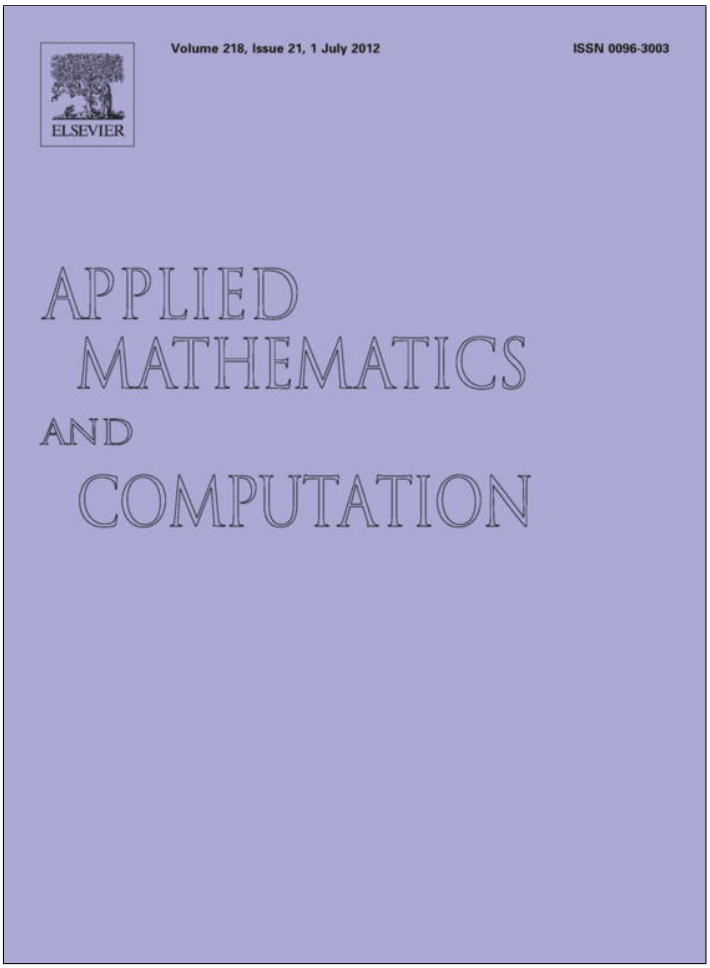

Applied Mathematics and Computation 218 (2012) 10737–10754

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Parallel solution of high frequency Helmholtz equations using high order finite difference schemes

Dan Gordon [a,*], Rachel Gordon [b]

[a] Dept. of Computer Science, University of Haifa, Haifa 31905, Israel
[b] Dept. of Aerospace Engineering, The Technion-Israel Institute of Technology, Haifa 32000, Israel

ARTICLE INFO

Keywords:
CARP-CG
Helmholtz equation
Heterogeneous domain
High frequency
High order scheme
L-shaped domain
Marmousi
Parallel processing

ABSTRACT

High frequency Helmholtz equations pose a challenging computational task, especially at high frequencies. This work examines the parallel solution of such problems, using 2nd, 4th and 6th order finite difference schemes. The examples include problems with known analytic solutions, enabling error evaluation of the different schemes on various grids, including problems on L-shaped domains. The resulting linear systems are solved with the block-parallel CARP-CG algorithm which succeeded in lowering the relative residual, indicating that it is a robust and reliable parallel solver of the resulting linear systems. However, lowering the error of the solution to reasonable levels with practical mesh sizes is obtained only with the higher order schemes. These results corroborate the known limitations of the 2nd order scheme at modeling the Helmholtz equation at high frequencies, and they indicate that CARP-CG can also be used effectively with high order finite difference schemes. Furthermore, the parallel scalability of CARP-CG improves when the wave number is increased (on a fixed grid), or when, for a fixed wave number, the grid size is decreased.

1. Introduction

Numerical solution of the Helmholtz equation at high frequencies (large wave numbers) is a challenging computational task. Compounding the problem is the well-known fact that 2nd order finite difference schemes do not model the problem very well and so require many grid points per wavelength. As a result, even if the linear system is solved to within a very small relative residual, the computed solution may be quite far from the true solution of the partial differential equation (PDE).

An additional problem with high wave numbers is the so-called "pollution" effect, which was first published in [1] (but it was not called by that name); see also [2]. It is generally considered that at least 8–12 grid points per wavelength are required to achieve a satisfactory solution. However, the pollution effect causes the relation between the wave number k and the number of grid points per wavelength (denoted N_g) to be non-linear: N_g should be taken as proportional to $k^{(p+1)/p}$, where p is the order of the accuracy of the scheme; see [3]. Hence, high order schemes have a clear advantage with large wave numbers.

The literature on the Helmholtz equation is very extensive, with over 8000 titles containing the word "Helmholtz". Many solution methods have been proposed and studied over the years, and a survey on this topic is beyond the scope of this paper. One of the most prominent approaches is the "shifted Laplacian" method, introduced by Bayliss et al. [4]. The shift

* Corresponding author.
E-mail addresses: gordon@cs.haifa.ac.il (D. Gordon), rgordon@tx.technion.ac.il (R. Gordon).

0096-3003/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.amc.2012.04.052

in the Laplacian is used as a preconditioner. Erlangga et al. [5] introduced the use of a complex shift. For a summary and some new results, see [6], and for recent results with this approach using higher order schemes; see [3]. For some other approaches, see [7–11].

This work presents numerical experiments with the block-parallel CARP-CG algorithm [12], applied to the Helmholtz equation with a large wave number, using 2nd, 4th and 6th order finite difference discretization schemes. CARP-CG is simple to implement on structured and unstructured grids. In [13], it was shown to be a robust and highly scalable solver of high frequency Helmholtz equations in homogeneous and heterogeneous media in 2D and 3D domains, using 2nd order finite difference schemes. CARP-CG is generally useful for linear systems with large off-diagonal elements [14,12], and also with discontinuous coefficients [15].

The high order schemes that are used in this work are all *compact*, meaning that they yield a 9-point and 27-point stencil matrix in 2D and 3D, respectively. Such schemes are usually advantageous for absorbing boundary conditions, since they require only one extra layer at the exterior boundaries. They are also simpler to implement than other high order schemes, and they are advantageous for CARP-CG since they do not require more inter-processor communications than second order schemes.

In this work we used the 2D 4th order scheme for variable k from Harari and Turkel [16], and the 2D 6th order scheme for constant k from Singer and Turkel [17]. The 3D, 6th order scheme for constant k is taken from Sutmann [18, Eq. (17)].

Our experiments in 2D and 3D included problems with known analytic solutions, enabling evaluation of the error, including problems on L-shaped domains. CARP-CG succeeded with all schemes at lowering the residual, indicating that it is a robust and reliable parallel solver of the resulting linear systems. However, lowering the *error* of the solutions to reasonable levels with practical mesh sizes was possible only with the higher order schemes. These results corroborate the known limitations of the low order scheme at modeling the Helmholtz equation (especially at high frequencies), and they indicate that CARP-CG is also effective with high order finite difference schemes. Furthermore, the parallel scalability of CARP-CG improves under any one of the following two conditions:

1. The grid is fixed and the wave number is increased.
2. The wave number is fixed and the grid size is decreased.

Some of the 2D results were presented at the 10th WAVES conference in 2011 [19]. The rest of this paper is organized as follows. Section 2 describes CARP-CG and explains its effectiveness. Section 3 describes the Helmholtz equation and its discretization, and Section 4 explains the setup of the numerical experiments. Sections 5 and 6 present the results for the 2D and 3D problems, and Section 7 discusses the inherent difficulty with the 2nd order scheme. Section 8 presents some conclusions.

2. Description of CARP-CG and its effectiveness

2.1. Description of CARP-CG

We present a brief and informal explanation of CARP-CG; for full details, see [12]. Consider a system of m linear equations in n variables, $Ax = b$. The Kaczmarz algorithm (KACZ) [20] is fundamental to CARP-CG. Starting from some arbitrary initial guess, KACZ successively projects the current iterate onto a hyperplane defined by one of the equations in cyclic order. Each cycle of projections is called a KACZ sweep. The projections can be modified with a relaxation parameter $0 < \omega < 2$. If equation i has a fixed relaxation parameter ω_i, then the projections are said to be done with cyclic relaxation.

In a landmark paper, Björck and Elfving [21] showed that if a forward sweep of KACZ 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-gradient (CG) algorithm. The iteration matrix is not actually calculated; instead, whenever it has to be multiplied by a vector, the calculation is done by performing a suitable KACZ double sweep. The resulting algorithm, called CGMN, is one cornerstone of CARP-CG.

A second cornerstone for CARP-CG is CARP [22], which is a block-parallel version of KACZ. Note that this is different from the standard block-sequential version of KACZ, which requires the equations in a block to be independent (i.e., there are no shared variables in a block). CARP divides the equations into blocks, which may overlap, and equations in a block need not be independent. In a parallel setting, every processor is in charge of a block of equations. Every processor has a copy, or "clone", of every variable that it shares with another block. The following two steps are now repeated until convergence:

1. Every processor performs a KACZ sweep on the equations of its assigned block, updating the block's variables. For shared variables, each processor updates its copy of the variable.
2. The processors exchange information about the new values of the clones. Every shared variable is now updated to be the average of all its clones in the different blocks, and the new value of every shared variable is distributed among the processors which share it.

If the blocks are chosen according to spatial partitioning, then the exchange of data occurs only at the boundaries between subdomains. CARP can be viewed as a type of domain decomposition (DD) technique which differs from other DD methods

because values from neighboring domains (the clones) are also modified by the internal processing. Furthermore, partitioning a domain into subdomains can be based entirely on efficiency considerations such as load-balancing and minimizing inter-processor communications. In this context, the compact high order difference schemes are advantageous since they do not require more elements from neighboring subdomains than the 2nd order scheme.

For a detailed parallel implementation of CARP, see [22]. An important point about CARP is that the averaging operations between shared variables are equivalent to certain KACZ row projections (with relaxation $\omega = 1$) in some superspace. Hence, CARP is equivalent to KACZ with cyclic relaxation in the superspace. This property provides a convenient convergence proof for CARP, and it enables the CG acceleration of CARP.

The CG acceleration is obtained in [12] as follows. Firstly, the construction of CGMN is extended to KACZ with cyclic relaxation parameters. It then follows that the superspace formulation of CARP can be accelerated by CG. This means that in the regular space, CARP can be accelerated by CG in the same manner as CGMN, i.e., by running CARP in a double sweep. On one processor, CARP-CG and CGMN are identical.

2.2. The Effectiveness of CARP-CG

The effectiveness of CARP-CG follows from two inherent features of KACZ, which will be explained below. Let $Ax = b$ be the given linear system. We denote the L_2-norm of a vector x by $||x||$. Consider the projection of the current iterate x onto the hyperplane defined by the ith equation, yielding a new iterate x' defined by

$$x' = x + \omega_i \frac{b_i - \langle a_{i*}, x \rangle}{||a_{i*}||^2} a_{i*}, \tag{1}$$

where $0 < \omega_i < 2$ is the relaxation parameter associated with the ith equation and a_{i*} is the ith row of A. Eq. (1) shows that KACZ inherently normalizes the equations, i.e., the ith equation is divided by $||a_{i*}||$. Therefore, we can assume that this normalization has already been done as a preliminary step; it is also more efficient to do so in practice. Note that KACZ is a geometric algorithm in the following sense: the iterates depend only on the hyperplanes defined by the equations and not on any particular algebraic representation of these hyperplanes. For this reason, we call the normalization GRS (geometric row scaling).

This inherent use of GRS by KACZ is the reason that CARP-CG can handle discontinuous coefficients effectively. GRS is a diagonal scaling of A, and the positive effect of such scalings, when dealing with discontinuous coefficients has been known for a long time; see, for example [23]. More recently, an extensive study in [24] has shown that normalization significantly improves the convergence properties of Bi-CGSTAB [25] and restarted GMRES [26] (both of them with and without the ILU (0) preconditioner) on nonsymmetric linear systems with discontinuous coefficients derived from convection–diffusion PDEs. However, for these algorithm/preconditioner combinations, these results held only for small-to moderate-sized convection terms; for large convection terms, [15] shows that CARP-CG is preferable.

An examination of the normalization in [24,15] shows that it "pushes" heavy concentrations of eigenvalues around the origin away from the origin. It is known that such concentrations are detrimental to convergence. These results explain why CARP-CG can handle the problem of discontinuous coefficients. Note that the normalization has a certain "equalization" effect on the coefficients of different equations (but not on different-sized coefficients in the same equation).

A second point concerning KACZ is the fact that it is actually SOR (successive over-relaxation) on the normal equations system $AA^Ty = b (x = A^Ty)$; see [21]. Using AA^T is generally considered detrimental because its condition number is the square of the condition number of A. However, the following fact, which apparently appeared first in [15, Theorem 3.1], seems to have been largely overlooked: assuming that GRS has been applied to A (as is inherent in KACZ), then the diagonal elements of AA^T are equal to 1, and all the off-diagonal elements are <1, provided no two rows of A are colinear. In other words, the two simple operations of applying GRS to A and then using AA^T, form a very simple means of dealing with systems with large off-diagonal elements, including cases of discontinuous coefficients.

3. The Helmholtz equation and discretization schemes

3.1. The Helmholtz equation

Let D be a given domain. We use the following notations:

- c is the speed of the waves (in m/s); c may vary in D.
- f is the frequency of the waves in Hertz.
- $\lambda = c/f$ is the wave length (in meters).
- The (dimensional) wave number is defined as $k = 2\pi f/c$, from which we have $\lambda = 2\pi/k$.

The Helmholtz equation in D is $-\left(\Delta + k^2\right)u = g$, where k is the wave number, and g is a prescribed right-hand-side. The following additional notations will be used.

- If D is a square measuring $L \times L$, then one can define the non-dimensional wave number as $\bar{k} = kL$. Generally, the term "large wave number" refers to large values of $\bar{k}$. We shall use k instead of $\bar{k}$ when the meaning is clear from the context.
- In two dimensions, the domain D is discretized with a uniform grid of $N \times N$ points ($N \times N \times N$ in 3D). N includes the boundary points, but for absorbing boundary conditions, one or more extra layers of points are added.
- $h = L/(N-1)$ (in meters) denotes the mesh spacing.
- From the above, we obtain the number of grid points per wavelength as $N_g = c/(fh) = 2\pi/(kh)$.

3.2. Discretization schemes

In our experiments, we discretized the equations by using 2nd, 4th and 6th order finite difference schemes. This yields a system of equations $Ax = b$, which, depending on the boundary conditions, may contain complex elements.

In two dimensions, the standard 2nd order central difference scheme (for constant and variable k) is given by the following constants, defined on a 5-point stencil:

$$\begin{aligned} &\text{At the center point}: A_0^{2D2} = -4 + (kh)^2, \\ &\text{At the 4 side points}: A_s^{2D2} = 1. \end{aligned} \tag{2}$$

In three dimensions, we have a 7-point stencil:

$$\begin{aligned} &\text{At the center point}: A_0^{3D2} = -6 + (kh)^2, \\ &\text{At the 6 side points}: A_s^{3D2} = 1. \end{aligned} \tag{3}$$

The 2D 4th order scheme for variable k [16] uses a 3×3 square of values, and in each case, the value of k is taken at the corresponding point. This is clarified in the following scheme by assigning an index to k corresponding to the point at which it should be taken.

$$\begin{aligned} &\text{At the center point}: A_0^{2D4} = -\frac{10}{3} + \left(\frac{2}{3} + \frac{\gamma}{36}\right)(k_0 h)^2, \\ &\text{At the 4 side points}: A_s^{2D4} = \frac{2}{3} + \left(\frac{1}{12} - \frac{\gamma}{72}\right)(k_s h)^2, \\ &\text{At the 4 corner points}: A_c^{2D4} = \frac{1}{6} + \frac{\gamma}{144}(k_c h)^2, \end{aligned} \tag{4}$$

where γ is an arbitrary constant (taken as zero in our experiments).

The 2D 6th order scheme for constant k [17] is obtained from the 4th order scheme by taking $\gamma = \frac{14}{5}$ and adding terms of the order of $(kh)^4$:

$$\begin{aligned} &\text{At the center point}: A_0^{2D6} = -\frac{10}{3} + \frac{67}{90}(kh)^2 + \frac{\delta - 3}{180}(kh)^4, \\ &\text{At the 4 side points}: A_s^{2D6} = \frac{2}{3} + \frac{2}{45}(kh)^2 + \frac{3 - 2\delta}{720}(kh)^4, \\ &\text{At the 4 corner points}: A_c^{2D6} = \frac{1}{6} + \frac{7}{360}(kh)^2 + \frac{\delta}{720}(kh)^4, \end{aligned} \tag{5}$$

where δ is an arbitrary constant (taken as zero in our experiments).

The 3D, 6th order scheme for constant k [18, Eq. 17] uses a 27-point stencil defined at all points of a $3 \times 3 \times 3$ cube, as follows.

$$\begin{aligned} &\text{At the center}: A_0^{3D6} = -\frac{64}{15h^2}\left(1 - \frac{(kh)^2}{4} + \frac{5(kh)^4}{256} - \frac{(kh)^6}{1536}\right), \\ &\text{On the 6 sides}: A_s^{3D6} = \frac{7}{15h^2}\left(1 - \frac{(kh)^2}{21}\right), \\ &\text{On the 12 edges}: A_e^{3D6} = \frac{1}{10h^2}\left(1 + \frac{(kh)^2}{18}\right), \\ &\text{At the 8 corners}: A_c^{3D6} = \frac{1}{30h^2}. \end{aligned} \tag{6}$$

3.3. Boundary conditions

Boundary conditions vary with the problems. In some cases, an impact on one side is modeled by a discontinuity on that side. In order to model unbounded domains, it is necessary to prevent reflecting waves from the boundaries. In our examples,

this was done by using the first-order absorbing boundary condition, derived from the Sommerfeld radiation condition [27, Section 28]

$$\partial u/\partial \vec{n} - iku = 0, \tag{7}$$

where $\vec{n}$ is the unit vector pointing outwards from the domain, and $i = \sqrt{-1}$.

In order to incorporate this boundary condition, we add an extra layer of grid points to each side for which (7) holds, and for each such extra grid point, we add the equation obtained by discretizing (7). For example, suppose we are dealing with the unit square (or unit cube), and we choose this boundary condition on the side $x = 1$. Then, following [3, Section 3], we obtain the following 6th order accurate equation for each new grid point:

$$\frac{1}{2h}\left(u_{N+1} + 2ikh\left(1 - \frac{k^2h^2}{6} + \frac{k^4h^4}{120}\right)u_N - u_{N-1}\right) = 0, \tag{8}$$

where h is the mesh spacing and $u_{N-1},\ u_N,\ u_{N+1}$ are the values of u interior to the boundary, on the boundary, and at the extra grid point, respectively. Thus, for every new grid point, we get one new equation.

Note that when the absorbing boundary condition is chosen on one side, then each (original) boundary point on that side will have a regular equation centered on it, and that equation will also involve one or more of the extra grid points. As a result, when using the 9-point stencil in 2D, each new grid point will (in general) participate in three regular equations. In 3D, with a 27-point stencil, each new grid point will participate in nine regular equations.

4. Setup of the numerical experiments

4.1. Implementation details

Tests were run on a Supermicro cluster consisting of 12 nodes connected by an Infiniband network. Each node consists of two Intel Xeon E5520 quad CPUs running at 2.27 GHz, so the cluster can provide a total of 96 cores. The two CPUs share 8 GB of memory and each has its own 8 MB cache. The cluster runs under Debian Linux, and message passing used the MPICH2 public domain MPI package.

Our implementation of the data structures used by CARP-CG is similar to that of the AZTEC numerical software package [28]. In a preprocessing stage, the following information is saved in each processor: a list of neighboring processors with which it communicates, a list of local nodes that are coupled with external nodes, and a local representation of the submatrix and subvectors used by the processor. The local submatrix in each processor is stored in the sparse matrix format called DMSR (distributed modified sparse row) [28], which is a generalization of the MSR format [29, Section 3.4]. Additionally, every processor stores the "clones" of the relevant external grid points.

CARP-CG is written in C and compiled with the GNU C compiler, with optimization parameter-O. All matrix–vector and scalar products are coded inline; this is one contributing factor to the efficiency of our code. CARP-CG was implemented with the MPI Cartesian topology routines for inter-processor communications. The initial estimate was taken in all cases as $x^0 = 0$, and the relaxation parameter was taken as $\omega = 1.5$ (except for Problem 2L).

4.2. Solving complex-valued equations

There are several known methods for solving complex equations; see [30]. According to this paper, and as confirmed by our experiments, the following approach is the most efficient. A complex system of equations $Ax = b$ can be broken down as follows: let $A = B + iC,\ b = c + id$, and $x = y + iz$, where B, C, c, d, z, y are all real-valued. This yields the system $(B + iC)(y + iz) = c + id$, which is equivalent to the two real-valued systems $By - Cz = c,\ Cy + Bz = d$. These two $n \times 2n$ systems can be solved by "merging" them into one real-valued $2n \times 2n$ system by alternately taking one equation from the first system and one from the second system. This gives us the system

$$\begin{pmatrix} b_{11}, & -c_{11}, & \dots, & b_{1n}, & -c_{1n} \\ c_{11}, & b_{11}, & \dots, & c_{1n}, & b_{1n} \\ & & \vdots & & \\ b_{n1}, & -c_{n1}, & \dots, & b_{nn}, & -c_{nn} \\ c_{n1}, & b_{n1}, & \dots, & c_{nn}, & b_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ z_1 \\ \vdots \\ y_n \\ z_n \end{pmatrix} = \begin{pmatrix} c_1 \\ d_1 \\ \vdots \\ c_n \\ d_n \end{pmatrix}. \tag{9}$$

Note that when solving the system (9) with CARP-CG, our first step is to normalize this system (and not the original $Ax = b$). In our experiments, we found that solving (9) with CARP-CG was about twice as fast as solving the original complex system.

5. 2D problems

We consider four problems in 2D, including one problem on an L-shaped domain. The first three have analytic solutions, so we can compare the results with the true solution. The fourth problem is the well known Marmousi model [31]. For Problems 1 and 2, the wave number was taken as $k = 300$, and the number of grid points per wavelength was taken as $N_g = 9, 12, 15, 18$. Table 1 shows the grid sizes for the various values of N_g. In Problem 1 the variables are complex, and in Problem 2 they are real.

5.1. Problem 1

This example is taken from [3] (but with x and y interchanged for convenience). The equation $\Delta u + k^2 u = 0$ is defined on the square $[-0.5, 0.5] \times [0, 1]$. Dirichlet boundary conditions were taken on three sides: $u = 0$ for $x = -0.5$ and $x = 0.5$, and $u = \cos(\pi x)$ for $y = 0$. On the side $y = 1$, a first order absorbing boundary condition (Sommerfeld radiation condition) was taken as $u_y - i\beta u = 0$, where $\beta^2 = k^2 - \pi^2$. The analytic solution to this problem is $u(x, y) = \cos(\pi x) e^{i\beta y}$. $k = 300$ in all the test runs of this problem.

Fig. 1 shows the relative residual results for the three schemes, for $N_g = 18$, on one processor. Here, all three convergence plots are quite similar, and those of the higher order schemes are almost identical.

Fig. 2 show the L_2-error plots of the three schemes for $N_g = 18$, on one processor. The 2nd order scheme stagnates at a value of 0.78 with this mesh size, probably due to the pollution effect. This figure indicates the importance of the high order schemes, especially the 6th order one for constant k.

Fig. 3 shows the error plots of the 6th order scheme for $9 \leqslant N_g \leqslant 18$, on one processor. Since higher values of N_g take more computation time, N_g should be chosen according to the desired accuracy.

The parallel performance of CARP-CG with the 6th order scheme on Problem 1 is shown in Fig. 4. The number of processors varies from one to 16, with $N_g = 15$. The different plots are due to the fact that mathematically, we are actually solving different systems of equations when we vary the number of processors. This is due to the averaging operations of CARP-CG, which, in the superspace, are equivalent to additional equations in the system.

Table 2 shows the number of iterations and runtimes required to reach a relative residual of 10^{-7} with the 6th order scheme for $N_g = 15$, on 1 to 16 processors. There appears to be a certain anomaly here with the number of iterations reaching

Table 1
Problems 1 and 2: grid sizes for different values of N_g, for $k = 300$.

N_g	9	12	15	18
Grid	431×431	575×575	717×717	861×861

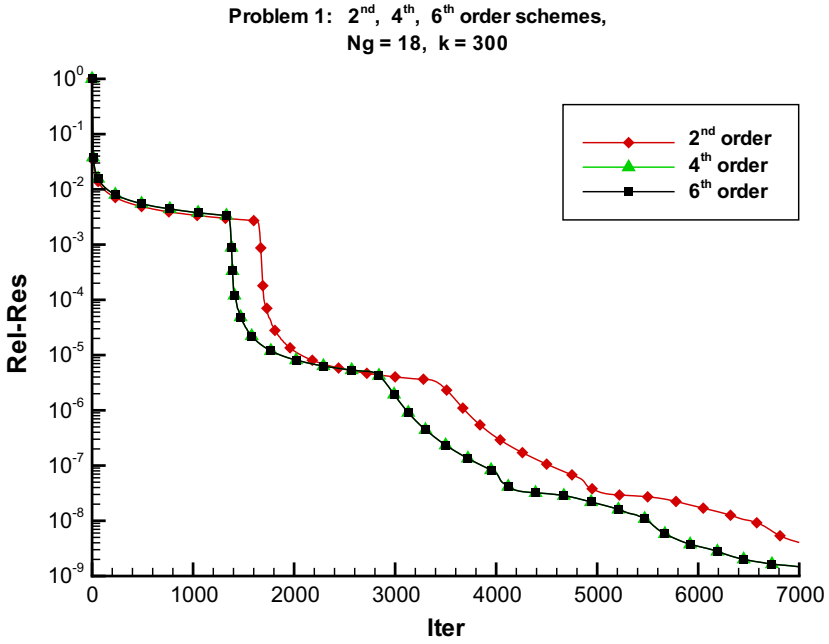

Fig. 1. Problem 1: relative residual for $N_g = 18$, one processor, with the 2nd, 4th and 6th order schemes.

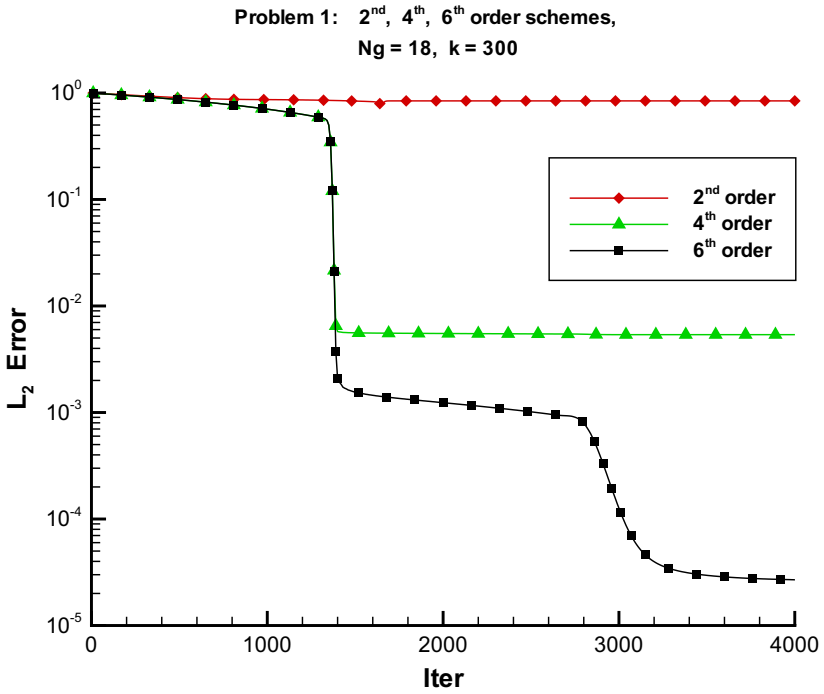

Fig. 2. Problem 1: L_2-error for $N_g = 18$, one processor, with the 2nd, 4th and 6th order schemes.

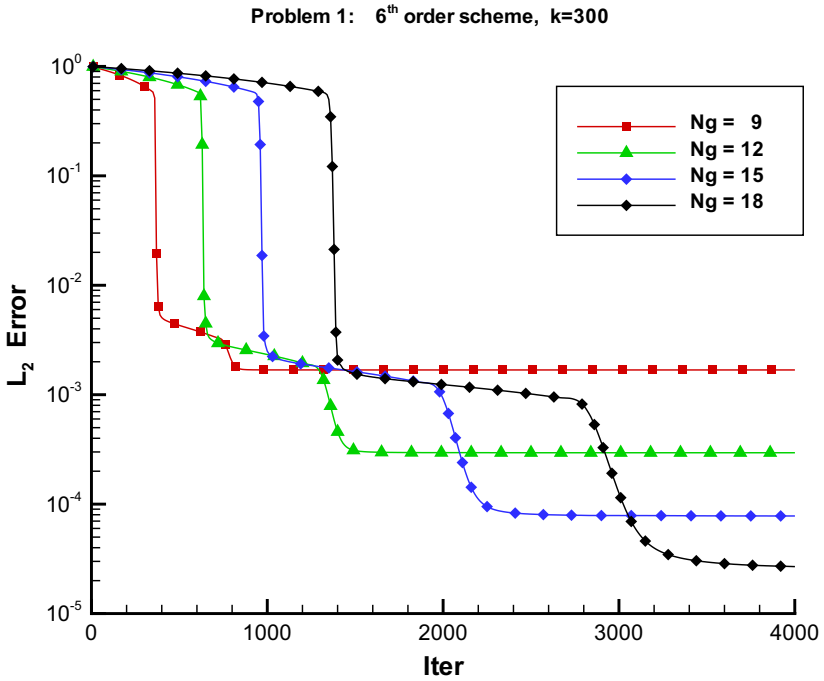

Fig. 3. Problem 1: L_2-error with 6th order scheme, for $9 \leqslant N_g \leqslant 18$, on one processor.

a maximum with eight processors. This is also apparent in Fig. 4. We conjecture that the cause of this is that with 8 processors, the eigenvalues of the system matrix (in the superspace) turn out to badly distributed as compared to 4 or 12 processors.

5.2. Problem 2

The PDE $\Delta u + k^2 u = 0$ is defined on the unit square $[0,1] \times [0,1]$. The analytic solution chosen for this problem was taken as $u(x,y) = \sin(\pi x)\cos(\beta y)$, where $\beta^2 = k^2 - \pi^2$. Dirichlet boundary conditions were determined by the values of u on the

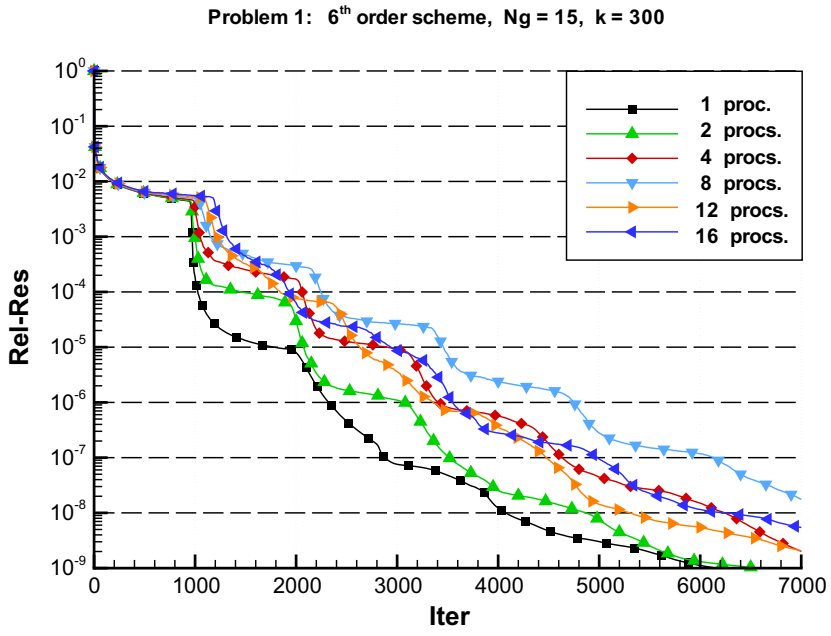

Fig. 4. Problem 1: relative residual for 1–16 processors with 6th order scheme, for $N_g = 15$.

Table 2
Problem 1: number of iterations and runtimes (s) required to reach a relative residual of 10^{-7}, on 1–16 processors with the 6th order scheme, for $k = 300$ and $N_g = 15$, on a grid of 717×717.

# Proc.	1	2	4	8	12	16
No. iter.	2881	3516	4634	6125	4478	4983
Runtime	(329)	(233)	(157)	(96)	(62)	(51)

boundary: $u = 0$ for $x = 0$ and $x = 1$, $u = \sin(\pi x)$ for $y = 0$, and $u = \sin(\pi x)\cos\beta$ for $y = 1$. For this problem, k was also taken as 300 on all test runs.

Fig. 5 shows the relative residual with the three schemes, for $N_g = 18$, on one processor. The two high order schemes achieve almost identical results, which are better than those of the 2nd order scheme.

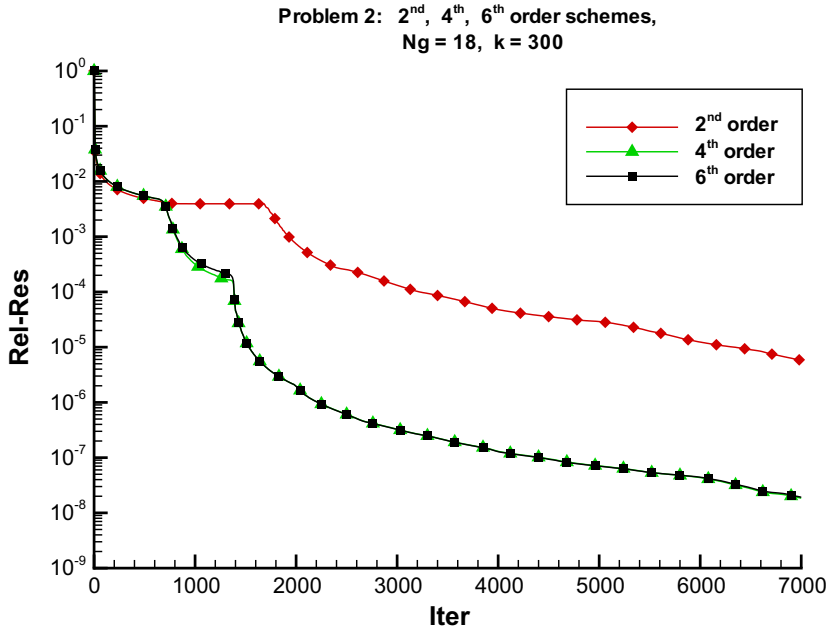

Fig. 5. Problem 2: relative residual for $N_g = 18$, on one processor, with the 2nd, 4th and 6th order schemes.

Fig. 6 shows the relative error, under the same conditions as in Fig. 5. The 2nd order scheme produces reasonable relative residual results, but it is clearly inadequate for any purpose. The 6th order scheme is preferable to the 4th order scheme for constant k because their computation times are identical.

Fig. 7 concentrates on the 6th order scheme and shows the L_2-error for N_g ranging from 9 to 18, on one processor. Since higher values of N_g require finer grids and hence longer time per iteration, N_g should be chosen according to the required accuracy.

The parallel performance of CARP-CG on this problem is shown in Fig. 8. Results are shown for 1–16 processors, for $N_g = 15$.

Table 3 shows the number of iterations and runtimes required to reach a relative residual of 10^{-7} with the 6th order scheme for $N_g = 15$, on 1 to 16 processors.

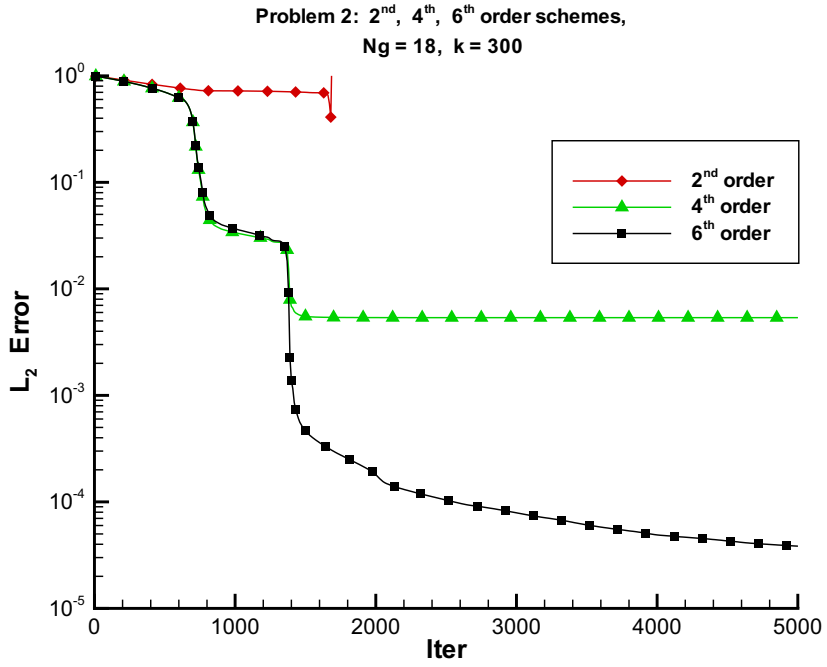

Fig. 6. Problem 2: L_2-error for $N_g = 18$, on one processor, with the 2nd, 4th and 6th order schemes.

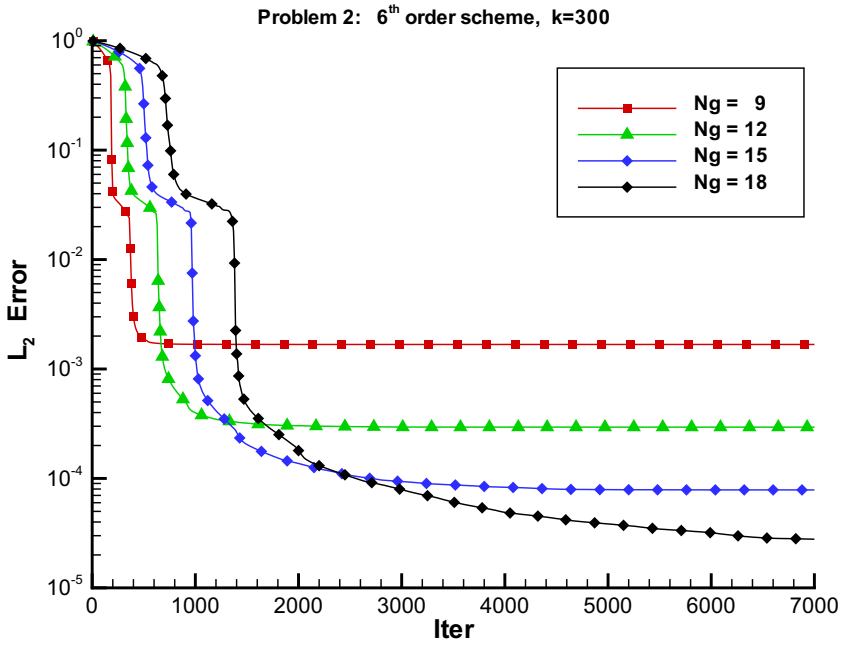

Fig. 7. Problem 2: L_2-error with the 6th order scheme, one processor, for $9 \leqslant N_g \leqslant 18$.

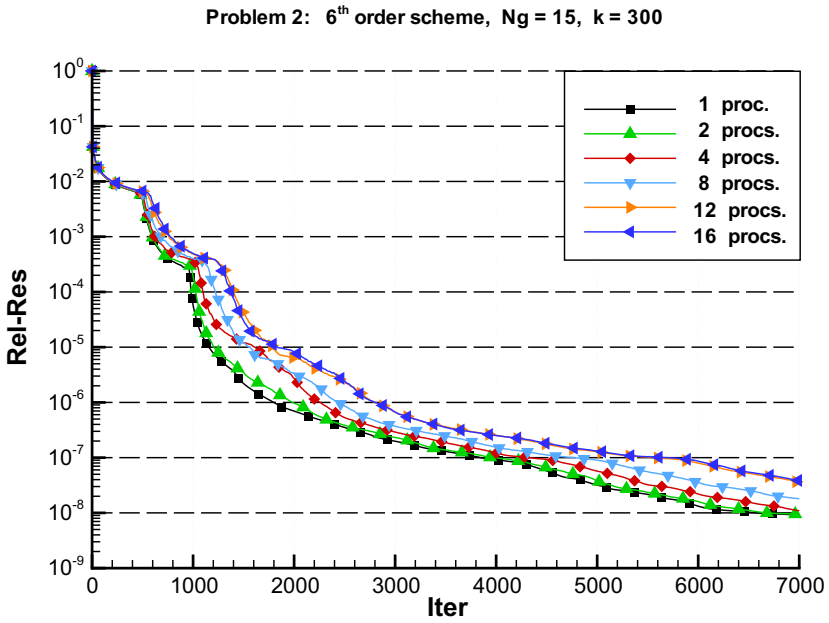

Fig. 8. Problem 2: relative residual for 1–16 processors with 6th order scheme, for $N_g = 15$.

Table 3
Problem 2: number of iterations and runtimes (s) required to reach a relative residual of 10^{-7} on 1–16 processors with the 6th order scheme, for $k = 300$ and $N_g = 15$, on a grid of 717×717.

# Proc.	1	2	4	8	12	16
No. iter.	3847	3981	4328	4774	5561	5691
Runtime	(207)	(117)	(65)	(37)	(30)	(24)

5.3. Problem 2L–an L-shaped domain

This problem is identical to Problem 2, except that one quadrant is removed from the unit square, resulting in an L-shaped domain, as shown in Fig. 9. Such domains are well-known to be problematic, because the inner corner gives rise to very strong gradients in its vicinity. In this section, w was taken as 1.6 since this produced slightly better results.

Fig. 10 shows the L_2-error with the 6th order scheme on one processor, for $k = 50$ and $N_g = 9, 12, 15, 18$. The corresponding grids are $73^2, 97^2, 120^2, 144^2$. Similarly to Problems 1 and 2, larger values of N_g provide better accuracy but smaller values are preferable if they can provide the required accuracy.

Fig. 11 shows the L_2-error with the 4th and 6th order schemes on one processor, for $N_g = 15$ and $k = 25, 50, 100$. The corresponding grids are $61^2, 120^2, 240^2$. The results with the 2nd order scheme are not shown because it achieved an L_2-error of

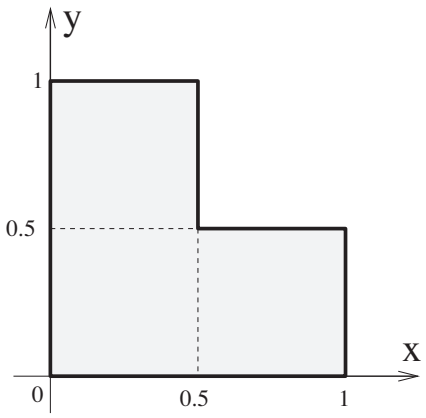

Fig. 9. The L-shaped domain.

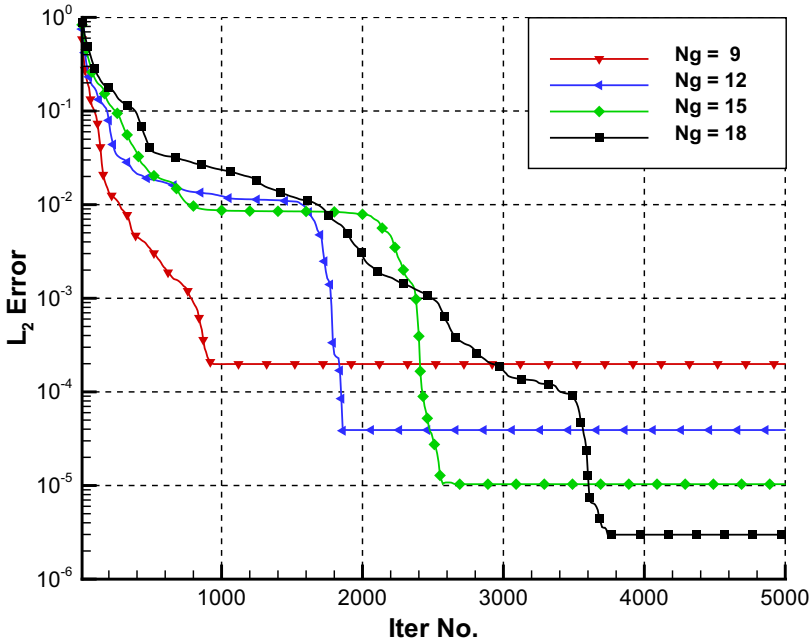

Fig. 10. Problem 2L: L_2-error with the 6th order scheme on one processor, for $k = 50$ and $9 \leqslant N_g \leqslant 18$.

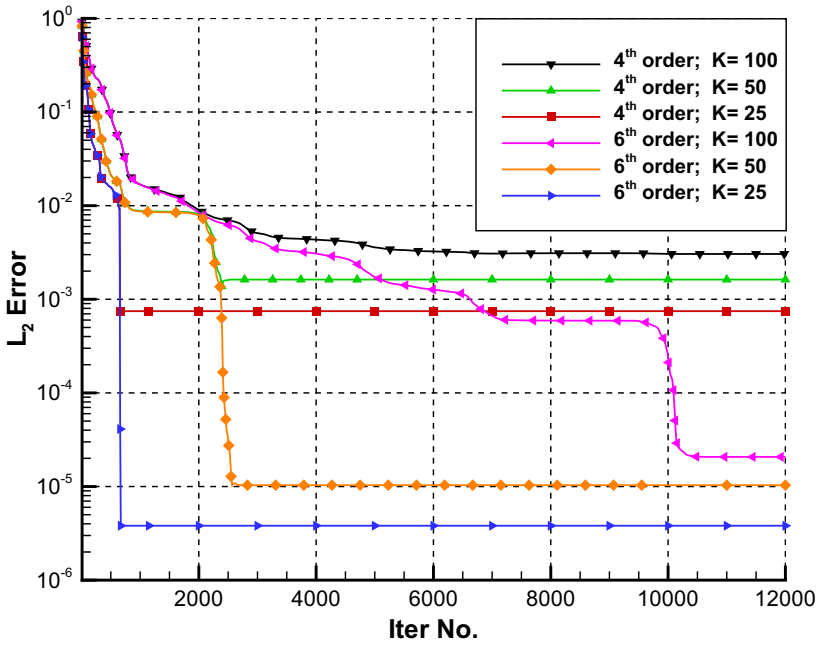

Fig. 11. Problem 2L: L_2-error on one processor with the 4th and 6th order schemes, for $k = 25$, 50, 100 and $N_g = 15$.

only 0.149 with $k = 50$ and 0.308 with $k = 100$, and then the error increased. The 6th order scheme is clearly preferable to the 4th (when k is constant).

Fig. 12 compares the L_2-error results for the original Problem 2 with the results for the L-shaped domain, on one processor, with the 4th and 6th order schemes, for $k = 50$ and $N_g = 15$. The corresponding grid is 120^2. We can see that many more iterations are required on the L-shaped domain before achieving the same error as on the original square.

Fig. 13 is similar to Fig. 12, except that now $k = 100$ and the grid is 240^2. The difference between the required number of iterations for each case is now much larger than for $k = 50$.

Table 4 shows the times (in seconds) per 1000 iterations, on one processor with the 6th order scheme, for the square and the L-shaped domains, for $k = 50$ and $N_g = 9, 12, 15, 18$. The corresponding grids are $73^2, 97^2, 120^2, 144^2$. As can be expected, the time per iteration on the L-shaped domain is approximately 75% of the time on the full square (for equal values of N_g).

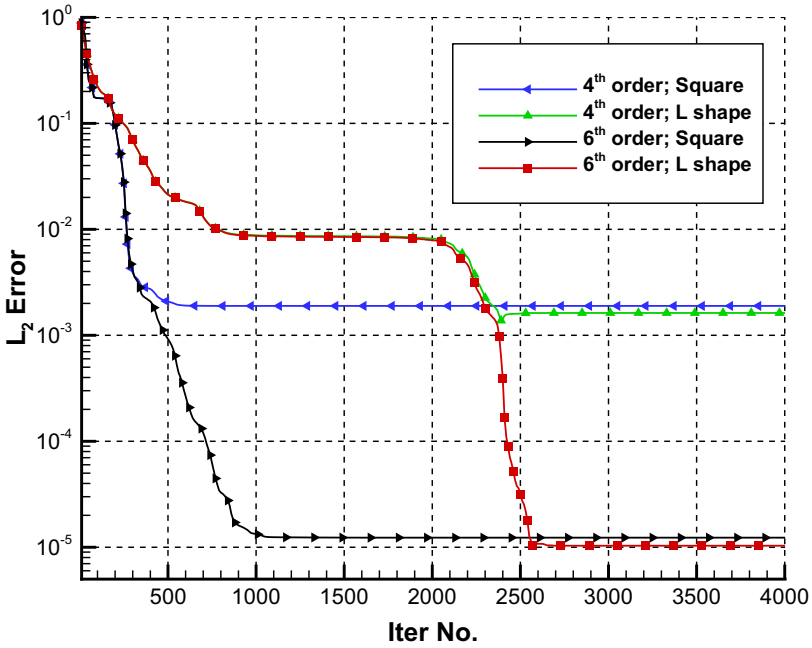

Fig. 12. Problem 2L: L_2-error with the 4th and 6th order schemes for the square and the L-shaped domains, for $k = 50, N_g = 15$ and grid 120×120.

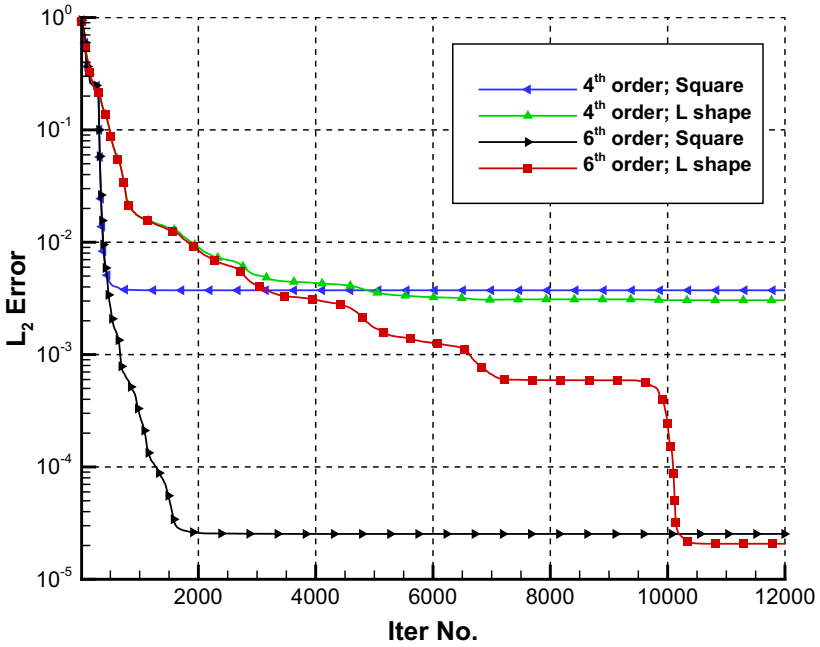

Fig. 13. Problem 2L: L_2-error with the 4th and 6th order schemes for the square and the L-shaped domains, for $k = 100$, $N_g = 15$ and grid 240×240.

Table 4
Time (s) per 1000 iterations on one processor for the square and L-shaped domains with the 6th order scheme, for $k = 50$ and $9 \leqslant N_g \leqslant 18$.

N_g and grid	$9; 73^2$	$12; 97^2$	$15; 120^2$	$18; 144^2$
Square	0.4876	0.8943	1.3715	2.0292
L-shaped	0.3636	0.6753	1.0270	1.5199

Large values of k are problematic with the L-shaped domain. For example, achieving 1% accuracy with $k = 300$ and $N_g = 9$ required 14,465 iterations (205.56 s) on one processor. The best L_2-error for this case was 2.583×10^{-3} (obtained at 32,730 iterations).

5.4. Problem 3–the Marmousi model

The third problem is the well-known Marmousi model [31]. The domain D corresponds to a 6000 m $\times$ 1600 m vertical slice of the Earth's subsurface, with the x-axis as horizontal and the y-axis pointing downwards. A point disturbance source is located at the center of the upper boundary. The wave equation is $\Delta u + k^2 u = g$, where $g(x,y) = \delta(x - (x_{\min} + x_{\max})/2)\delta(y)$. The domain is highly heterogeneous, with velocity c ranging from 1500 m/s to 4450 m/s. Three grid sizes are available: $751 \times 201, 1501 \times 401$, and 2001×534.

This problem was discretized using 2nd and 4th order finite difference schemes, as detailed in Eqs. (2) and (4). The 6th order scheme (5) was not used since k varies in the domain. The Sommerfeld radiation condition, discretized according to (7), was used on all boundaries.

The frequency f is determined by the user, and in our test cases, it ranged from 20 to 80. Since the domain is heterogeneous, the value of N_g varies within the domain, with values determined by the frequency and the grid size. For example, with the mid-sized grid and $f = 40$, we get $9.375 \leqslant N_g(40) \leqslant 27.8125$.

Fig. 14 shows the relative residual results for one to 16 processors, using the 4th order scheme (4), with frequencies of 20, 40 and 60. The following points can be seen:

1. The number of required iterations *decreases* as the frequency increases.
2. As the number of processors increases, more iterations are required for convergence to some specific relative residual.
3. CARP-CG scales better as the frequency increases in the following sense: when the number of processors increases, the percentage increase in the required number of iterations *decreases* with increasing frequency.

Similar results were obtained with the 2nd order scheme.

Unfortunately, there is no known analytic solution for the Marmousi problem. Problems 1 and 2 showed that the 2nd-order scheme achieved very poor error results, even with 18 grid points per wavelength, compared to the high order schemes. We can conclude from this that even in cases with no known analytic solutions, the high order schemes are much closer to the true solution.

In order to evaluate the relative performance of the 2nd and 4th order schemes on this problem, we ran the 4th order scheme on the mid-sized grid, with frequency $f = 40$, until a relative residual of 10^{-13} was achieved. The result was saved as the "true" solution, and this was compared with the results of the 2nd order scheme. Fig. 15 shows the relative L_2-error of the 2nd order scheme w.r.t. to this "true" solution, as a function of the relative residual, for frequency $f = 40$. We can see that the 2nd order scheme comes nowhere near the result of the 4th order scheme. It is clear from this that the solution with the 4th order scheme is much closer to the true solution.

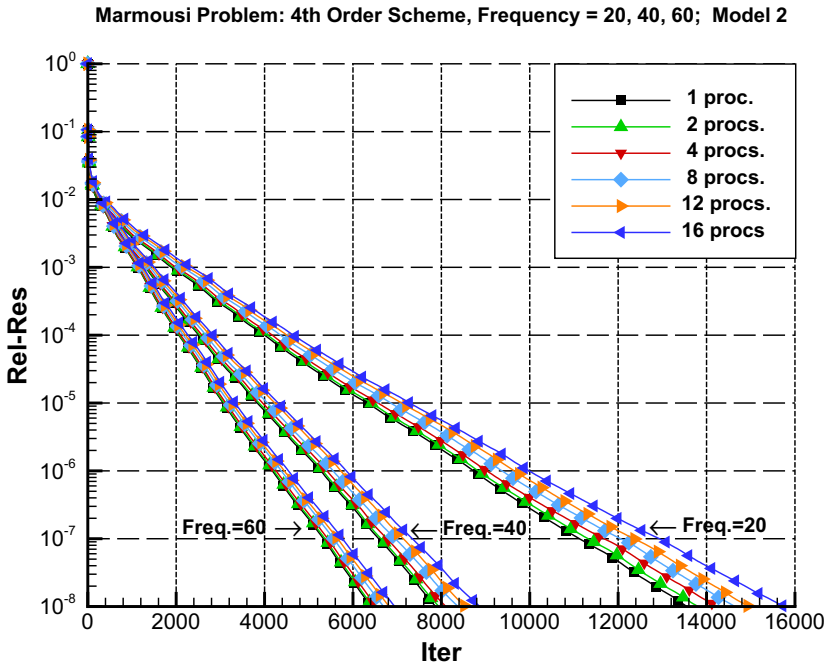

Fig. 14. Problem 3: convergence of the relative residual on 1–16 processors, for frequencies 20, 40 and 60, on the mid-sized grid (1501×401).

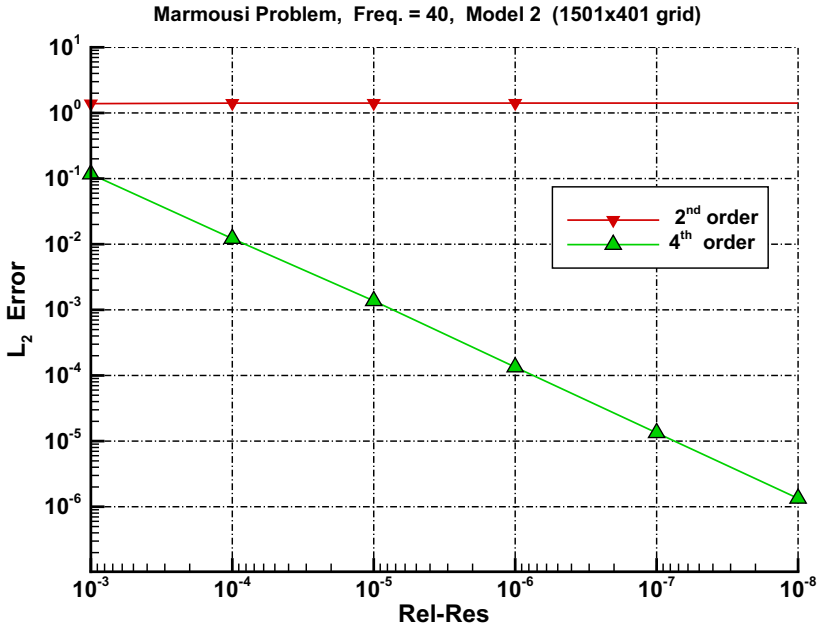

Fig. 15. Problem 3: relative error of the 2nd order scheme compared to the solution obtained with the 4th order scheme, for frequency $f = 40$.

Table 5 shows the number of iterations and runtimes required to reach a relative residual of 10^{-7} with the 4th order scheme on one to 24 processors, on the three grid sizes. The frequency f was chosen so that the number of grid points per wavelength, N_g, is at least 6.5.

6. 3D examples

6.1. Problem 5

We consider the Helmholtz equation $\Delta u + k^2 u = 0$ on the unit cube $D = [0,1]^3$, with the analytic solution

$$u = \cos(\pi x)\sin(\pi y)(\alpha\cos(\beta z) - i\sin(\beta z)), \tag{10}$$

where $\beta = \sqrt{k^2 - 2\pi^2}$ and $\alpha = \beta/k$.

The boundary conditions are as follows. On the side $z = 0$, we take the first order absorbing boundary condition (7), which reduces to $\frac{\partial u}{\partial z} + iku = 0$. On the other sides, we take Dirichlet boundary conditions, as determined by the values of u from (10). It can be verified that u satisfies the Helmholtz equation throughout the domain and the absorbing boundary condition on the side $z = 0$.

We use the 3D, 6th order discretization scheme from [18, Eq. 17], as detailed in Eq. (6). The absorbing boundary condition on the side $z = 0$ is discretized according to [3, Eq. 8], as detailed in Eq. (8). Due to the large number of equations (13,824,000 complex variables with the 240^3 grid), the runtime experiments were done on 12 processors. $k = 100$ in this problem.

Fig. 16 shows the relative residual plots for Problem 4 with the 2nd and 6th order schemes, for $N_g = 9, 12, 15$. The corresponding grids are $144^3, 192^3, 240^3$. The figure shows that the 6th order scheme is advantageous.

Fig. 17 shows the L_2-error plots with the 2nd and 6th order schemes, for the same values of N_g and grid sizes as in the previous figure. Similarly to Problems 1 and 2, we can see that the 2nd order scheme is incapable of producing acceptable results at these values of N_g.

Table 5
Problem 3: Iterations and runtimes (s) to reach a relative residual $\leqslant 10^{-7}$ with the 4th order scheme, on the three grid sizes with 1–24 processors. $N_g \geqslant 6.25$ in all cases.

No. proc.	1	2	4	8	12	16	24
751×201	2734	2752	2838	2964	3053	3153	3420
$f = 30$	(85)	(50)	(28)	(16)	(14)	(14)	(14)
1501×401	5332	5367	5428	5561	5655	5765	5995
$f = 60$	(668)	(389)	(206)	(118)	(97)	(91)	(86)
2001×534	7083	7106	7170	7310	7414	7521	7734
$f = 80$	(1608)	(914)	(489)	(274)	(217)	(212)	(179)

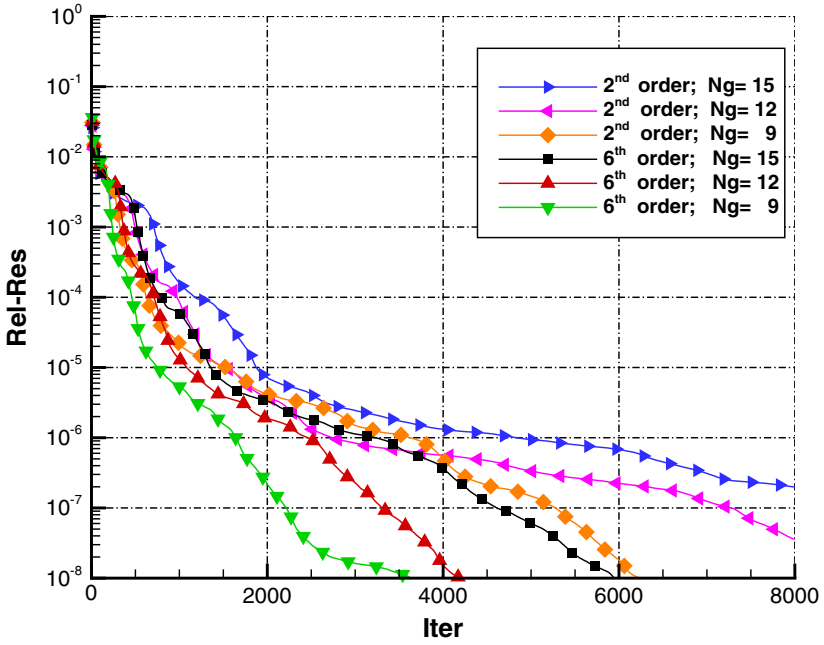

Fig. 16. Problem 4: relative residual of 2nd and 6th order schemes, for $k = 100$ and $N_g = 9, 12, 15$.

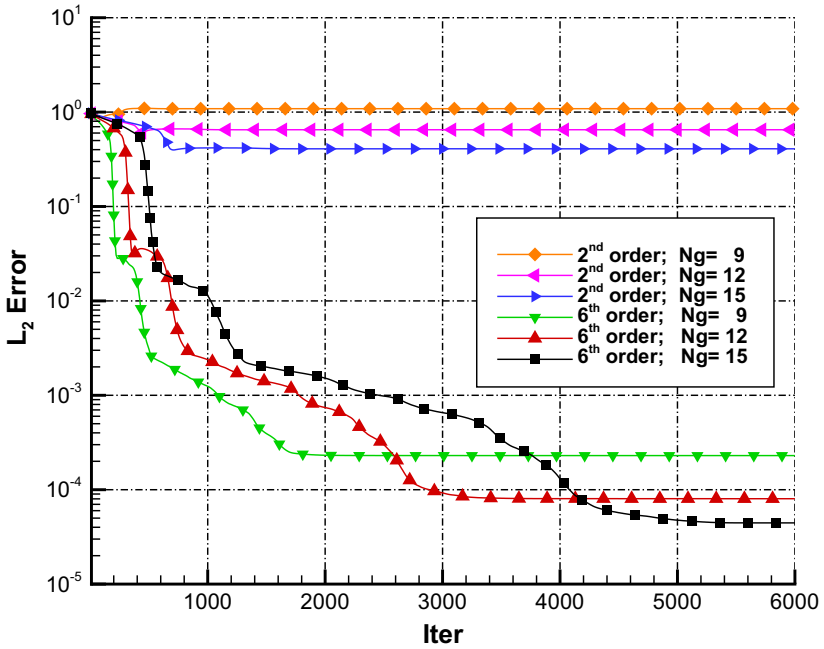

Fig. 17. Problem 4: L_2-error of 2nd and 6th order schemes, for $k = 100$ and $N_g = 9, 12, 15$.

6.2. Problem 5

The Helmholtz equation is taken over the unit cube $[0,1]^3$, with a point disturbance at the center of the $z = 0$ side, so the equation is $\Delta u + k^2 u = \delta(x - \frac{1}{2})\delta(y - \frac{1}{2})\delta(z)$. First order absorbing boundary conditions (7) are taken on all sides. We use the 3D 6th order discretization scheme from [18, Eq. 17], as detailed in Eq. (6). The absorbing boundary conditions are discretized according to [3, Eq. 8], as detailed in Eq. (8).

Fig. 18 shows the convergence plots of Problem 5 for $k = 100$, on 2–16 processors (there was insufficient memory to run the problem on one processor). The problem was ran on grid sizes of $105^3, 145^3$ and 193^3, and $N_g h = 2\pi/k$ was kept fixed, so the corresponding values of N_g were 6.5, 9 and 12. We can see that as N_g decreases, the number of iterations required to

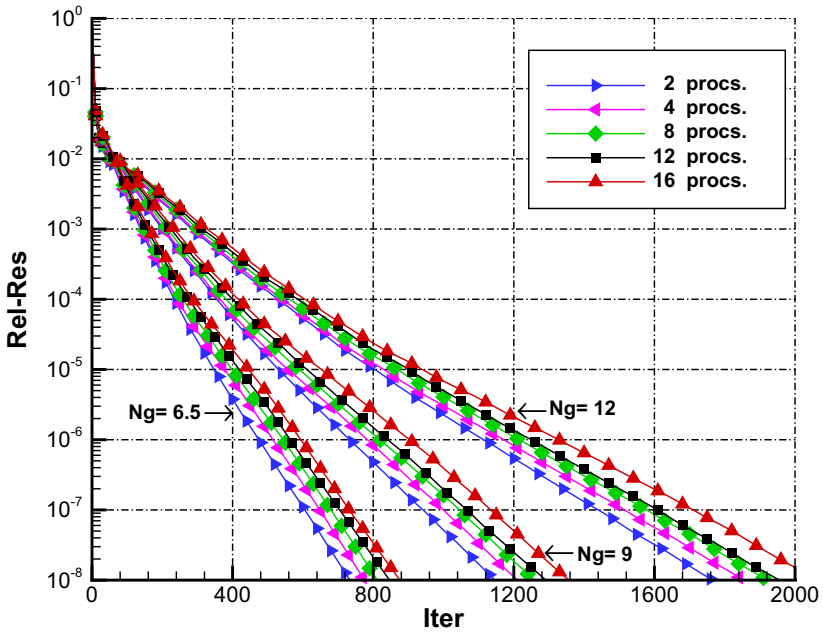

Fig. 18. Problem 5: Convergence plots on 2–16 processors, for $N_g = 6.5, 9, 12$.

reach a certain relative residual also decreases, and the parallel scalability improves (meaning that when we consider the percentage increase in the number of iterations as the number of processors increases, then smaller values of N_g produce lower percentages).

7. Discussion

A natural question that arises in view of the results is the following: the high order schemes require more nonzero elements than the 2nd order scheme, so each iteration takes longer. Suppose we wish to obtain a certain level of accuracy, and we want to do it in the minimal time. Can we do it with the 2nd order scheme faster than with the high order schemes by increasing the grid sizes? As we shall see, for the three problems with analytic solutions that we consider here (1, 2 and 4), the answer is negative.

Table 6 presents the time per iteration of the 2nd and 6th order schemes for Problems 1, 2 and 4 (problems with analytic solution), and shows the percentage increase in time for the 6th order scheme w.r.t. the 2nd order scheme. Problems 1 and 2 were ran on one processor with $k = 300$ and $N_g = 18$ (grid size = 861^2). Problem 4 was ran with $k = 100$ and $N_g = 15$ (grid size = 240^3). We can see that the percentage increase in time per iteration is 40% for the 2D problems and 145% for the 3D problem.

Notwithstanding the fact that the 6th order scheme takes more time per iteration, its accuracy is so much better than that of the 2nd order scheme, that it can achieve the same level of accuracy with much lower values of N_g. This way, the 6th order scheme can achieve any required accuracy with a much smaller grid, and hence, in less time. This is demonstrated in Table 7 for Problems 1, 2 and 4. The table shows the number of iterations and time for the 2nd order scheme to achieve its *minimal*

Table 6
Problems 1, 2 & 4: time (s) per iteration of the 2nd and 6th order schemes, and the percentage increase in time for the 6th order scheme.

Prob.	# Proc.	k	N_g	Grid	2nd order	6th order	% increase(%)
1	1	300	18	861^2	0.119	0.167	40
2	1	300	18	861^2	0.058	0.081	40
4	12	100	15	240^3	0.280	0.686	145

Table 7
Problems 1, 2 & 4: number of iterations and time (s) to reach the minimal L_2-error of the 2nd order scheme, compared to the iterations and time of the 6th order scheme to reach the same L_2-error with a much lower value of N_g.

Prob.	# Proc.	k	L_2-err.	N_g	2nd	order	N_g	6th	order
1	1	300	0.78	18	1650	(**197**)	6.5	109	(**2.23**)
2	1	300	0.41	18	1680	(**97**)	6.5	95	(**0.97**)
4	12	100	0.40	15	710	(**195**)	6.5	95	(**5.76**)

L_2-error, and compares it with the number of iterations and time for the 6th order scheme to achieve the same result with a much lower value of N_g. We can see that the 6th order scheme takes about 1% of the time for the 2D pro blems and 3% of the time for the 3D problem, as compared with the 2nd order scheme. Problems 1 and 2 were ran on one processor and Problem 4 on 12 processors. For $N_g = 6.5$, the grid size is 311^2 for Problems 1 and 2, and 104^3 for Problem 4.

A related question is the following: how much will we have to refine the grid so that the 2nd order scheme can achieve the same error as the 6th order scheme? To answer this, we recall some basic theory. Suppose p is the order of accuracy of some scheme and h is a given mesh spacing. Then the error E obtainable (in theory) with a scheme of order p is proportional to h^p, i.e., for some constant $c, E = ch^p$.

Suppose now that with a given mesh spacing h_1 we can obtain accuracy E_1 with a scheme of order p, and we want to obtain a given accuracy E_2. Let h_2 be the required mesh spacing. Then, for $i = 1, 2$ we have $E_i = ch_i^p$. From this we get

$$h_2 = h_1 \left(\frac{E_2}{E_1}\right)^{\frac{1}{p}}. \tag{11}$$

Example 1. Consider the case of Problem 4 with $p = 2$. The error obtained with the 2nd order scheme for $N_g = 15$ was $E_1 = 0.40$, and it was obtained with a mesh spacing of $h_1 = 1/239$. Suppose we set our error goal as $E_2 = 1.2973 \times 10^{-3}$, which is the error obtained with the 6th order scheme for $N_g = 6.5$. Substituting these numbers in Eq. (11) gives us $h_2 \approx 1/4197$. Hence, the required grid size is 4198^3, which is approximately 63,909 times larger than the grid of 105^3 required by the 6th order scheme. The required number of grid points per wavelength would be $N_g = 2\pi \times 4197/100 \approx 264$.

8. Conclusions

The results show the usefulness of CARP-CG for solving the Helmholtz equation with large wave numbers, using 4th and 6th order finite difference schemes. The high order schemes are clearly preferable over the 2nd order scheme, which requires unrealistic mesh sizes to achieve comparable results. Both high order schemes take the same time per iteration, so the 6th order scheme is preferable when the wave number is constant. Furthermore, the parallel scalability of CARP-CG improves when the wave number is increased (on a fixed grid), or when, for a fixed wave number, the grid size is decreased.

Results on an L-shaped domain indicate that CARP-CG coupled with high order schemes can handle this difficult problem. However, for large wave numbers, the number of iterations required to reach a given error goal can be much greater than for the full square. This suggests that mesh refinement in the vicinity of the corner could be a useful approach.

Acknowledgement

The authors thank Dan Givoli and Eli Turkel for several helpful discussions. Thanks are also due to the anonymous reviewers for their useful comments, including the suggestion of an L-shaped domain.

References

[1] A. Bayliss, C.I. Goldstein, E. Turkel, On accuracy conditions for the numerical computation of waves, Journal of Computational Physics 59 (1985) 396–404.
[2] I.M. Babuška, S.A. Sauter, Is the pollution effect of the FEM avoidable for the Helmholtz equation considering high wave numbers?, SIAM Review 42 (2000) 451–484
[3] Y.A. Erlangga, E. Turkel, Iterative schemes for high order compact discretizations to the exterior Helmholtz equation, ESAIM: Mathematical Modelling and Numerical Analysis 46 (3) (2012) 647–660.
[4] A. Bayliss, C.I. Goldstein, E. Turkel, An iterative method for the Helmholtz equation, Journal of Computational Physics 49 (1983) 443–457.
[5] Y.A. Erlangga, C. Vuik, C.W. Oosterlee, On a class of preconditioners for solving the Helmholtz equation, Applied Numerical Mathematics 50 (2004) 409–425.
[6] Y.A. Erlangga, Advances in iterative methods and preconditioners for the Helmholtz equation, Archives of Computational Methods in Engineering 15 (2008) 37–66.
[7] I. Duff, S. Gratton, X. Pinel, X. Vasseur, Multigrid based preconditioners for the numerical solution of two-dimensional heterogeneous problems in geophysics, International Journal of Computer Mathematics 84 (8) (2007) 1167–1181.
[8] H.C. Elman, O.G. Ernst, D.P. O'Leary, A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations, SIAM Journal on Scientific Computing 23 (4) (2001) 1291–1315.
[9] R.-E. Plessix, W.A. Mulder, Separation-of-variables as a preconditioner for an iterative Helmholtz solver, Applied Numerical Mathematics 44 (2003) 385–400.
[10] M. Bollhöfer, M.J. Grote, O. Schenk, Algebraic multilevel preconditioner for the Helmholtz equation in heterogeneous media, SIAM Journal on Scientific Computing 31 (5) (2009) 3781–3805.
[11] D. Osei-Kuffuor, Y. Saad, Preconditioning Helmholtz linear systems, Applied Numerical Mathematics 60 (2010) 420–431.
[12] D. Gordon, R. Gordon, CARP-CG: a robust and efficient parallel solver for linear systems applied to strongly convection-dominated PDEs, Parallel Computing 36 (9) (2010) 495–515.
[13] D. Gordon, R. Gordon, Robust and Highly Scalable Parallel Solution of the Helmholtz Equation with Large Wave Numbers, in: Technical Report, Department of Computer Science, University of Haifa, Israel, Feb 2012, <http://cs.haifa.ac.il/~gordon/helm.pdf>.
[14] D. Gordon, R. Gordon, CGMN revisited: robust and efficient solution of stiff linear systems derived from elliptic partial differential equations, ACM Transactions on Mathematical Software 35 (3) (2008) 18:1–18:27.

[15] D. Gordon, R. Gordon, Solution methods for linear systems with large off-diagonal elements and discontinuous coefficients, Computer Modeling in Engineering and Sciences 53 (1) (2009) 23–45.
[16] I. Harari, E. Turkel, Accurate finite difference methods for time-harmonic wave propagation, Journal of Computational Physics 119 (1995) 252–270.
[17] I. Singer, E. Turkel, Sixth order accurate finite difference schemes for the Helmholtz equation, Journal of Computational Acoustics 14 (2006) 339–351.
[18] G. Sutmann, Compact finite difference schemes of sixth order for the Helmholtz equation, Journal of Computational and Applied Mathematics 203 (2007) 15–31.
[19] D. Gordon, R. Gordon, Parallel solution of high-frequency Helmholtz equations using high-order finite difference schemes, in: Proceedings of the 10th International Conference on Mathematical and Numerical Aspects of Waves, Pacific Institute for Mathematical Sciences, Vancouver, Canada, 2011, pp. 331–334. <http://www.sfu.ca/uploads/page/02/WAVESProceedingsFINAL.pdf>.
[20] S. Kaczmarz, Angenäherte Auflösung von Systemen linearer Gleichungen, Bulletin de l'Académie Polonaise des Sciences et Lettres A35 (1937) 355–357.
[21] Å. Björck, T. Elfving, Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations, BIT 19 (1979) 145–163.
[22] D. Gordon, R. Gordon, Component-averaged row projections: a robust block-parallel scheme for sparse linear systems, SIAM Journal on Scientific Computing 27 (2005) 1092–1117.
[23] O.B. Widlund, On the effects of scaling of the Peaceman–Rachford method, Mathematics of Computation 25 (113) (1971) 33–41.
[24] D. Gordon, R. Gordon, Row scaling as a preconditioner for some nonsymmetric linear systems with discontinuous coefficients, Journal of Computational and Applied Mathematics 234 (12) (2010) 3480–3495.
[25] H.A. van der Vorst, Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 13 (1992) 631–644.
[26] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 7 (1986) 856–869.
[27] A. Sommerfeld, Partial Differential Equations in Physics, Academic Press, New York, 1964.
[28] R.S. Tuminaro, M.A. Heroux, S.A. Hutchinson, J.N. Shadid, in: AZTEC user's guide, Tech. Rep. SAND99-8801J, Sandia National Laboratories, Albuquerque, New Mexico, 1999.
[29] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., SIAM, Philadelphia, PA, 2003.
[30] D. Day, M.A. Heroux, Solving complex-valued linear systems via equivalent real formulations, SIAM Journal on Scientific Computing 23 (2) (2001) 480–498.
[31] The Marmousi Model. Workshop on Practical Aspects of Seismic Data Inversion. in: Proc. 52nd Annual Meeting of the European Association of Geophysicists and Engineers, 1990.