Published in *Wave Motion* 98, 2020, #102649
https://doi.org/10.1016/j.wavemoti.2020.102649

CADD: a seamless solution to the Domain Decomposition problem of subdomain boundaries and cross-points

Dan Gordon[*] Rachel Gordon[†]

Abstract

The solution of wave problems using Domain Decomposition (DD) requires that the subdomain boundaries should be virtually non-existent, so that waves are not affected by the boundaries. This is a primary problem in DD, and it intensifies in the case of cross-points at which three or more subdomains meet. This topic has received a lot of attention in recent years, with special treatment of cross-points. This paper explains and demonstrates that this problem does not exist in Component-Averaged Domain Decomposition (CADD). CADD is implemented here with the authors' CARP-CG algorithm, but it is shown that other implementations are also possible. The reason for the non-existence of this problem in CARP-CG is that in some superspace of the problem space, CARP-CG is mathematically equivalent to the CG acceleration of the Kaczmarz algorithm with cyclic relaxation parameters, applied to a single linear system. Due to its advantages, CARP-CG was adopted by some geophysics researchers as the solver of the Helmholtz and the elastic wave equations for full waveform inversion (FWI).

Keywords. *C*ADD; CARP; CARP-CG; CGMN; component-averaged domain decomposition; elastic wave equation; Kaczmarz; Helmholtz equation; linear equations; parallel processing; partial differential equations; sparse systems; wave problems.

1 Introduction

A major problem in solving wave problems using domain decomposition (DD) is that of merging the solutions of subdomains across subdomain boundaries. The problem is to create the effect that the boundaries are virtually nonexistent so that waves traveling across the boundaries are unaffected. A recent approach to this problem is the use of perfectly matched layers (PMLs). PMLs, introduced as absorbing boundary conditions in [4, 5], are widely prevalent in wave problems, where they simulate an infinite domain. In DD, PMLs have been adapted and used at subdomain boundaries to create the effect that waves coming from either direction are unaffected by the subdomain boundaries. This requirement creates an even harder problem at so-called cross-points, where three or more subdomain boundaries meet; see [7, 14, 16, 29, 31, 32]. PMLs as subdomain boundaries have three problems: they widen the subdomain boundaries significantly, they

[*]Dept. of Computer Science, University of Haifa, Haifa 34988, Israel. Email: `gordon@cs.haifa.ac.il`
[†]Dept. of Aerospace Engineering, The Technion–Israel Inst. of Technology, Haifa 32000, Israel. Email: `rgordon@g.technion.ac.il`

require problem-specific equations at the PMLs, and they need special treatment at cross-points – see [16]. In the following, the method of "component-averaged domain decomposition" (CADD) is explained, and it is shown that in contrast to the use of PMLs, CADD adds only two virtual grid points at subdomain boundaries, the operations at the boundaries are unrelated to the equations, and there is no special treatment at cross-points.

CADD is implemented with the authors' CARP and CARP-CG algorithms [18,20] to DD. CARP is a block-parallel version of the Kaczmarz algorithm (KACZ) [26], which is a sequential algorithm, and CARP-CG is the CG acceleration of CARP. Both CARP and CARP-CG are ideally suitable DD as follows: the domain is divided into subdomains by boundaries passing between grid points. Then, each subdomain is handled by a different processor, with all processors working in parallel. After one such operation, neighboring subdomains exchange data about shared variables, and the parallel processing is resumed until convergence. Details are given in the next section.

CARP-CG is particularly efficient on two types of problematic linear systems:

- *Discontinuous coefficients.* KACZ inherently "normalizes" the equations by dividing each equation by the L_2-norm of its coefficients, and this preprocessing has an equalization effect on the coefficients of different equations (see [21] for the general effect of this preprocessing).
- *Large off-diagonal elements.* Assume the original system is $Ax = b$, with the equations already normalized for KACZ. Then, according to [6], KACZ is actually successive overrelaxation (SOR) on the system $AA^Ty = b,\ x = A^Ty$, and when A is normalized, all the diagonal elements of AA^T are 1, and the off-diagonal elements are < 1 (assuming no two rows of A are colinear); see [19]. Typical examples of such systems arise in convection-dominated partial differential equations (PDEs) [20, figs. 8,9] and the Helmholtz equation at high frequencies [22].

The rest of the paper is organized as follows. The next section explains KACZ, CARP and CARP-CG in detail and presents some previous work with CARP-CG. Section 3 demonstrates the effectiveness of CARP-CG on some cases that may require special treatment at the boundaries and cross points with other DD methods. Section 4 shows that KACZ can be replaced in CADD by several other solvers and Section 5 concludes with an overview of the paper.

2 Background

2.1 KACZ, CARP, and CARP-CG

KACZ is best described by its simple geometric explanation: starting from some initial point in the solution space $\mathbb{R}^n$ or $\mathbb{C}^n$, the current iterate is repeatedly projected orthogonally towards the hyperplane defined by one of the system's equations. Usually, the projections follow cyclically the given order of the linear system.

We consider an $m \times n$ linear system

$$Ax = b, \tag{1}$$

where $b = (b_1, \dots, b_m)^T$, and the jth row of A is denoted by a_j. Denote by $x^0, x^1, \dots$ the sequence of KACZ iterates, and suppose that x^{k+1} is obtained from x^k by projecting x^k onto the jth hyperplane, i.e.,

$$x^{k+1} = x^k + \lambda \frac{b_j - \langle a_j, x^k \rangle}{\|a_j\|_2^2} a_j, \tag{2}$$

where $0 < \lambda < 2$ is a relaxation parameter. The limitation on λ is essential for convergence. We can save some computation time by initially dividing each equation $\langle a_j, x \rangle = b_j$ by $\|a_j\|_2$, thus avoiding the division by $\|a_j\|_2^2$ at every step; this is sometimes called normalizing the equations. Each cycle of projections is called a KACZ *sweep*. The relaxation parameter λ need not be constant, and if equation j has a fixed relaxation parameter λ_j, then the projections are said to be done with *cyclic relaxation*. An important property of KACZ with cyclic convergence is that it always converges to some point, provided all cycles are complete, even if the linear system is inconsistent; see [12, Thm. 1.1]. In the field of biomedical image reconstruction, KACZ is also known as ART (algebraic reconstruction technique); see [24, 25].

Thus, by its mathematical definition, KACZ is inherently sequential. However, for very large problems, it is essential to parallelize it. The standard way of parallelizing KACZ, called multicoloring (MC), consists of identifying blocks of equations such that in each block, the coefficient vectors are pairwise orthogonal; and then, in each block, the projections can be done in parallel. However, the blocks have to be handled sequentially.

In [18], the authors introduced a block-parallel implementation of KACZ called CARP (component-averaged row projections), described as follows. Given a linear system $Ax = b$, where A is $n \times n$ and regular, CARP operates as follows: the system is divided into m blocks of equations, $A^\ell x = b^\ell$, $1 \le \ell \le m$, and the blocks may even overlap. For every variable x_i belonging to two or more blocks, one of these blocks is assigned to be the "owner" of x_i. The method of this assignment depends on the particular application, and it is most easily explained within the concept of domain decomposition: assume the domain is partitioned by boundaries passing *between* grid points. In theory, overlapping partitionings are also possible, but this only complicates matters and there is no advantage to it. This partitioning forms a division of the grid points into subdomains, and all the equations whose stencil centers appear in the same subdomain form a single block.

The owner block of a variable x_i is determined as follows: consider the grid point g associated with x_i, then the owner block of x_i is the block associated with the subdomain containing g. Note that this definition applies only in the case of disjoint blocks. We will define a variable x_i as a *boundary variable* if it appears in equations belonging to two or more blocks.

CARP now operates in parallel on the blocks as follows. We assume that every block of equations has an assigned processor operating on the equations of that block. Starting from some initial value x^0, every block owning a boundary variable x_i^0 sends the value of x_i^0 to all blocks which have some equation containing x_i^0. The copies of x_i^0 are called the *clones* of x_i^0. Blocks which receive such copies operate on the clones and not on the original variables. If x_i^0 appears in k blocks, then it remains as x_i^0 in its owner block and, in $k-1$ blocks, x_i^0 is replaced by variables called $x_{i,2}^0, \dots, x_{i,k}^0$. For convenience, we also refer to x_i^0 as $x_{i,1}^0$. As a result, the blocks of the modified equations have no common variables, so they can be processed in parallel.

The following four steps are now repeated until some stopping criterion is satisfied:

1. All block processors, in parallel, perform some fixed number of KACZ iterations on the *modified* equations of their assigned block.
2. The new value of every clone is transferred to the owner of the clone's original variable.
3. Every block, for every owned boundary variable, computes the average of the boundary variable and all its clones, and sets this average as the new value of the boundary variable. In terms of the above notations, if $x^0_{i,1},\dots,x^0_{i,k}$ were changed by the KACZ operations to $y^0_{i,1},\dots,y^0_{i,k}$, respectively, then the new value of x^0_i is $x^1_i = (\sum_{j=1}^{k} y^0_{i,j})/k$.
4. Every block processor, for every owned boundary variable, transmits the variable's new value to its neighboring blocks as the new clone(s).

Note that inter-processor communications are needed only between pairs of blocks associated with adjoining subdomains. Furthermore, the parallel processing of all the blocks differs from most DD methods, which do not process adjacent blocks in the same time step.

Consider now the superspace of the problem space consisting of all the variables and all the clones. An *Averaging Lemma* [18, 20] shows that the averaging operations are mathematically equivalent to KACZ projections in the superspace (with relaxation parameter = 1). If, as before, some variable has $k-1$ clones, then it takes $k-1$ KACZ projections in the superspace to set all k values equal to their average. Hence, it takes k equations in the superspace to average the variable and its clones. It follows that the total number of equations in the superspace is $m = n + \ell$, where ℓ is the total number of clones. Clearly, the number of variables in the superspace is also m.

In a landmark paper, Björck and Elfving [6] showed that if a forward sweep of KACZ (with a fixed relaxation parameter) is followed by a backward sweep (i.e., the projections are done in reverse order), then the resulting iteration matrix is symmetric and positive semi-definite. Therefore, the double KACZ sweep can be accelerated using the conjugate gradients (CG) algorithm. The iteration matrix is not calculated; instead, whenever it has to be multiplied by a vector, the calculation is done by performing a suitable KACZ double sweep. This algorithm is called CGMN in [6].

The CG acceleration of CARP is obtained in [20] as follows. Firstly, the construction of CGMN is extended to KACZ with cyclic relaxation parameters (and called CGMNC). It is then shown that the superspace formulation of CARP can be accelerated by CG. This means that in the regular space, CARP can also be accelerated by CG by running CARP in a double sweep. On one processor, CARP-CG and CGMN are identical. Now, based on the cyclic convergence of CARP and (theoretical) convergence of CG, it follows that CARP-CG always converges, even on inconsistent systems.

The above exposition of CARP-CG provides a mathematical explanation as to why the subdomain boundaries are virtually non-existent and do not affect the solution: in the superspace of the variables and the clones, CARP-CG is a CG acceleration of KACZ with cyclic relaxation parameters.

The following question arises: what, then, is the difference between KACZ and CARP? The answer is that KACZ is a sequential algorithm, whereas CARP is a parallel "divide and conquer" algorithm. The same applies to CARP-CG and CGMNC.

One should note that in implementing CARP-CG, the optimal behavior of the algorithm depends on several factors: the number of processors available, their efficiency, the efficiency of the inter-processor communications, the amount of memory available to each processor, and the time required by each processor to complete one iteration. Clearly, if too many processors are used, then the inter-processor communications would take more time, and if too few processors are used, then the overall computation time would be slow.

Another factor affecting the rate of convergence is the partitioning method of the domain, and to complicate matters further, the optimal partitioning is problem-dependent, so there is no universally-optimal partitioning. Due to all these different parameters and their interaction, it is impossible to provide general guidelines for determining how the system of equations should be partitioned. The value of λ also affects the rate of convergence, but its optimal value is easily determined after a few iterations. A study of these parameters can be found in [18, §4.5.2] and [20, §6.2].

2.2 Some previous work with CARP-CG

In [20], the speedup of CARP with CG was developed and an extended comparison was made with classical methods on a variety of elliptic PDEs. It was found that when the convection term was small, then CARP-CG did not perform as well as other methods, such as (restarted) GMRES and Bi-CGSTAB, with and without various preconditioners. However, just by increasing the convection term on the same equation caused CARP-CG to converge faster than all the other methods, and even faster than all the other methods on the low-convection PDE; see Figures 8 and 9 in [20], which differ only in the size of the convection factor.

After CARP-CG was used for the Helmholtz equation with high frequencies in [22], it was used for the problem of "full waveform inversion" (FWI) in exploration geophysics [35,36]. High order schemes for the Helmholtz equation are essential for the Helmholtz equation due to the so-called pollution effect [1,2]. Several such schemes were studied in [22], and a new sixth order compact scheme for variable wave numbers was developed in [34]. A compact scheme is one in which the stencil is of size 3×3 in 2D and $3\times 3\times 3$ in 3D. Such a scheme is particularly useful when CARP-CG is used, because it does not require more data transfer across subdomain boundaries than regular second order schemes. CARP-CG was also used in [23], where a new approach to absorbing boundary conditions for wave problems was developed.

Li et al. [28] successfully applied CARP-CG to the elastic wave equation in the frequency domain, using a fourth-order accurate staggered grid. In a subsequent paper, [27], an optimal fourth-order staggered grid for the viscoelastic wave equation was developed and tested, using CARP-CG.

In a totally different direction, CARP-CG was used in [15], in combination with multicoloring, as the parallel linear solver for difficult interior eigenvalue problems arising in graphene modeling.

CARP-CG was also incorporated as one of the algorithms implemented in the exascale sparse solver repository [33].

3 Sample results with subdomain boundaries and cross-points

In this section we present test problems which demonstrate visually that there are no problems with subdomain boundaries or cross-points when using CARP-CG. The examples, which include convection-dominated and Helmholtz problems, also demonstrate the advantages of high order schemes and the gradient absorbing boundary conditions [23] for the Helmholtz problem. The examples include two convection dominated PDEs with a known solution, one Helmholtz example with a known solution, and one Helmholtz equation with three different wave numbers without a known solution.

Evaluation methodology: We describe here the methodology only for Problems 1 and 2. The other cases will be explained within the text of the problems. In the first two problems, we have a known solution which serves as the Dirichlet boundary conditions. We first made a few test runs to determine the optimal (or near optimal) value of the relaxation parameter λ. From examples in previous works, it was shown that small changes in λ have very little effect on the rate of convergence. We then ran the solver with a large bound on the number of iterations, and every 10 iterations we calculated the relative residual (rel-res) and the relative error (rel-err). This way we determined the best rel-err that can be obtained and the required number of iterations to achieve it (maxit). We then ran the solver without calculating rel-err and rel-res for maxit iterations to obtain the time of the computation. In problems 1 and 2 we displayed cross-sections of the solution and the pointwise absolute value of the relative error, i.e., $\|u_t - u_c\|/\|u_t\|$, where u_t and u_c are the true and calculated values, respectively, at a grid point of the cross-section.

We wish to stress the difference between rel-res and rel-err. When rel-res reaches very low values, it means is that you have a good solver for the system of equations. But if the linear system does not represent the problem with sufficient accuracy, then the computed solution will be far from the true solution. This can be seen clearly in [22, Problems 1 & 2], which shows rel-res and rel-err results on two problems, with 2nd, 4th and 6th order accurate finite difference schemes.

3.1 Convection-dominated problems

The two problems in this subsection are Problems 2 and 6 from [20, §4]. These problems were previously studied extensively by other authors (see the references in [20]). In both problems, we used the standard second order finite difference scheme, i.e., a 7-point stencil in 3D.

Fig. 1 shows a $2\times2\times4$ partitioning of a domain, with two partitions in the x- and y-directions, and four partitions in the z-direction. This domain has three cross-points, each bordering eight subdomains.

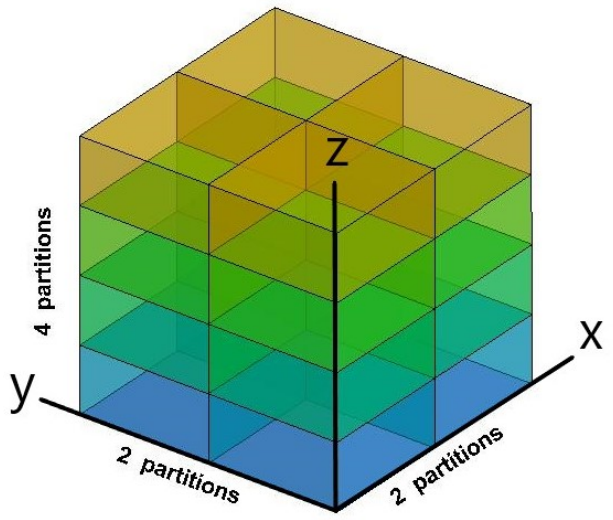


Figure 1: A $2 \times 2 \times 4$ partitioning of the unit cube.

Problem 1. The PDE is

$$\Delta u + 1000 e^{xyz}((u_x + u_y - u_z)) = F, \tag{3}$$

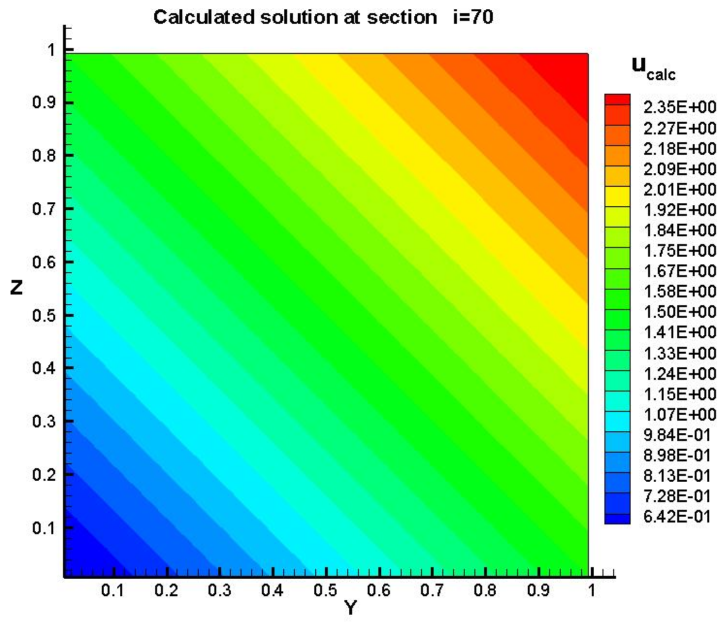
where $\Delta u = u_{xx} + u_{yy} + u_{zz}$, the 3D domain is $[0,1]^3$, the preassigned solution is $u = x + y + z$, and the RHS F is calculated from Eq. (3): $F = 2(x+y+z) + 1000e^{xyz}(2z+1)$. The Dirichlet boundary conditions are calculated by using the values of the preassigned solution at the boundaries. The domain is divided into a grid of 142^3 grid points, and partitioned into subdomains as in Fig. 1.

Computation details:

- Relaxation parameter: $\lambda = 1.60$.
- rel-err = 3.77E-14.
- rel-res = 1.98E-14.
- No. of iterations: 500.
- Time: 24.76 sec.

In the two figures below we display only the image at the inner points of the grid, i.e., without the Dirichlet boundary. Hence, both images are slightly smaller than 1×1.

Figure 2 shows the calculated solution (top) and the relative error between the analytic and calculated solution (bottom), both at the cross-section $i = 70$, which is adjacent to the subdomain boundary at $x = 1/2$ and to three cross-points. The analytic solution is not shown because it is visually indistinguishable from the calculated solution. The bottom image also shows the subdomain boundaries and cross-points. Note that there is no visible effect of the boundaries and cross-points on the calculated solution or the error.



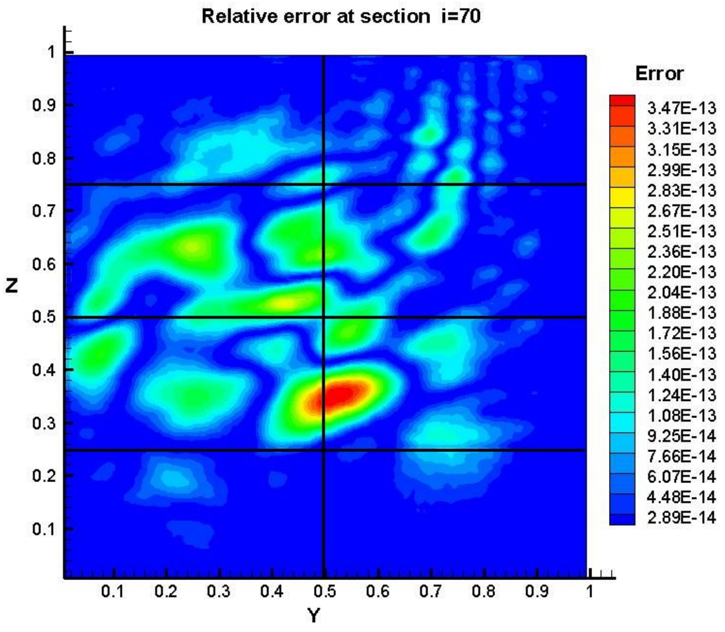


Figure 2: Solution (top) and relative error (bottom) for Problem 1 at cross-section $i = 70$, adjacent to the subdomain boundary at $x = 1/2$ and to 3 cross-points. The black lines in the bottom image show the intersection of the cross-section with the subdomain boundaries. Neither image shows any effect of the subdomain boundaries and cross-points.

Problem 2. The PDE is

$$\Delta u - 1000((1-2x)u_x + (1-2y)u_y + (1-2z)u_z) = F. \tag{4}$$

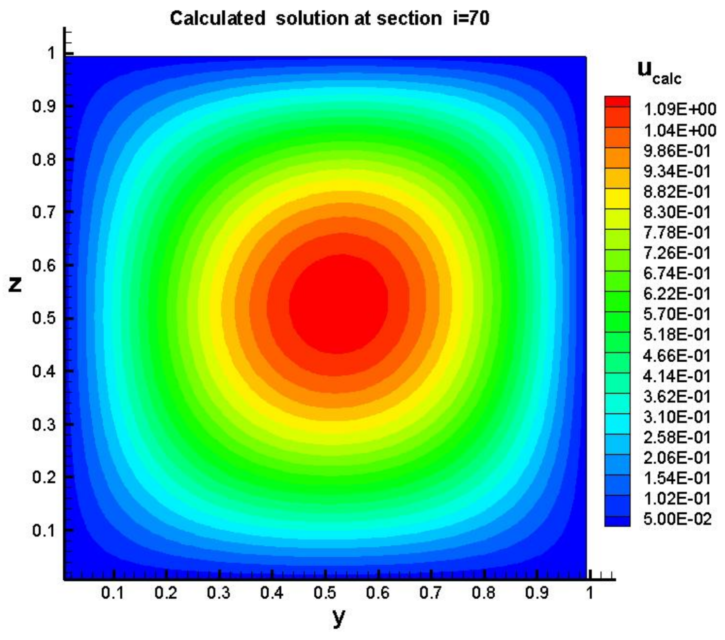
The preassigned solution is $u = e^{xyz}\sin(\pi x)\sin(\pi y)\sin(\pi z)$, and the 3D domain is $[0,1]^3$. The RHS F is calculated from Eq. (4), and the Dirichlet boundary conditions are calculated by using the values of the preassigned solution at the boundaries (i.e., zero on all boundaries). The domain was divided into a grid of 140^3 grid points and partitioned into subdomains as shown in Fig. 1.

Computation details:

- Relaxation parameter: $\lambda = 1.60$.
- rel-err = 7.93E-5.
- rel-res = 2.48E-14.
- No. of iterations: 220.
- Time: 10.93 sec.

The relative error in this problem is much larger than in Problem 2, but it is reasonable for practical applications. In both cases, the stopping criteria are the same (as explained previously), but Problem 2 is more difficult for CARP-CG than Problem 1. To reach a more accurate solution requires a finer grid and/or a higher-order finite difference scheme, but we decided to display both problems under identical conditions.

Fig. 3 shows a cross-section of the calculated solution (top) and the relative error w.r.t. the analytic solution (bottom) of Eq. 4, taken at the grid point $i = 70$, which is in immediate proximity to the subdomain boundary at $x = 1/2$ and to three cross-points. The analytic solution is not shown because it is visually indistinguishable from the calculated solution. The bottom image also shows the subdomain boundaries and cross-points, which show no effect on either image. As can be seen, there is a distinct region of non-symmetry, probably due to the e^{xyz} factor in the solution.



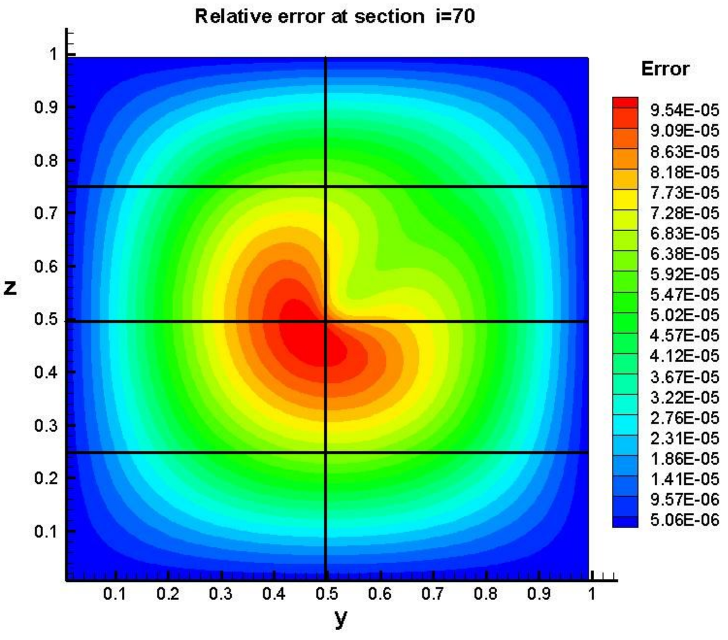


Figure 3: Calculated solution (top) and relative error (bottom) for Problem 2 at cross-section $i = 70$, adjacent to the subdomain boundary at $x = 1/2$ and to 3 cross-points. The black lines show the intersection of the cross-section with the subdomain boundaries. Neither image shows any effect of the subdomain boundaries and cross-points.

3.2 The Helmholtz equation

Problem 3. This problem is similar to the first problem described in [23, §6.1]. We consider the

3D Helmholtz equation on the unit cube: $\Delta u + k^2 u = F$, where

$$F(x,y,z) = \frac{1}{h^2}\cos^5\left(\frac{\pi(x-x_0)}{2hn_c}\right)\cos^5\left(\frac{\pi(y-y_0)}{2hn_c}\right)\cos^5\left(\frac{\pi(z-z_0)}{2hn_c}\right).$$

F is a simulated impact function of compact support, h is the mesh spacing, and n_c is a parameter controlling the size of the compact support. F is taken as zero outside a small cube of $(2n_c+1)^3$ grid points, centered at the impact point. In this problem, $n_c = 8$. Note that F and all its derivatives up to the 4th order vanish at the boundary of the small cube. The division by h^2 is a scaling operation. The frequency was $f = 10$, so the wave number was $k = 20\pi$.

The equation was discretized by using the compact 6th order finite difference scheme for variable wave numbers of [34]. This scheme uses a $3\times3\times3$ stencil. Compact stencils are ideal for CADD methods since they do not require the transfer of more information across subdomain boundaries than the standard 7-point stencil.

It is well known – see [30, §28] – that for a constant wave number k and a given RHS F (of compact support with source at the origin) in an infinite domain in $\mathbb{R}^n$, the solution at a point $x \in \mathbb{R}^n$ is given by

$$u(x) = (G * F)(x) = \int_{\mathbb{R}^n} G(x-y)F(y)dy, \tag{5}$$

where '$*$' denotes the convolution and G is the Green's function for dimension n. For $n = 3$,

$$G(x) = \frac{e^{ik||x||}}{4\pi||x||}.$$

Since F vanishes outside the small cube, we can easily calculate the integral in Eq. (5) to obtain an estimate of the solution. In order to obtain a good approximation of the convolution, we used, on the inner cube, a grid with twice the resolution of the large grid (a finer mesh did not improve the results).

The unit cube was divided by a grid of $169\times169\times169$ points, and the point of impact was taken at $(0.5, 0.5, 0.25)$. For the CARP-CG computation, the domain was divided into 24 subdomains by 23 boundaries perpendicular to the z-axis. This partitioning was chosen because it is more convenient for use with the $3\times3\times3$ stencil. For the boundary conditions, we used the "gradient method" (GM) absorbing boundary conditions (ABCs) of [23]. This ABC is minimalistic in comparison to PML ABCs, as it requires just one extra outer layer of grid points on all sides. This brings the total number of grid points to 171^3.

The concept behind GM is that directional derivatives at a boundary point should be taken in the direction of the *gradient* of the wave field, regardless of the orientation of the boundary at that point. In this problem, the gradient direction points away from the impact point. This concept can be applied to any boundary condition by replacing derivatives normal to the boundary by derivatives in the gradient direction. In this case, GM was applied to the second order BGT boundary condition – see [3].

In this problem, we calculated only the relative residual (rel-res) every 10 iterations, but not the relative error (rel-err). The calculation was stopped at 10,000 iterations since rel-res reached a reasonably low value. Additional details:

- Relaxation parameter: $\lambda = 0.7$.
- rel-err = 1.33E-4.
- rel-res = 2.07E-12.
- No. of iterations: 10,000.
- Time: 2,304 sec.

Fig. 4 shows the computed solution (top) and the relative error (bottom) at the cross-section taken at the 75th grid point in the x-direction. Neither of these images shows any irregularities due to the 23 subdomain boundaries. The bottom image shows faint traces caused by the internal small cube used for the impact function. Also, the bottom figure appears to contain just two colors, while the error scale shows the whole spectrum of colors. The reason for this is that the image actually contains a few pixels whose colors deviate from the two main colors. These pixels are on the upper half of the left and right boundaries. The bottom image also shows some faint traces of reflections from the boundary, but these are at least two orders of magnitude smaller than the wave field. Note that the images are based on the larger grid of 171^3 points, so they deviate very slightly from the unit cube.

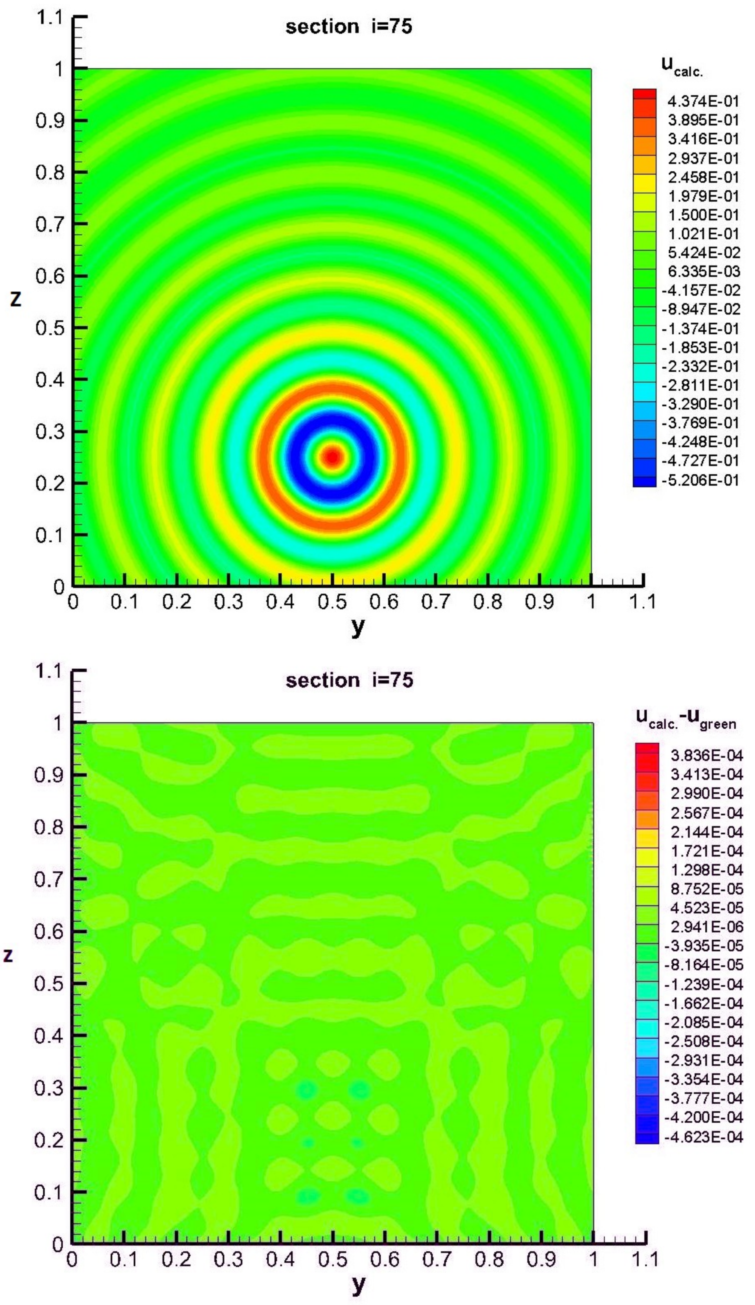


Figure 4: Computed solution (top) and relative error (bottom) for Problem 3 at cross-section $i = 75$. Neither figure shows any effect of the 23 subdomain boundaries orthogonal to the z-axis.

Problem 4. This problem is similar to problem 3, with two exceptions: the impact point is at the center of the region, which is divided by a "wedge" into three sub-regions, each with a different wave number. This is shown in Fig. 5, where it can be seen that the wave numbers are $k = 50$, $k = 200$, and $k = 100$. There is no Green's function in this case, so only the computed wavefield is shown.

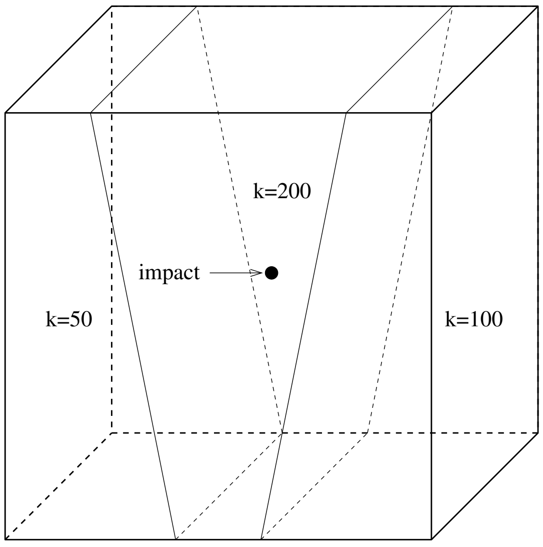


Figure 5: Problem 4: dividing the region into three sub-regions with different wave numbers.

The program was ran with the relative residual displayed every 10 iteration. The computation was stopped at 10,000 iterations because the relative residual was sufficiently small. Other details:

• Relaxation parameter: $\lambda = 1.5$.
• rel-res = 2.86E-11.
• No. of iterations: 10,000.
• Time: 2,650 sec.

Fig. 6 shows the computed wavefield. The image was modified slightly from the original by a 5% reduction of the green color in order improve the visibility of details in the corners. It can be seen that the wavelength of each sub-region corresponds to the wavenumber of that sub-region. Also, the top and bottom parts of the wedge show interference patterns caused by reflections from the boundaries between the sub-regions.

Note that for the computation by CARP-CG, the domain was not partitioned at the boundaries between the three sub-regions, but by 23 boundaries orthogonal to the z-axis. This partitioning shows no visible effect on the wavefield. Also, there are no visible reflections from the boundaries.

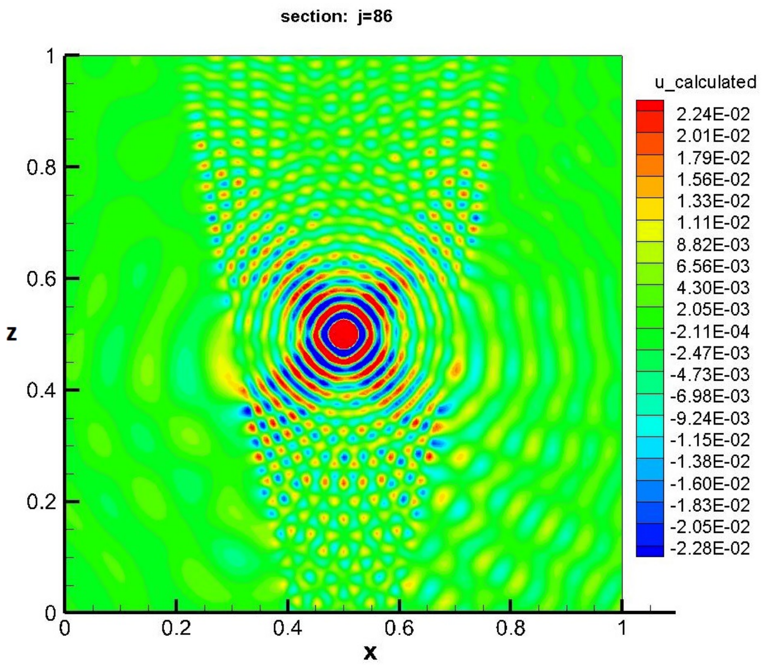


Figure 6: Computed solution to problem 4. The image shows no traces of any effect by the 23 subdomain boundaries orthogonal to the z-axis.

4 Alternatives to KACZ in CADD

A natural question that arises is whether KACZ is the only possible algorithm for CADD. It turns out that several other algorithms can replace KACZ. It was shown in [17] that several algorithms are mathematically identical to KACZ in some (large) superspace of the problem space. These algorithms are the following:

- The Cimmino algorithm [11] (CIMM), which operates as follows: the current iterate is projected towards all the hyperplanes, and then all the projections are averaged to form the next iterate. Similarly to KACZ, these projections can be done with a relaxation parameter.
- Subset Cimmino: the equations are divided into blocks (not necessarily disjoint), and then CIMM is iteratively applied in sequence to all the blocks.
- The CAV algorithm [10]. This algorithm is similar in structure to CIMM, but every projection onto the hyperplane $\langle a_i, x\rangle = b_i$ has its own weight determined by the sparsity of the columns which contain the nonzero elements of a_i.
- The BICAV algorithm [9] is a "subset Cimmino" version of CAV in which the blocks are processed sequentially.
- The String Averaging algorithm [8], can be described as follows: the equations are divided into k blocks, and then, for every block of equations B, independently from the other blocks, KACZ is applied in sequence to the equations of B, starting from the current iterate. This operation produces k results. Then, some operator is applied to the k results, producing the next iterate. If the operator is a weighted average of the results from the blocks, then this algorithm is also

KACZ in some superspace.

It should be noted that since these algorithms are, in essence, KACZ, then their replacement in CADD enables their acceleration by CG, just as in CARP-CG.

In previous work with Y. Censor, we made various comparisons between KACZ, CIMM, CAV and BICAV in the context of image reconstruction from projections. Note that KACZ is often called ART (algebraic reconstruction technique) in this topic. For comparisons between KACZ, CIMM and CAV see [10], Figures 2–13. For various comparisons between KACZ, CIMM, CAV and BICAV see [9], Figures 3, 4, and 6. The gist of these experiments is that CIMM is extremely slower than KACZ, while CAV and BICAV are approximately on par with KACZ.

Regardless of the above relations between KACZ and CIMM, CIMM may be a promising alternative to KACZ for the following reason: Elble et al. [13] showed that by using GPU computing, CIMM is much more efficient than KACZ. This is due to the fact that KACZ is inherently sequential by its definition, but CIMM can utilize GPUs efficiently due to its inherent parallelism. By using CIMM with GPUs we get 2 levels of parallelism: inside each block of equations, and global parallelism between the blocks. In addition, the process can also be accelerated by CG as in CARP-CG.

5 Conclusions

The purpose of this paper is to demonstrate that there is a simple solution to the problem of subdomain boundaries and cross-points in domain decomposition. This solution is problem independent, and has been shown in previous works to be viable for several well known difficult problems, namely convection-diffusion-reaction problems with very large convection terms, the Helmholtz equation at high frequencies, the elastic wave equation, including viscoelasticity, and eigenvalue computations in difficult cases.

The block-parallel algorithm that was used for these problems by the authors and other researchers was CARP-CG [20], which is a parallel CG-acceleration of the Kaczmarz algorithm [26]. The ability of CARP-CG to solve Domain Decomposition problems seamlessly, i.e., without any error at the boundaries or cross-points, was explained theoretically and demonstrated by several examples.

Although CARP-CG is based on the Kaczmarz algorithm, this paper explains that various other algorithms can replace Kaczmarz in CARP-CG because, in some superspace of the problem space, these algorithms are just Kaczmarz, as shown in [17]. This is a very promising topic for future research; in particular, the most interesting algorithm to replace Kaczmarz in CARP-CG is the Cimmino algorithm [11], because it is amenable to GPU computation inside each subdomain.

Acknowledgments. The authors wish to thank the anonymous reviewers for their many helpful comments.

References

[1] I. M. Babuška and S. A. Sauter. Is the pollution effect of the FEM avoidable for the Helmholtz equation considering high wave numbers? *SIAM Review*, 42:451–484, 2000.

[2] A. Bayliss, C. I. Goldstein, and E. Turkel. On accuracy conditions for the numerical computation of waves. *J. of Computational Physics*, 59:396–404, 1985.

[3] A. Bayliss, M. Gunzburger, and E. Turkel. Boundary conditions for the numerical solution of elliptic equations in exterior regions. *SIAM J. of Applied Mathematics*, 42:430–451, 1982.

[4] J.-P. Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *J. of Computational Physics*, 114:185–200, 1994.

[5] J.-P. Berenger. Three-dimensional perfectly matched layer for the absorption of electromagnetic waves. *J. of Computational Physics*, 127:363–379, 1996.

[6] Å. Björck and T. Elfving. Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. *BIT*, 19:145–163, 1979.

[7] Y. Boubendir, A. Bendali, and M. B. Fares. Coupling of a non-overlapping domain decomposition method for a nodal finite element method with a boundary element method. *International Journal for Numerical Methods in Engineering*, 73(11):1624–1650, 2008.

[8] Y. Censor, T. Elfving, and G. T. Herman. Averaging strings of sequential iterations for convex feasibility problems. In D. Butnariu, Y. Censor, and S. Reich, editors, *Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications*, volume 8 of *Studies in Computational Mathematics*, pages 101–113. Elsevier, Amsterdam, 2001.

[9] Y. Censor, D. Gordon, and R. Gordon. BICAV: A block-iterative parallel algorithm for sparse systems with pixel-dependent weighting. *IEEE Trans. on Medical Imaging*, 20(10):1050–1060, Oct. 2001.

[10] Y. Censor, D. Gordon, and R. Gordon. Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems. *Parallel Computing*, 27(6):777–808, May 2001.

[11] G. Cimmino. Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari. *La Ricerca Scientifica XVI, Series II, Anno IX*, 1:326–333, 1938.

[12] P. P. B. Eggermont, G. T. Herman, and A. Lent. Iterative algorithms for large partitioned linear systems, with applications to image reconstruction. *Linear Algebra & its Applications*, 40:37–67, 1981.

[13] J. Elble, N. Sahinidis, and P. Vouzis. GPU computing with Kaczmarz's and other iterative algorithms. *Parallel Computing*, 36:215–231, June 2010.

[14] B. Engquist and L. Ying. Sweeping preconditioner for the Helmholtz equation: moving perfectly matched layers. *Multiscale Modeling & Simulation*, 9(2):686–710, 2011.

[15] M. Galgon, L. Krämer, J. Thies, A. Basermann, and B. Lang. On the parallel iterative solution of linear systems arising in the FEAST algorithm for computing inner eigenvalues. *Parallel Computing*, 49:153–163, 2015.

[16] M. J. Gander and F. Kwok. Optimized Schwarz methods at cross points. *SIAM J. on Scientific Computing*, 34(4):A1849–A1879, 2012.

[17] D. Gordon. The Cimmino-Kaczmarz equivalence and related results. *Applied Analysis & Optimization*, 2(2):253–270, 2018. http://www.ybook.co.jp/online2/opaao/vol2/p253.html.

[18] D. Gordon and R. Gordon. Component-averaged row projections: A robust, block-parallel scheme for sparse linear systems. *SIAM J. on Scientific Computing*, 27:1092–1117, 2005.

[19] D. Gordon and R. Gordon. Solution methods for linear systems with large off-diagonal elements and discontinuous coefficients. *Computer Modeling in Engineering & Sciences*, 53(1):23–45, Nov. 2009.

[20] D. Gordon and R. Gordon. CARP-CG: a robust and efficient parallel solver for linear systems, applied to strongly convection-dominated PDEs. *Parallel Computing*, 36(9):495–515, Sept. 2010.

[21] D. Gordon and R. Gordon. Row scaling as a preconditioner for some nonsymmetric linear systems with discontinuous coefficients. *J. of Computational & Applied Mathematics*, 234(12):3480–3495, Oct. 2010.

[22] D. Gordon and R. Gordon. Parallel solution of high frequency Helmholtz equations using high order finite difference schemes. *Applied Mathematics & Computation*, 218(21):10737–10754, July 2012.

[23] D. Gordon, R. Gordon, and E. Turkel. Compact high order schemes with gradient-direction derivatives for absorbing boundary conditions. *J. of Computational Physics*, 297(9):295–315, Sept. 2015.

[24] R. Gordon, R. Bender, and G. T. Herman. Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography. *J. of Theoretical Biology*, 29(3):471–481, Dec. 1970.

[25] G. T. Herman. *Fundamentals of Computerized Tomography: Image Reconstruction From Projections*. Springer, 2nd edition, 2009.

[26] S. Kaczmarz. Angenäherte Auflösung von Systemen linearer Gleichungen. *Bulletin de l'Académie Polonaise des Sciences et Lettres*, A35:355–357, 1937.

[27] Y. Li, B. Han, L. Métivier, and R. Brossier. Optimal fourth-order staggered-grid finite-difference scheme for 3D frequency-domain viscoelastic wave modeling. *J. of Computational Physics*, 321:1055–1078, 2016.

[28] Y. Li, L. Métivier, R. Brossier, B. Han, and J. Virieux. 2D and 3D frequency-domain elastic wave modeling in complex media with a parallel iterative solver. *Geophysics*, 80(3):T101–T118, 2015.

[29] S. Loisel. Condition number estimates for the nonoverlapping optimized Schwarz method and the 2-Lagrange multiplier method for general domains and cross points. *SIAM J. on Numerical Analysis*, 51(6):3062–3083, 2013.

[30] A. Sommerfeld. *Partial Differential Equations in Physics*. Academic Press, New York, 1964.

[31] C. C. Stolk. A rapidly converging domain decomposition method for the Helmholtz equation. *J. of Computational Physics*, 241:240–252, 2013.

[32] C. C. Stolk. An improved sweeping domain decomposition preconditioner for the Helmholtz equation. *Advances in Computational Mathematics*, 43:45–76, 2017.

[33] J. Thies, M. Galgon, F. Shahzad, A. Alvermann, M. Kreutzer, A. Pieper, M. Röhrig-Zöllner, A. Basermann, H. Fehske, G. Hager, B. Lang, and G. Wellein. Towards an exascale enabled sparse solver repository. In H. J. Bungartz, P. Neumann, and W. Nagel, editors, *Software for Exascale Computing – SPPEXA 2013-2015*, volume 113 of *Lecture Notes in Computational Science and Engineering*, pages 73–89. Springer, Cham, Switzerland, 2016. DOI: 10.1007/978-3-319-40528-5_13.

[34] E. Turkel, D. Gordon, R. Gordon, and S. Tsynkov. Compact 2D and 3D sixth order schemes for the Helmholtz equation with variable wave number. *J. of Computational Physics*, 232(1):272–287, Jan. 2013.

[35] T. van Leeuwen, D. Gordon, R. Gordon, and F. Herrmann. Preconditioning the Helmholtz equation via row projections. In *Proc. 74th EAGE Conference, Copenhagen, Denmark*, pages 60–65. EAGE, June 2012.

[36] T. van Leeuwen and F. J. Herrmann. 3D frequency-domain seismic inversion with controlled sloppiness. *SIAM J. on Scientific Computing*, 36(5):S192–S217, 2014.