain with two processes and two SR levels with the range of prolongation for one process.

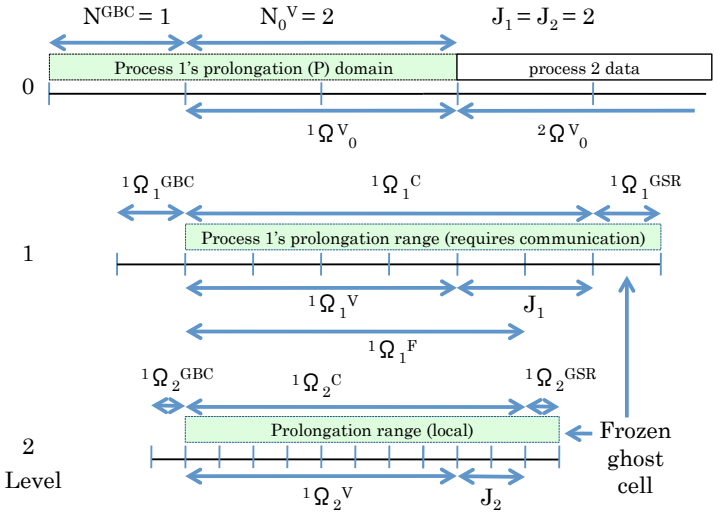

FIG. 2. *1D two process SR, with the number of boundary condition cells* N^{GBC}.

A few details to note: the residual is modified to accommodate the lack of an update in the region ${}^{p}\Omega_k^C \setminus {}^{p}\Omega_k^F$ (line 5 in Algorithm 4), and the range of prolongation is ${}^{p}\Omega^C \cup {}^{p}\Omega^{GSR}$ is expanded to include the SR domain ghost cells (line 11 in Algorithm 4) that are frozen until the next prolongation operation. Algorithms 3 and 4 are the SR FAS-FMG algorithm with annotations for the domain of each operation.

Algorithm 3 FMG with SR and α pre-V-cycle smoothing steps.

$u \leftarrow$ **function** $FASFMGSR$
  $u_0 \leftarrow FMG$
  for $k = 1$ to K **do**
    $u_k \leftarrow \Pi_{k-1}^{k} u_{k-1}$ // on grid ${}^{p}\Omega_k^C \cup {}^{p}\Omega_k^{GSR}$
    $u_k \leftarrow S^{\alpha}(L_k, u_k, f_k)$ // on grid ${}^{p}\Omega_k^C$
    $u_k \leftarrow FASMGVSR(L_k, u_k, f_k)$
  end for
  return u_K

5. Experimental observation of parameter requirements. In this section we investigate the parameters required to maintain an acceptable level of accuracy in the SR FMG solver. There are several parameters that define the SR solver: the number of SR levels K and the total number levels $M + 1$, A and B of (5), which define the buffer schedule, and the length of the subdomains on the transition level ${}^{p}N_0^V$.

5.1. Model problem and solver. We use a multigrid refinement ratio of two, piecewise constant restriction, and linear prolongation for both the FMG and V-cycle prolongation, with a 2nd-order Chebyshev polynomial presmoother and postsmoother in the V-cycle and a 1st-order Chebyshev polynomial (damped Jacobi) pre-V-cycle smoother (an F(1,2,2) cycle). This is a textbook multigrid efficient solver for our

Algorithm 4 FAS V-cycle with SR.

$u \leftarrow$ **function** $FASMGVSR(L_k, u_k, r_k)$
1: $u_k \leftarrow S^{\nu 1}(L_k, u_k, r_k)$ // on grid ${}^p\Omega_k^C$
2: $u_{k-1} \leftarrow \hat{I}_k^{k-1}(u_k)$ // on grid ${}^p\Omega_k^F$
3: $t_{k-1} \leftarrow u_{k-1}$
4: $r_{k-1} \leftarrow I_k^{k-1}(r_k - L_k u_k) + L_{k-1}u_{k-1}$ // on grid ${}^p\Omega_k^F$
5: $r_{k-1} \leftarrow L_{k-1}u_{k-1}$ // on grid ${}^p\Omega_k^C \setminus {}^p\Omega_k^F$
6: **if** $k = 1$ **then**
7: $w_{k-1} \leftarrow FASMGV(L_{k-1}, u_{k-1}, r_{k-1})$
8: **else**
9: $w_{k-1} \leftarrow FASMGVSR(L_{k-1}, u_{k-1}, r_{k-1})$
10: **end if**
11: $u_k \leftarrow u_k + I_{k-1}^k(w_{k-1} - t_{k-1})$ // on grid ${}^p\Omega_k^C \cup {}^p\Omega_k^{GSR}$
12: $u_k \leftarrow S^{\nu 2}(L_k, u_k, r_k)$ // on grid ${}^p\Omega_k^C$
13: **return** u_k

problem. The solution is prescribed as $u = \prod_{i=1}^{3}\left(x_i^4 - R_i^2 x_i^2\right)$, where $\mathbf{R} = (2, 1, 1)$, for the Laplacian, on a rectangular domain $\Omega = \{x_1, x_2, x_3 \geq 0, x_1 \leq 2, x_2, x_3 \leq 1\}$, and a 4 x 2 x 2 process grid. The problem has homogenous Dirichlet boundary conditions and a 27-point finite volume stencil that is 2nd-order accurate.

5.2. Experiments. We define an acceptable level of error, in the infinity norm, to be less than about 10% more than the conventional solver error (e_{conv}). We sample the parameter space of K, A, B, ${}^pN_0^V$, to find the manifold where the solver error transitions from acceptable to unacceptable. Table 1 shows the ratio (e_r) of the SR error (e_{SR}) to e_{conv} ($e_r \equiv e_{SR}/e_{conv}$), with $A = 2, 4, 6, 8$ (tables), $B = 0, 1, 2, 3$ (rows), and $\log_2 {}^pN_0^V$ and K (columns), and underlines the largest acceptable point in each column. The total number of multigrid levels can be inferred from ${}^pN_0^V$ and the process grid (i.e., $M = K + \log_2 {}^pN_0^V + \log_2 P_z = K + \log_2 {}^pN_0^V + 2$). This data shows that A and B both correlate with increased accuracy, which is expected because they both increase J. We observe that doubling the length of ${}^pN_0^V$ with K ($\log_2 {}^pN_0^V \propto K$) and increasing B with K ($B \propto K$) appears to maintain an asymptotically exact solver.

To further investigate the effect of ${}^pN_0^V$ on error we fix $A = 8$, $B = 0$ ($J_k = 8$), and $K = 5$. The relative error as a function of ${}^pN_0^V$ is shown in Table 2. This data shows a reduction in the error by a factor of about two with a doubling of ${}^pN_0^V$.

5.2.1. Maximum segmental refinement buffer schedule. The buffer length of the coarsest SR grid, J_1, is an important parameter because these cells require communication, since they are the range of prolongation to the coarsest SR level and the data source for all subsequent finer grid processing. To investigate the relationship of J_1 to accuracy we test with a maximum buffer schedule (MBS), where J_1 is a parameter and rest of the SR buffers completely support grid $k = 1$: ${}^p\Omega_k^C = {}^p\Omega_k^F$ for $k < K$. This is not a practical buffer schedule, because the number of buffer cells increases exponentially with refinement. Moreover, the MBS removes one source of error: the lack of update of the solution and τ correction in ${}^p\Omega_k^C \setminus {}^p\Omega_k^F$. Table 3 shows the error ratio as a function of ${}^pN_0^V$ with fixed $K = 4$ and $J_1 = 4$ and the MBS. This data shows slightly less degradation of the solution with increasing ${}^pN_0^V$ than that of Table 2, but we observe a similar doubling of the error with each halving of ${}^pN_0^V$. This indicates a dependence of accuracy on ${}^pN_0^V$, which was not recognized

TABLE 1
Ratio of SR to conventional multigrid solution error (e_r) as a function of A, B, K, and ${}^pN_0^V$.

	$\log_2 {}^pN_0^V$ (K)		
B	4(6)	3(5)	2(4)
0	17	7.2	2.7
1	2.9	2.1	1.2
2	1.5	1.2	NA
3	1.2	1.1	NA

(a) A = 2

	$\log_2 {}^pN_0^V$ (K)		
B	4(6)	3(5)	2(4)
0	5.7	2.6	1.2
1	2.0	1.4	1.0
2	1.3	1.1	NA
3	1.1	1.0	NA

(b) A = 4

	$\log_2 {}^pN_0^V$ (K)		
B	4(6)	3(5)	2(4)
0	2.8	1.4	1.0
1	1.5	1.1	NA
2	1.3	1.0	NA
3	1.1	NA	NA

(c) A = 6

	$\log_2 {}^pN_0^V$ (K)		
B	4(6)	3(5)	2(4)
0	1.5	1.1	1.0
1	1.3	1.0	NA
2	1.1	1.0	NA
3	1.0	NA	NA

(d) A = 8

TABLE 2
e_r with $A = 8$, $B = 0$, $K = 5$.

$\log_2 {}^pN_0^V$	5	4	3	2
$N = N_K$	1024	512	256	128
e_r	1.02	1.05	1.13	1.28

in the original analysis [7]. However, we have no theoretical understanding of this phenomenon. Future work is required to corroborate these results or refute them, with, say, a vertex-centered method, improve the method if possible, and obtain a better theoretical understanding of SR to explain this phenomenon.

TABLE 3
e_r with maximum buffer schedule, $K = 4$ and $J_1 = 4$.

$\log_2 {}^pN_0^V$	6	5	4	3	2	1
$N = N_K$	1024	512	256	128	64	32
e_r	1.02	1.05	1.11	1.25	1.4	1.9

6. A proposed asymptotically exact segmental refinement method. This section proposes a new SR data model or method that, given the observations of section 5, would efficiently provide an asymptotically exact solver at the expense of some loss of parallelism. This method is simple and most likely overly conservative, but it lends itself to simple analysis. We show how a communication complexity model and corresponding machine memory model can be used to define an SR data model (SR eliminates a class of communication on fine grids). We conclude with some general observations on the SR technique and potential future work in composing SR methods to generate multilevel SR methods.

6.1. A multigrid V-cycle communication model. SR inherits the computational depth of conventional distributed memory multigrid, and the coarse grid solves are identical. Consequently, a more refined complexity model is required to distin-

guish the communication characteristics of SR from those of conventional multigrid. SR has been analyzed in terms of the communication patterns and savings and the extra computation costs for a two-level method [12, 11]. In addition, the memory complexity analysis with a $\log^D$ term for the SR buffer cells memory complexity has been studied [6, section 8.7]. This section proposes a new SR data model that we posit is asymptotically exact and an abstract memory model that resolves the communication that is eliminated by this new method.

We define two types of multigrid communication: vertical intergrid (c_V) and horizontal intragrid (c_H) communication. A $V(2,2)$-cycle requires eight message phases or bulk synchronous communication steps: six horizontal phases for the four smoothing steps, the residual, and the τ correction, and two vertical phases for restriction and prolongation. Our model focuses on these communication phases and the "distance" of each message phase.

6.2. A memory model. This section defines a natural memory model for our proposed new SR method. We define a "word" of data as a small patch of cells (e.g., $2^D - 32^D$ cells or a high order finite element) and assume that each "process" computes on one data word. Consider a two-level memory model with Q words of fine grid memory, partitioned into $\sqrt{Q}$ partitions, each of size $\sqrt{Q}$. We define near communication as communication between processes within a memory partition and far communication as communication between memory partitions.

6.3. Proposed asymptotic segmental refinement data model. The observations in section 5 suggest that a data model that increases ${}^pN_0^V$ with the number of levels, and perhaps adds a quadratic term to (5), would be asymptotically exact. One approach is to keep ${}^pN_0^V$ constant and determine an appropriate buffer schedule, but this would be less reliable and less natural. It is notable that a nonasymptotic model is of practical use since, as an example, fixing $K = 5$ reduced the size of the conventional (full communication) solver by a factor of 32K (2^{KD}), which is a significant constant.

We propose a data model that provides sufficient accuracy for an asymptotically exact solver by extending the parallel octree of the coarse grids to the entire multigrid hierarchy, using $\sqrt{Q}$ processes in each SR subdomain and setting the size of the transition level to fit into one memory partition. With K SR levels, this model has $M = 2K$ multigrid levels and $K+1$ conventional levels. Figure 3 shows a 1D example of this data model with two SR levels, $Q = 16$, and an SR patch length $N_0^V = 4$ where a word is one cell.

6.4. Communication complexity. We use the multigrid V-cycle communication model of section 6.1, the machine model of section 6.2, and the SR data model of section 6.3 to analyze the communication complexity of the SR method. We ignore FMG prolongation because there are M FMG prolongations as opposed to $\mathcal{O}(M^2)$ V-cycle restrictions and prolongations. Thus, FMG processes a V-cycle once on the finest grid, twice on the first coarse grid with $1/2^D$ as many active processes, and so on for $M+1$ levels and $(M+1)\cdot\frac{M}{2} \approx \frac{M^2}{2}$ grid visits total. This leads to a computation depth of $log_2^2(N)$ for FMG. There are six c_H communication phases and two c_V phases per grid visit with one visit on the finest grid, two on the first coarse grid, and so on with M visits to the coarsest grid. Correspondingly, there are about $M^2/8 = M^2/2^D$ visits on the fine (SR) half of the grid hierarchy and $3\cdot M^2/8$ on the coarse (conventional) half of the grid hierarchy. We ignore vertical data locality and assume that all vertical communication is far communication in the finest K levels

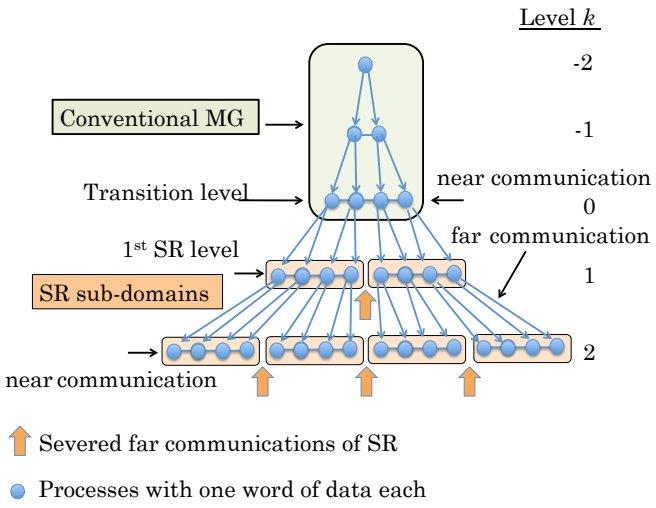

FIG. 3. *1D example of the asymptotic data model.*

and that all communication is near communication on the coarsest $K+1$ levels.

6.4.1. 3D bisection bandwidth. Here we consider a four-level memory model generated by bisecting the memory and domain of the current model. The communication complexity between these two partitions is bisection bandwidth. In the case of conventional multigrid, the highest order term of bisection bandwidth is from the ghost cell exchange on the finest grid. As a result, on a 3D cube with N cells in each dimension, $\mathcal{O}(N^2)$ cells (the area of the face between the two partitions) times the length of the ghost region ($\mathcal{O}(1)$) is communicated $\mathcal{O}(1)$ times resulting an a communication complexity of $\mathcal{O}(N^2)$.

The highest order term in the SR bisection bandwidth complexity is from the buffer region exchange on the transition level. Assume the number of buffer cells required is quadratic in K, because our data in section 5 suggests this is required for an asymptotically exact solver. The "area" of data sent in this buffer cell exchange is $\mathcal{O}({}^pN_0^2) = \mathcal{O}(\sqrt{N}^2) = \mathcal{O}(N)$, has a depth K^2, and is executed $\mathcal{O}(\log_2 N)$ times. Thus, the communication complexity is $\mathcal{O}(N \cdot K^2)\mathcal{O}(\log_2 N) = \mathcal{O}(N \cdot \log_2^3 N)$. SR reduces the bisection communication requirements from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log_2^3 N)$.

6.4.2. Near and far communication complexity. We compare the near/far communication complexity of conventional and SR multigrid. Table 4 tabulates the communication complexity of conventional and SR multigrid with $M+1$ levels. The coarsest $K+1$ levels of both solvers use the same FMG solver on one memory partition (Figure 3). There are six c_H communication phases and two c_V phases per grid visit. Consequently, the removal of far horizontal communication complexity is the distinguishing characteristic of SR.

6.4.3. Conclusion and future work. Any one SR data model removes horizontal communication at some level of the memory hierarchy—the $6c_H$ term in far communication in Table 4, and at the arrows in Figures 1 and 3. Communication,

TABLE 4

Communication phases ($\times \log_2^2 N/8$) of conventional distributed memory multigrid and segmental refinement multigrid.

Communication type	Near	Far
Coarse grids	$3 \cdot (6c_H + 2c_V)$	0
Conventional fine grids	$6c_H$	$6c_H + 2c_V$
SR fine grids	$6c_H$	$2c_V$

in some memory model, is used only for the vertical operator restriction and prolongation, which have tree-like graphs. Tree algorithms are efficient for the global communication required for the solve of an elliptic system. The critical observation of SR is that horizontal communication of traditional parallel multigrid is used for local processes and is not global; hence "far" communication is potentially not necessary.

We have investigated two SR data models that are "two-level" in that there is one "transition" level between a conventional coarse grid solver and decoupled finer grids. An additional step is to compose these two models, by using the method in this section as the coarse grid solver for the method in section 4, and create a three-level SR method. We expect an asymptotically exact "multilevel" SR method that starts with the method in this section as a "coarse grid" solver and reduces the size of ${}^pN_k^V$, by a factor of two on each finer level, resulting in just one process per SR subdomain on a fine level, thereby recovering more parallelism, and continuing with the method in section 4 on each process. This is a subject for future work.

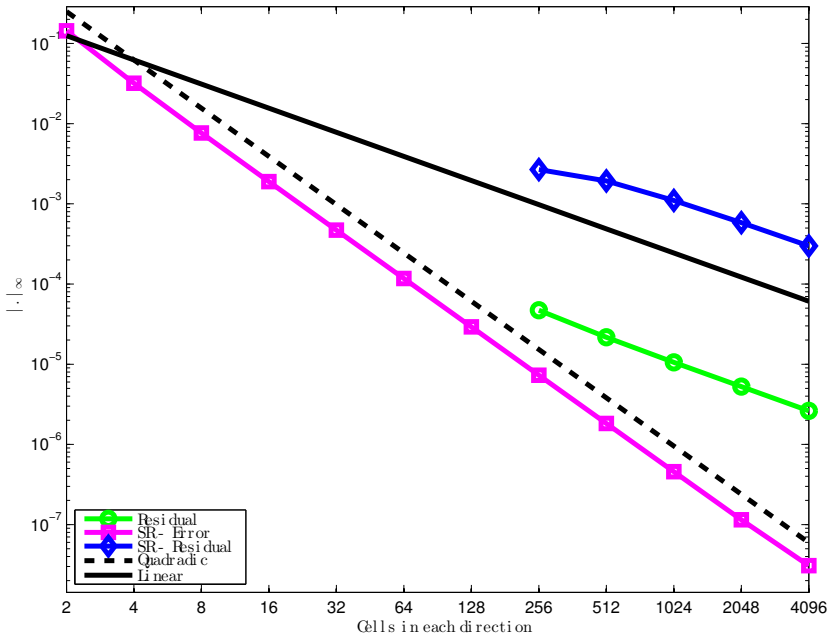

FIG. 4. *Convergence verification: F-cycle with V(2,2) cycles, Laplacian* $u = \left(x^4 - L_i^2 x^2\right)$, $L = (2, 1, 1)$, 128^3 *cell/core.*

7. Timing studies. The main purpose of this paper is to experimentally investigate the asymptotic mathematical behavior of the SR technique, but this section

presents some performance data data on *Edison*, the Cray XC30 at NERSC with up to 64K cores using the problem in section 5.1, which provides some computational context for the methods developed here. We use eight of the 12 cores on each socket and thus utilize 96K cores at scale, or about 75% of the machine, and investigate weak scaling with 128^3 and 32^3 cells per core on the fine grid, with four and three SR levels, respectively, and ${}^pN_0^V = 8$ and 4, respectively. The solver is preloaded with one solve, which verifies accuracy, followed by eight timed solves for the 128^3 cells per core case and 512 solves for the 32^3 cells per core case to normalize times.

Figure 4 plots the infinity norm of the error and residual in the FMG solve and verifies that our solvers are asymptotically exact and that 2nd-order accuracy is achieved in the solution and 1st-order in the residual. The residuals for SR are larger than those of the conventional method but are still 1st-order convergent. Figure 5 plots the solve times for the SR solver and the conventional multigrid solvers and shows modest gains in scalability with SR. The solve times for a V-cycle solve with a relative residual tolerance of 10^{-4} are also shown. Figure 6 demonstrates the stagnation in error reduction with a V-cycle solver, converged to a constant residual reduction, and that SR is maintaining perfect 2nd-order accuracy.

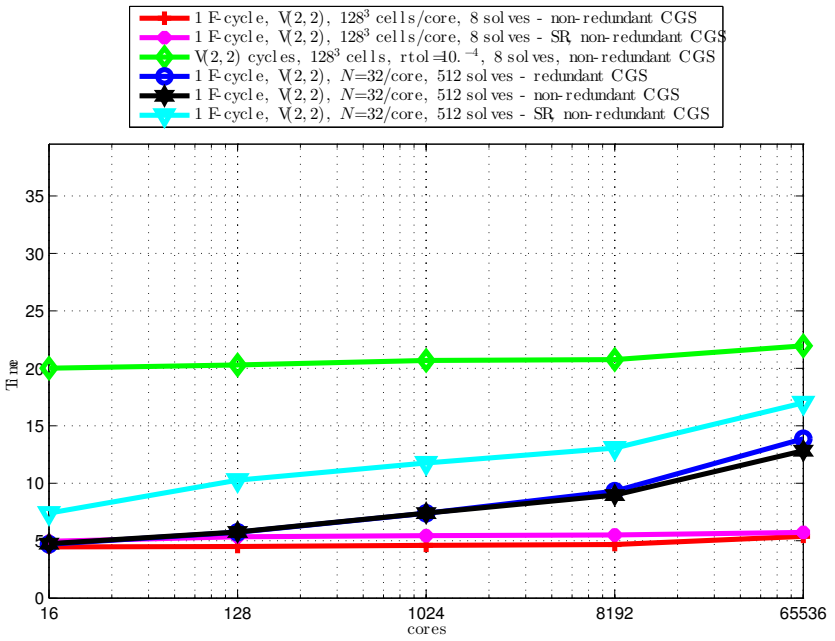

FIG. 5. *Weak scaling: Solve times (8 solves) on Edison, Laplacian* $u = \left(x^4 - L_i^2 x^2\right)$, $L = (2,1,1)$, 128^3 *cell/core.*

Our parallel multigrid implementation has the capability to compute coarse grids redundantly, where all processors are active on all levels redundantly computing coarse grid corrections, or processors can be left idle on coarse grids. Redundantcoarse

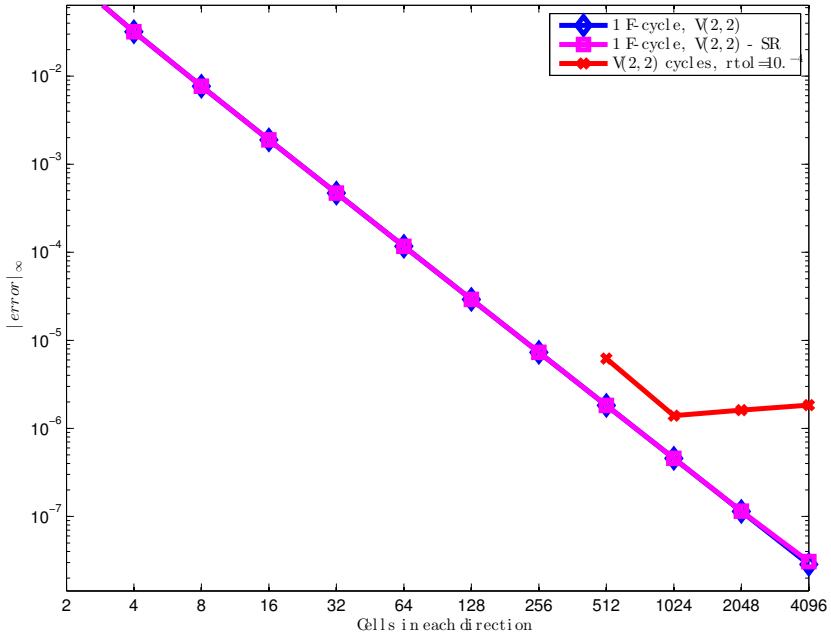

FIG. 6. *Errors of SR and conventional solver (same) and V-cycles with relative tolerance of* 10^{-4}*, Laplacian* $u = \left(x^4 - L_i^2 x^2\right)$*,* $L = (2, 1, 1)$*,* 128^3 *cell/core.*

grid solves result in a "butterfly" communication pattern, and the idle processors result in a tree communication pattern. This approach has the advantage of requiring no communication in the prolongation phase, hence reducing the number of bulk synchronous communication steps, at the expense of sending more data with more messages overall. We observe in Figure 5 that redundant coarse grid solves are slightly slower, which suggests that a larger number of messages costs more than the savings in the number of bulk synchronous phases. However, this effect is small, and our code is not instrumented to understand this phenomenon effectively. Understanding the use of redundant coarse grid solves at extreme scale is the subject of future work and is likely to be more important on future extremes scale machines (e.g., we only just observe a difference in our largest runs and Edison as a very good *Aries* network).

8. Conclusions. We continue the work of [7], with the first published multilevel numerical results of the SR multigrid method. We demonstrate that SR can maintain the semantics of textbook efficient multigrid FMG-FAS with processing that is more attractive on modern memory-centric architectures than conventional distributed memory multigrid by decoupling fine grid processing, which improves data locality, amortizes latency costs, and reduces data dependencies. We have experimentally investigated the asymptotic behavior of SR and have found an accuracy dependence not previously recognized. We analyzed the communication complexity, with a two-level memory model, of an SR data model, where we show that the method

removes horizontal communication as defined by the memory model. The degree to which the memory model, on which any given SR method removes communication, is a useful performance model for any given machine can be used as a metric for the potential efficacy of the method. We experimentally verify that our SR data model is an asymptotically exact solver on 64K cores of a Cray XC30 and provide timing and scaling data.

We have observed modest improvement in scaling with SR with a simple data model that supports only a few SR levels. Future work includes developing SR data models that accommodate more levels of the memory hierarchy, testing on machines with deeper memory hierarchies, and fully exploiting SR's data locality with, for instance, loop fusion [16]. A vertex-centered discretization and high order I_0^1 prolongation would be of interest to better understand the asymptotic complexity of SR and corroborate the observation of the accuracy dependence on ${}^pN_0^V$. SR may be particularly sensitive to the order of prolongation because it is used to set the "frozen" ghost cells in the SR buffer region. Further work involves extending the application of SR to more domains, such as variable coefficient and nonlinear problems and unstructured grid problems.

All source code, data, and run and parsing scripts used to produce this paper are publicly available [1].

Acknowledgment. We would like to thank Achi Brandt for his generous guidance in developing these algorithms.

REFERENCES

[1] M. Adams, *SRGMG*, https://bitbucket.org/madams/srgmg, 2016.
[2] M. F. Adams, R. Samtaney, and A. Brandt, *Toward textbook multigrid efficiency for fully implicit resistive magnetohydrodynamics*, J. Comput. Phys., 229 (2010), pp. 6208–6219, doi:10.1016/j.jcp.2010.04.024.
[3] R. Bank and T. Dupont, *An optimal order process for solving finite element equations*, Math. Comp., 36 (1981), pp. 35–51.
[4] A. Brandt, *Multi–level adaptive technique (MLAT) for fast numerical solution to boundary value problems*, in Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics, H. Cabannes and R. Teman, eds., Lecture Notes in Phys. 18, Springer-Verlag, Berlin, 1973, pp. 82–89.
[5] A. Brandt, *Multi-level adaptive solutions to boundary value problems*, Math. Comput., 31 (1977), pp. 333–390.
[6] A. Brandt, *Multigrid Techniques:* 1984 *Guide with Applications to Fluid Dynamics*, GMD–Studien Nr. 85, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, West Germany, 1984.
[7] A. Brandt and B. Diskin, *Multigrid solvers on decomposed domains*, in Domain Decomposition Methods in Science and Engineering: The Sixth International Conference on Domain Decomposition, Contemp. Math. 157, American Mathematical Society, Providence, RI, 1994, pp. 135–155.
[8] A. Brandt and O. E. Livne, *Multigrid Techniques*, SIAM, Philadelphia, 2011, doi:10.1137/1.9781611970753.
[9] N. Dinar, *Fast Methods for the Numerical Solution of Boundary Value Problems*, Ph.D. thesis, Weizmann Institute of Science, Rehovot, Israel, 1979.
[10] R. P. Fedorenko, *A relaxation method for solving elliptic difference equations*, Z. Vycisl. Mat. i. Mat. Fiz., 1 (1961), pp. 922–927; also in USSR Comp. Math. Math. Phys., 1 (1962), pp. 1092–1096.
[11] M. Mohr, *Low communication parallel multigrid*, in Euro-Par 2000 Parallel Processing, A. Bode, T. Ludwig, W. Karl, and R. Wismüller, eds., Lecture Notes in Comput. Sci. 1900, Springer-Verlag, Berlin, Heidelberg, 2000, pp. 806–814, doi:10.1007/3-540-44520-X_111.
[12] M. Mohr and U. Rude, *Communication Reduced Parallel Multigrid: Analysis and Experiments*, Tech. report 394, University of Augsburg, Augsburg, Germany, 1998.

[13] R. V. Southwell, *Relaxation Methods in Engineering Science*, Oxford University Press, Oxford, UK, 1940.

[14] J. L. Thomas, B. Diskin, and A. Brandt, *Textbook multigrid efficiency for the incompressible Navier–Stokes equations: High Reynolds number wakes and boundary layers*, Comput. Fluids, 30 (2001), pp. 853–874.

[15] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, San Diego, 2001.

[16] S. Williams, D. D. Kalamkar, A. Singh, A. M. Deshpande, B. Van Straalen, M. Smelyanskiy, A. Almgren, P. Dubey, J. Shalf, and L. Oliker, *Optimization of geometric multigrid for emerging multi- and manycore processors*, in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '12, IEEE Computer Society Press, Los Alamitos, CA, 2012, pp. 1–11.