JOURNAL OF COMPUTATIONAL PHYSICS **139,** 343–358 (1998)
ARTICLE NO. CP975874

An Efficient ADI-Solver for Scattered Data Problems with Global Smoothing

Erlend Arge* and Angela Kunoth†,[1]

**Numerical Objects AS, Forskningsveien 1, P.O. Box 124, Blindern, N-0314 Oslo, Norway;*
†Institut für Geometrie und Praktische Mathematik, RWTH Aachen,
Templergraben 55, 52056 Aachen, Germany
E-mail: sea@nobjects.com, kunoth@igpm.rwth-aachen.de

Received December 19, 1996; revised October 7, 1997

For the approximate representation of large data sets over a parameter domain in $\mathbb{R}^2$, many geological and other applications require the construction of surfaces which have minimal energy, i.e., minimal curvature. One way to achieve this is by solving a fourth-order elliptic partial differential equation. Its discretization by a difference scheme makes it particularly easy to incorporate (appropriate approximations of) known data points. In this paper, we investigate the performance of different solution methods for the resulting symmetric linear system of equations since this is the most CPU-demanding step in the scattered data approximation procedure. Specifically, we test first the performance of a preconditioned conjugate gradient method with an SSOR and an RILU preconditioner. However, since the partial differential operator does not contain mixed derivatives, using an alternating-direction-implicit scheme (ADI method) which has been employed successfully in the past for second-order problems, together with a Cholesky factorization of the corresponding one-dimensional operators has also been tried for the problem at hand. The computational studies that we have performed here show that for our problem an instationary ADI method is superior to the above-mentioned preconditioned conjugate gradient solvers both with respect to work load and accuracy of the solution. For the fourth-order model problem considered in this paper, the instationary ADI method with Wachspress parameters results in a number of iterations that is essentially independent of the number of variables. © 1998 Academic Press

Key Words: scattered data approximation; fourth-order elliptic problems; difference methods; preconditioned conjugate gradient methods; ADI methods.

[1] Corresponding author. The work of this author was supported by the European Community within the project "Human Capital and Mobility" under Contract ERB CHBG CT 93 04 08.

1. INTRODUCTION

To represent surfaces approximating scattered data, many users prefer *grid functions*. These are discrete functions defined on a grid overlapping $\Omega \subset \mathbb{R}^2$ in the parameter domain with equally spaced nodes in each coordinate direction called a *regular grid*. By using spline interpolation methods as described, e.g. in [2], from the grid functions one can easily obtain continuous functions on $\mathbb{R}^2$ for visualization and other purposes.

To generate a grid function on a given regular grid that approximates a large set of scattered data, a method consisting of three steps has been developed in [3]. In the first and second stages called *regularization* and *approximation*, the grid function is assigned function values at nodes lying in regions with high data density by a local approximation scheme. The last step, denoted by *extrapolation*, determines the remaining function values of the grid function in a global fashion by minimizing its "curvature" (in a sense yet to be made precise).

This method does not interpolate the given scattered data set. Its main area of application is rather in situations where there are large point sets that should be approximated and the data should not be interpolated. Such data occurs in many applications, and the three-step procedure for scattered data approximation described above is used in several commercial products through the SINTEF Scattered Data Library (SISCAT) [5, 18]. One example of an application area for this method is the construction of a geological surface over a bivariate regular grid, where the data is given on contours or seismic tracks.

Due to the typically large amount of resulting data, methods like using radial basis functions (e.g., thin plate splines [16]) cannot be employed any more for the construction of a grid function having minimal energy. The extrapolation process given in [3] yields to the problem of solving a fourth-order symmetric elliptic partial differential equation. This equation is discretized using a difference method which makes it particularly easy to incorporate the scattered data conditions, leading to a large system of linear equations. Since the system matrix P is symmetric, positive definite, and sparse, a natural choice to solve the linear equations is to use a *conjugate gradient method* (CG method). Unfortunately, the convergence speed of this iterative method is disturbingly low due to its dependence on the spectral condition number of P which in the present case is of order $O(h^{-4})$, where h is the grid spacing. Therefore, preconditioning with the aim of reducing the condition number becomes an essential task, thus resulting in the *preconditioned conjugate gradient method* (PCG method). Here we have used two standard preconditioners in the PCG method which both reduce the condition number to $O(h^{-2})$. The first one is an *SSOR preconditioner* which is based on an additive decomposition of P. The second choice is the *RILU preconditioner* derived from an incomplete factorization of P into triangular matrices (see, e.g., [8 or 12].

However, it turns out in the computational studies that an instationary *alternating-direction-implicit method* (ADI method) is superior to the above-mentioned PCG methods for the present problem with respect to the overall amount of work and the reduced error. In fact, the number of iterations for the ADI method grows very slowly as a function of the number of variables which is not at all the case for the PCG schemes. In the ADI approach the matrix P is decomposed additively into its parts from the difference operators in each coordinate direction in the parameter domain. The iteration is based on alternatingly solving the corresponding linear systems based on one-dimensional fourth-order problems *exactly* (after reordering of nodes) by computing the inverses of these *pentadiagonal* matrices directly by means of Cholesky decomposition. A description of all these methods, although only for second-order problems, together with convergence proofs can be found, e.g., in [12].

The ADI method as described here has also been applied to linear systems arising from scattered data problems with discontinuities as developed in [4]. In this paper, we will focus on a model problem capturing the main difficulties of the extrapolation problem of the scattered data approximation method, and we will not consider the discontinuities here. First tests [20] indicate that the method still works well, even though one of the main theoretical requirements, the commutativity condition (21) below of the difference operators in each coordinate direction is violated.

This paper is structured as follows. In the next section, we briefly recall the regularization and approximation steps of the scattered data procedure from [3] and give a short review of the extrapolation step. Based on some observations regarding the main difficulties in solving the linear equations connected to the extrapolation step, we formulate a model problem in Section 3. In Section 4, we briefly discuss the preconditioned conjugate gradient method as well as give a description of the stationary and instationary ADI methods and apply them to the fourth-order problem at hand. In Section 5 we experiment numerically with these solvers on the model problem in Section 3. Section 6 gives some concluding remarks.

2. REVIEW OF THE SCATTERED DATA METHOD

Following the ideas for the construction of grid functions in [3], let $\{(s_k, y_k) \in \mathbb{R}^2 \times \mathbb{R}, k$ in some finite index set $K\}$ be a given set of scattered data. Let further $G := \Omega \cap (h_1\mathbb{Z} \times h_2\mathbb{Z})$ be a grid with grid spacing $h_1, h_2 > 0$, where Ω is some bounded rectangular domain in $\mathbb{R}^2$. We assume that Ω contains the scattered points $S := \{s_k, k \in K\}$.

In the *regularization* step, a subset $D \subset G$ is determined by requiring that in a region around each $\alpha \in D$ there are sufficiently many data points. The precise construction is that a density function describing the scattered data density at each point in Ω is determined. Then the nodes in D are those in G where the density exceeds a certain threshold value. For a discussion of the regularization step, see [14].

In the *approximation* step, a data set $\{(\alpha, z_\alpha) \in \mathbb{R}^2 \times \mathbb{R}, \alpha \in D\}$ is computed by the application of some local approximation scheme. Choosing for each $\alpha \in D$ a local approximation operator depending on a relatively small subset of $\{(s_k, y_k), k \in K\}$ in the neighborhood of α then yields values for the grid function on D. For this purpose, one may, e.g., employ Shepard's method [17] or a polynomial least squares approach, or use radial basis functions like thin plate splines [16].

In the *extrapolation* step, the values on $U = G\backslash D$ are computed by employing a global scheme for minimizing a measure of the curvature of the surface under the restriction of interpolating the values on D. The various point sets are shown in Fig. 1.

Minimizing the curvature of a continuous function u over a bivariate domain Ω at each point of u in every direction leads to considering the problem

$$\min_{u \in \mathcal{C}^2(\Omega)} I(u)$$

at $r = 0$, where

$$I(u) = \int_\Omega \int_0^\pi w(\phi) \left(\frac{\partial^2}{\partial r^2} u(x + r\cos\phi, y + r\sin\phi) \right)^2 d\phi\, dx\, dy. \tag{1}$$

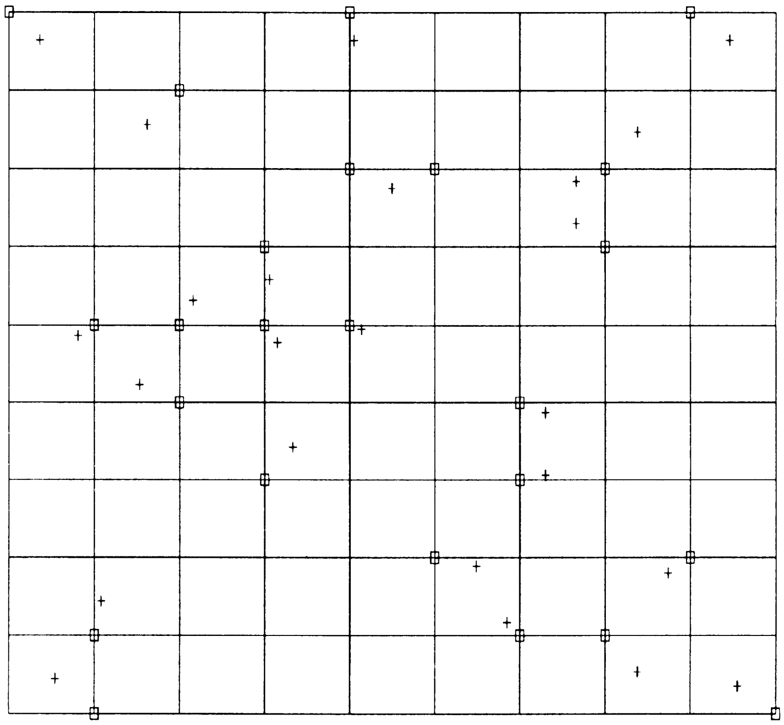

FIG. 1. This figure explains the notation used to define the different sets of points. The entire grid is denoted by G, the scattered data points, marked by "+" are denoted by S, the grid points located close to the data points are denoted by D and are marked by "□," and finally the set of grid points not close enough to any data point are unmarked and denoted by $U = G \backslash D$.

Here $w(\phi), 0 \leq \phi < \pi$, is some weight distribution. We note that with $w \equiv 1$ the Euler equation for (1) is essentially the same as the Euler equation for the thin plate spline functional [10],

$$I(u) = \int_\Omega ((u_{xx})^2 + 2(u_{xy})^2 + (u_{yy})^2)\, dx\, dy, \tag{2}$$

where we abbreviate $f_x := \partial f/\partial x,\ f_y := \partial f/\partial y$ for any sufficiently smooth function $f = f(x, y)$. Thus, (1) can be viewed as a generalization of (2).

For our purpose we choose

$$w(\phi) = \delta_0(\phi) + \delta_{\pi/2}(\phi),$$

where δ_x is the delta distribution. Thus, the functional becomes

$$I(u) = \int_\Omega ((u_{xx})^2 + (u_{yy})^2)\, dx\, dy \tag{3}$$

which means that the curvature is minimized along the x- and y-axes of the parameter domain. This functional is very well suited for the modeling of faulted geological surfaces [4] and, as will see later on, leads to a system of linear algebraic equations which are amenable to being solved by an ADI method. For more general weights w see [13].

The minimization of $I(u)$ as defined in (3) leads to the Euler equations

$$\int_{\Omega} (u_{xx} v_{xx} + u_{yy} v_{yy})\, dx\, dy = 0 \tag{4}$$

for any bounded test function $v \in \mathcal{C}^2(\Omega)$. One approach to solve (4) would be to employ finite elements in such a Galerkin-type formulation. We will, however, continue to work in the grid setting and compute the values on U directly through a finite difference approach since this makes it easy to incorporate the known values on D.

Assuming now sufficiently high smoothness of u, we can integrate Eq. (4) by parts to obtain

$$\int_{\Omega} (u_{xxxx} + u_{yyyy}) v\, dx\, dy = 0. \tag{5}$$

Here, we assume either that u satisfies the natural boundary conditions (i.e., second- and third-order normal derivatives vanish) or that the test function is restricted to some appropriate subspace of $\mathcal{C}^2(\Omega)$. Of course, without posing any further conditions on u, any cubic polynomial satisfies (5).

Since our goal is to interpolate the data given on $D \subset G \subset \Omega$, this leads to the requirement for solving

$$\begin{aligned} u_{xxxx} + u_{yyyy} &= 0 \qquad \text{in } \Omega \backslash D, \\ u(\alpha) &= z_\alpha, \quad \alpha \in D, \end{aligned} \tag{6}$$

and in addition some natural boundary conditions, if necessary, to guarantee uniqueness of u.

To discretize (6), we will approximate the derivatives by finite differences on the grid $G = \Omega \cap (h_1\mathrm{Z} \times h_2\mathrm{Z})$. The discretized differential operator should only be applied to the nodes of $U = G \backslash D$ as the interpolation conditions are to be fulfilled on D. This leads to the $N = \#U$ equations

$$\frac{1}{h_1^4}(Hz)_\alpha + \frac{1}{h_2^4}(Vz)_\alpha = 0, \quad \alpha \in U, \tag{7}$$

where the horizontal and vertical difference operators H and V are defined by

$$(Hz)_\alpha = z_{\alpha - 2d^1} - 4z_{\alpha - d^1} + 6z_\alpha - 4z_{\alpha + d^1} + z_{\alpha + 2d^1}$$

and

$$(Vz)_\alpha = z_{\alpha - 2d^2} - 4z_{\alpha - d^2} + 6z_\alpha - 4z_{\alpha + d^2} + z_{\alpha + 2d^2}$$

with direction vectors $d^1 = (1, 0)^T$ and $d^2 = (0, 1)^T$.

We will in the following assume that $h_1 = h_2$, in which case we might replace (7) by $(Hz)_\alpha + (Vz)_\alpha = 0$. This leads to the definition of the difference operator,

$$(Pz)_\alpha = (Hz)_\alpha + (Vz)_\alpha; \tag{8}$$

i.e., P can be identified with a 9-point-difference stencil of the form

$$\begin{bmatrix} & & 1 & & \\ & & -4 & & \\ 1 & -4 & 12 & -4 & 1 \\ & & -4 & & \\ & & 1 & & \end{bmatrix}. \tag{9}$$

Since G is a finite grid, the stencil has to be modified close to the boundary of Ω. For example, if $\alpha \in U$ and $\alpha + 2d^1 \notin G$, the horizontal operator H could be modified to

$$(Hz)_\alpha = z_{\alpha-2d^1} - 4z_{\alpha-d^1} + 5z_\alpha - 2z_{\alpha+d^1},$$

and if also $\alpha + d^1 \notin G$, it would read

$$(Hz)_\alpha = z_{\alpha-2d^1} - 2z_{\alpha-d^1} + z_\alpha.$$

These alterations correspond to a discretization of the natural boundary conditions $u_{xxx} = u_{xx} = 0$ across a vertical boundary. Similar conditions can be posed on V. For further details on such discretizations see [1, p. 137].

Given the above discretization of boundary conditions, together with mild restrictions on the number and location of the nodes in D, it is shown in [4] that the operator P is symmetric and positive definite on the space of grid functions supported in U; i.e., for any grid functions u and v such that $u_\alpha = v_\alpha = 0$ for all $\alpha \notin U$ we have

$$(Pu, v) = (u, Pv),$$
$$(Pu, u) > 0, \quad u \neq 0,$$

where $(\cdot, \cdot)$ denotes the Euclidean inner product. The conditions for positive definiteness are that there exist at least four nodes in D that are not zeroes of any bilinear function which is a very weak condition. In fact, the results in [4] also cover the introduction of discontinuities in the grid function (faults) treated as internal boundary conditions toward the faults. Thus, corresponding discretizations always yield a system which is uniquely solvable.

Since P is symmetric, positive definite, and sparse, a natural approach for solving the linear equations (7) is to use some kind of iterative method like the conjugate gradient iteration. Without preconditioning, this can be implemented simply by storing z in a matrix and applying the stencil (9) to z in order to perform the matrix–vector product, i.e., it is not necessary to assemble the matrix. This is done in [3], where it is indeed observed in the computations that the number of conjugate gradient iterations grows very fast as a function of the size of the grid as predicted by the theory. In fact, for a sparse data set D and a fine grid G, the spectral condition number of P is $O(N^2)$, where $N = \#U$. The convergence rate of the CG method is then close to zero (cf. (13)), which means that there is hardly any improvement in each iteration step when N is large. While N is smaller and thus also the condition number of P when more data D is given, the convergence rate also depends on the distribution of the data which is reflected in the constants in $O(N^2)$. In view of this observation, the model problem given in the next section is chosen such that all the given data is assembled around the boundary. Thus, the example is rather “ill-conditioned” in the sense that the constants in $O(N^2)$ will be quite large.

3. A MODEL PROBLEM

Given an integer $n \geq 2$, let $h = 1/(n-1)$. In the model problem we will use a domain Ω of the form $\Omega = [-2h, 1+2h]^2$ and we define $G = \Omega \cup h\mathrm{Z}^2$. Furthermore, let

$$D = \{\alpha \in G : \alpha \notin [0,1]^2\},$$

and define $U = G \backslash D$. Then U consists of $N = n \times n$ nodes on a uniform grid covering the unit square. The data nodes D lie on two more outer grid lines parallel to the boundaries of the unit square. The input data will be sampled from a test function on D; thus we do not consider the regularization and approximation steps of the scattered data approximation algorithm. Neither do we consider derivative-type boundary conditions. The boundary conditions will be of Dirichlet type, clamping the value and the cross boundary derivative of the solution at the boundary.

With this setup we are able to study the effect of approximating scattered data sets containing regions with little data. Large values of n are to be interpreted as large "holes" in a scattered data set.

Let f be some test function and define

$$z_\alpha = f(\alpha) \quad \text{for all } \alpha \in D. \tag{10}$$

To compute z_α for $\alpha \in U$ we must solve the linear system

$$(Pz)_\alpha = (Hz)_\alpha + (Vz)_\alpha = 0, \quad \alpha \in U. \tag{11}$$

In order to apply the equation solvers in the next section, we will need to formulate (11) by assembling the coefficient matrix. For this purpose let $b_\alpha = -(P\bar{z})_\alpha$, where $\bar{z}_\alpha = f(\alpha)$ if $\alpha \in D$ and $\bar{z}_\alpha = 0$ otherwise. Then a grid function z with $z_\alpha = 0$ for all $\alpha \in D$ solves our problem if

$$(Pz)_\alpha = (Hz)_\alpha + (Vz)_\alpha = b_\alpha, \quad \alpha \in U.$$

Enumerating the grid nodes *vertically*, beginning with the lower left corner of U, one obtains a linear system of equations

$$Pz = (H+V)z = b, \tag{12}$$

where we keep the notation P, H, and V for the corresponding $N \times N$ matrices. With the vertical ordering V has a dense band of band width 5. That is, V has two side diagonals on each side of the main diagonal. It is also block diagonal with blocks of size $n \times n$, each block corresponding to the one variable problem induced by a vertical grid line. The matrix H has band width $5(n-1)$. However, using a *horizontal* ordering the situation for H and V is reversed.

For the PCG methods used in the next section, the ordering is not of primary importance since we will only need to perform matrix–vector products, and hence, we can store P in a matrix format supporting sparse matrices. However, for the ADI method we will use Cholesky factorizations of matrices related to H and V, and with the vertical ordering this would create fill-in in H. To avoid this we will use a vertical ordering for V and a horizontal ordering for H in the implementation of the ADI method.

4. SOLVING THE SYSTEM OF LINEAR EQUATIONS

4.1. *Preconditioned Conjugate Gradient Methods*

The convergence speed R of the CG method (Algorithm 4.1 below with $C = I$, the identity matrix) depends on the spectral condition number

$$\kappa(P) := \frac{\lambda_{\max}(P)}{\lambda_{\min}(P)} = O(h^{-4}), \tag{13}$$

where $\lambda_{\max}(P)$ and $\lambda_{\min}(P)$ are the maximal and minimal eigenvalues of P, in such a way that it slows down exponentially with decreasing grid spacing, i.e.,

$$R = -\log\left|\frac{\sqrt{\kappa(P)}-1}{\sqrt{\kappa(P)}+1}\right|. \tag{14}$$

An example of this dramatic effect for the present situation is given in [3]. The fourth power of h in (13) relates to the fact that the differential operator in (8) is of order four.

It is therefore of primary importance to precondition the linear system, i.e., to replace (12) by

$$CPz = Cb,$$

where C is some symmetric positive definite matrix approximating P^{-1}. One would call C a "good" preconditioner for P if $\kappa(CP) \ll \kappa(P)$ and if the matrix–vector product CP is not too costly, typically of the same computational complexity $O(N)$ as for performing Pz.

Below we provide an outline of the preconditioned conjugate gradient method as it can be found in, e.g. [12].

Algorithm 4.1. PCG method.

```
$r \leftarrow b - Pz$, $z$ start vector
$g \leftarrow Cr$
$p \leftarrow g$
$\sigma \leftarrow (r, g)$
residual_error $\leftarrow \sqrt{(r,r)}$
while (residual_error $>$ error_wanted)  {
        $\alpha \leftarrow \frac{\sigma}{(p, Pp)}$
        $z \leftarrow z + \alpha p$
        $r \leftarrow r - \alpha Pp$
        $g \leftarrow Cr$
        $\sigma_{\mathtt{new}} \leftarrow (r, g)$
        $\beta \leftarrow \frac{\sigma_{\mathtt{new}}}{\sigma}$
        $\sigma \leftarrow \sigma_{\mathtt{new}}$
        $p \leftarrow g + \beta p$
        residual_error $\leftarrow \sqrt{(r,r)}$
        }
```

Here $(\cdot, \cdot)$ denotes as before the Euclidean inner product.

In the examples in Section 5 we have set C to be the SSOR preconditioner, derived from the classical successive overrelaxation iteration for symmetric matrices, and the RILU preconditioner, based on incomplete factorization. For fourth-order problems, one expects then $\kappa(CP) = O(h^{-2})$. For more information on these and other types of preconditioners, see [8 or 12].

4.2. *ADI Methods*

An alternative approach to solve (12) which is somewhat different from the PCG method with SSOR or RILU preconditioner is motivated by the fact that the stencil (9) does not contain mixed derivatives. The idea is to define an iteration based on the splitting $P = H + V$, leading to the ADI method. The theory for the ADI method in this subsection, as it was developed for second-order problems, has been taken from the survey [6], where also all the proofs of the statements and a number of examples and computations (although from more than 30 years ago) can be found.

Introducing an iteration parameter $\rho > 0$, we may write Eq. (12) as

$$Pz = (H + \rho I + V - \rho I)z = b$$

which gives the relations

$$z = (V + \rho I)^{-1}(b - (H - \rho I)z),$$
$$z = (H + \rho I)^{-1}(b - (V - \rho I)z).$$

Usually the matrices H and V are assumed to be symmetric and positive semi-definite. For the model problem in Section 3 they are actually positive definite. Therefore, for any $\rho > 0$, $V + \rho I$ and $H + \rho I$ are positive definite and, consequently, their inverses are well defined.

Adopting the notion *stationary* ADI method from [12, p. 197], the procedure with a single iteration parameter ρ reads as follows.

Algorithm 4.2 (Stationary ADI method).

```
$z$ start vector, $\rho$ iteration parameter
$r \leftarrow b - Pz$
residual_error $\leftarrow \sqrt{(r,r)}$
while (residual_error > error_wanted)  {
        $z \leftarrow (V + \rho I)^{-1}(b - (H - \rho I)z)$
        $z \leftarrow (H + \rho I)^{-1}(b - (V - \rho I)z)$
        $r \leftarrow b - Pz$
        residual_error $\leftarrow \sqrt{(r,r)}$
        }
```

By ordering the grid points vertically when assembling V and horizontally when assembling H as mentioned before, both matrices will be pentadiagonal; i.e., they have the four side diagonals directly adjacent to the main diagonal (band-width 5). The same ordering principles are used for $H + \rho I$ and $V + \rho I$. The inverses of $H + \rho I$ and $V + \rho I$ are determined by Cholesky decomposition ahead of the iteration loop. Since these are 5-banded, the decompositions are performed quickly without fill-ins.

The application of the inverses inside the iteration loop are performed by forward–backward substitutions. Again by the 5-banded structure, these actions are $O(N)$ operations. By using a vertical ordering of the N-vector, we must, however, reorder the vectors into a horizontal ordering before an operation involving H is performed.

It can be shown that Algorithm 4.2 converges for any fixed iteration parameter $\rho > 0$ [6]. In fact, every stationary iterative method $x \leftarrow Tx + b$ converges if and only if the *spectral radius* $\Lambda(T) := \max_\ell |\lambda_\ell(T)|$ of the *error reduction matrix* T satisfies $\Lambda(T) < 1$, where $\lambda_\ell(T)$ are the eigenvalues of T (see, e.g., [11]). In Algorithm 4.2, one has $T = (V + \rho I)^{-1}(H - \rho I)(H + \rho I)^{-1}(V - \rho I)$ for which it can be shown with arguments from linear algebra that its spectral radius is less than one for any positive ρ since H, V, and ρI are real symmetric positive definite matrices [6, p. 195].

According to [6], the optimal choice of ρ depends on the smallest and biggest eigenvalues of H and V. In our implementation we computed these eigenvalues using power iterations (cf. [11]). It turned out that the eigenvalues could be found with very few iterations; in the example in Section 5 three power iterations proved to be sufficient to determine satisfactory approximations to the maximum and minimum eigenvalues.

Now, denote by a_H, b_H and a_V, b_V the minimal and maximal eigenvalues of H and V, respectively. Define

$$F_1 := \left(\frac{b_H - \sqrt{a_H b_H}}{b_H + \sqrt{a_H b_H}}\right)\left(\frac{b_V - \sqrt{a_H b_H}}{b_V + \sqrt{a_H b_H}}\right),$$

$$F_2 := \left(\frac{\sqrt{a_V b_V} - a_H}{\sqrt{a_V b_V} + a_H}\right)\left(\frac{b_V - \sqrt{a_V b_V}}{b_V + \sqrt{a_V b_V}}\right).$$

If $F_1 \leq F_2$ then the optimal choice for the iteration parameter is

$$\rho := \sqrt{a_H b_H}; \tag{15}$$

otherwise, one should set

$$\rho := \sqrt{a_V b_V}. \tag{16}$$

With such ρ, the convergence rate of the stationary ADI method, Algorithm 4.2, is then at least

$$R = -\log F, \quad F := \min\{F_1, F_2\}.$$

For the situation at hand of one-dimensional fourth-order differential operators H and V, we typically have

$$a_V \sim a_H \sim h^4, \quad b_V \sim b_H \sim 1. \tag{17}$$

Here $X \sim Y$ means that there exist constants c_1, c_2 independent of any parameters X or Y may depend on such that $c_1 Y \leq X \leq c_2 Y$, and h is the grid spacing as in (13). Thus, $\rho \sim h^2$ with either of the two choices (15) or (16), and the convergence rate of the stationary ADI method is then at least

$$R = -\log\left(\frac{c_1 - h^2 c_2}{c_1 + h^2 c_2}\right)\left(\frac{c_3 - h^2 c_4}{c_3 + h^2 c_4}\right)$$

with some constants c_1, c_2, c_3, c_4. Thus, if $h \sim 2^{-j}$ and j large, the speed of convergence will be comparably slow. Tests for second-order problems performed in [6] show that the stationary ADI method behaves essentially like the classical successive overrelaxation method.

However, the rate of convergence of ADI methods can be considerably improved by using a possibly different iteration parameter in each iteration step (see [6 or 1, pp. 206–212]). Following again [12], we call the resulting procedure an *instationary* ADI method. The algorithm for determining these parameters denoted now by ρ_i, $i = 1, \ldots, m$, for some fixed $m \in \mathbb{N}$ can be formulated as follows and has to be done only once.

Algorithm 4.3 (Determination of iteration parameters).

1. Let $\mathbf{a} := \min\{a_H, a_V\}$ and $\mathbf{b} := \max\{b_H, b_V\}$ where a_H, a_V, b_H, b_V are the minimal and maximal eigenvalues of H and V, respectively, and set

$$\mathbf{c} = \frac{\mathbf{a}}{\mathbf{b}}.$$

2. Find the smallest integer m such that

$$(\sqrt{2} - 1)^{2m} \leq \mathbf{c}. \tag{18}$$

3. Determine ρ_i, $i = 1, \ldots, m$, as

$$\rho_i = \mathbf{b}\left(\frac{\mathbf{a}}{\mathbf{b}}\right)^{(2i-1)/2m}, \quad i = 1, \ldots, m, \tag{19}$$

or

$$\rho_i = \mathbf{b}\left(\frac{\mathbf{a}}{\mathbf{b}}\right)^{(i-1)/(m-1)}, \quad m \geq 2, \quad i = 1, \ldots, m. \tag{20}$$

The parameters determined in (19) are called *Peaceman–Rachford parameters*, while the ones in (20) are commonly referred to as *Wachspress parameters*. These parameters are now used successively in a cyclic order in the following procedure.

Algorithm 4.4 (Instationary ADI method).

```
$i = 1$
$z$ start vector, $\rho_i$ iteration parameter
$r \leftarrow b - Pz$
residual_error $\leftarrow \sqrt{(r, r)}$
while (residual_error $>$ error_wanted)  {
        $z \leftarrow (V + \rho_i I)^{-1}(b - (H - \rho_i I)z)$
        $z \leftarrow (H + \rho_i I)^{-1}(b - (V - \rho_i I)z)$
        $r \leftarrow b - Pz$
        residual_error $\leftarrow \sqrt{(r, r)}$
        $i \leftarrow i + 1$
        if ($i = m + 1$) {
              $i = 1$
        }  }
```

Remark 4.5. *The instationary ADI method in Algorithm* 4.4 *converges provided that H, V are positive definite and that they commute, i.e.,*

$$HV = VH; \tag{21}$$

see [6]. *For the model problem from Section* 3, *where the given data is assembled around the boundary, this is, indeed, the case for the corresponding H and V. When data is given inside the domain to approximate, e.g. discontinuities across faults, the respective operators H and V do not satisfy the commutativity condition any more. However, in the first corresponding tests* [20] *the instationary ADI method still worked well and converged fast. This could be an indication that this method may still be used practically, as long as the amount of data in the interior does not force $HV - VH$ to deviate from the zero matrix in too many places. A corresponding theoretical and computational study will be reported elsewhere.*

In Algorithm 4.4, the inverses $(V + \rho_i I)^{-1}$, $(H + \rho_i I)^{-1}$ are determined exactly by Cholesky decomposition of $V + \rho_i I$ and $H + \rho_i I$ and can be computed ahead of the iteration for each of the iteration parameters. However, for large systems this might be too costly with respect to storage. In this case the alternative would be to recompute these factorizations at each iteration step. As is seen from the examples, the number of iterations turns out to be not much larger than m; hence, this could be a good alternative for large problems. In the examples below we did precompute the factorizations.

The Wachspress parameters seem to be superior to the Peaceman–Rachford parameters in many cases [1, p. 209], and this holds true also for the present problem (cf. Section 5). The rate of convergence in this case is average for fixed m:

$$R = -\frac{2}{m}\log\left(\frac{1 - \sqrt[2(m-1)]{\mathbf{c}}}{1 + \sqrt[2(m-1)]{\mathbf{c}}}\right)^2.$$

Recalling (17) gives $\mathbf{c} \sim h^4$ for the fourth-order problem we are concerned with, so that $m \geq 3$ yields already an improved rate of convergence, compared to the stationary ADI method. In view of (18) which roughly corresponds to $2^{-2m} \leq h^4$ in this case, a grid spacing $h \sim 2^{-j}$ yields $m \geq 2j$ so that m becomes bigger, the finer the mesh size is. For large m, one has $\sqrt[2(m-1)]{\mathbf{c}} \simeq \sqrt{2} - 1$, so that

$$R \simeq -\frac{2}{m}\log(3 - 2\sqrt{2}) \simeq \frac{3.5}{m}.$$

This means that the convergence rate, on average, decreases when m grows. However, as is confirmed by Table 3, when the number of grid points in each coordinate direction is doubled, m is only increased by one and, accordingly, the number of iterations grows very slowly.

Remark 4.6. *In recent years, there have been more investigations on the construction of optimal ADI-parameters for very general* (*but usually second-order*) *problems including nonsymmetric differential operators*; *see, e.g.* [19].

5. COMPUTATIONAL RESULTS

These experiments are performed in the framework of the model problem discussed in Section 3. Initially, we considered several test functions f in (10) sampled on D. The corresponding iteration counts were comparable for all the runs so that the interpolation values on D for the runs corresponding to the tables below are sampled from the quadratic polynomial

$$f(x, y) = 3x^2 + 4y^2 + 9xy + 6x + 8y$$

only. Since f then satisfies the corresponding partial differential equation (6) exactly, we are able to compare the numerical solutions to the analytic solution of the problem.

Define the discrete L^2 norm by

$$\|u\|_h = \left(h^2 \sum_{\alpha \in U} u_\alpha^2 \right)^{1/2}.$$

For all the runs the iterations are stopped when the kth residual $r^k = b - Pz^k$ satisfies

$$\|r^k\|_h \leq 10^{-3}.$$

The number of iterations for a given run is denoted by k_*. We choose $z^0 = 0$ as the start vector for the iterations for all runs.

Before the iteration loop starts for the ADI methods, the maximal and minimal eigenvalues of H and V are determined by using three power iterations. In addition, the Cholesky factorizations for $H + \rho_i I$ and $V + \rho_i I$ are precomputed for each iteration parameter. Since the work for one iteration of the PCG method and one iteration of the ADI method is not directly comparable, we have estimated the total work involved for each of the methods. This work is designated in the number of N-flops, where one N-flop is the amount of work required to perform one inner product.

In Table 1 we have compared the number of iterations and the number of N-flops for all methods. Good relaxation parameters for the preconditioners in the PCG method are determined experimentally. The values chosen are 0.955 for the RILU preconditioner and 1.9 for the SSOR preconditioner. Note that the number k_* of iterations for the PCG methods, as well as for the stationary ADI method, grows very fast as a function of the number N of variables. However, for the instationary ADI method, where we have used Wachspress iteration parameters, the number of iterations is low and grows at a very low rate. As can be seen, the work load counted in N-flops favors the instationary ADI method already

TABLE 1

The Number of Iterations (k_*) and the Number of Inner Product Equivalents (N-Flops) Needed to Fulfill the Stopping Criterion $\|r^k\|_h \leq 10^{-3}$

	PCG RILU		PCG SSOR		Stat. ADI		Instat. ADI	
N	k_*	N-flops	k_*	N-flops	k_*	N-flops	k_*	N-flops
100	16	384	20	480	57	2,127	10	536
400	36	864	37	888	183	6,537	13	659
1,600	93	2,232	77	1,848	*	*	15	765
6,400	302	7,248	179	4,296	*	*	18	888

TABLE 2
The Error $\|f - z^{k_*}\|_h$ between the Analytic Solution f and the Final Iterate z^{k_*}

N	PCG RILU	PCG SSOR	Stat. ADI	Instat. ADI
100	2.7×10^{-4}	4.0×10^{-4}	5.0×10^{-4}	3.7×10^{-4}
400	1.3×10^{-3}	2.6×10^{-3}	7.4×10^{-4}	6.6×10^{-4}
1,600	1.1×10^{-2}	1.3×10^{-2}	*	5.1×10^{-4}
6,400	1.4×10^{-1}	9.6×10^{-2}	*	1.5×10^{-3}

at $N = 400$, i.e., for a 20×20 grid. The "*" in the column for the stationary ADI method means that this method would need more than 400 iterations to reach the tolerance $\|r^k\|_h \leq 10^{-3}$.

Table 2 shows the error $\|f - z^{k_*}\|_h$ between the analytic solution f and the final iterate z^{k_*} for the runs in Table 1. We observe that the error is essentially bounded for the instationary ADI method, while it increases for the PCG methods. Thus, even if $\|r^k\|_h \leq 10^{-3}$ for all methods, the solution z^{k_*} looses accuracy with decreasing h for the PCG methods, while the accuracy is independent of h for the instationary ADI method. This property of the instationary ADI method is confirmed in Table 3. The reason for the growing error in the PCG solution is that the preconditioned iteration matrix $A = CP$ is "ill-conditioned" due to the type of data used. Therefore, the error $\|z^* - z^{k_*}\|_h$, $z^* = P^{-1}b$, need not be small even if $\|r^{k_*}\|_h = \|P(z^* - z^{k_*})\|_h$ is small. This effect is not observed for the instationary ADI method.

In Table 3 we have shown the results of running the instationary ADI method on large grids, using both the Peaceman–Rachford and the Wachspress parameters. The number of iterations is low and essentially independent of the grid size for both parameter types. Also, the accuracy in the solution remains bounded as the grid size increases. Concluding our experiments, it is fair to say that these computational results are very good for a fourth-order problem.

TABLE 3
Results of Running the Instationary ADI Method on Large Grids Using the Two Types of Iteration Parameters

	Peaceman–Rachford				Wachspress		
N	m	k_*	$\|f - z^{k_*}\|_h$	N-flops	k_*	$\|f - z^{k_*}\|_h$	N-flops
10,000	9	36	8.1×10^{-4}	1,536	17	5.1×10^{-4}	871
40,000	10	40	1.7×10^{-3}	1,694	21	1.1×10^{-3}	1,029
90,000	11	44	1.5×10^{-3}	1,852	20	1.5×10^{-3}	1,012
160,000	12	46	9.3×10^{-3}	1,940	22	9.6×10^{-4}	1,100
250,000	13	40	7.2×10^{-3}	1,748	23	3.0×10^{-3}	1,153

Note. The length of the cycles is denoted by m (number of iteration parameters in Algorithm 4.3). The number of iterations needed to fulfill the stopping criterion $\|r^k\|_h \leq 10^{-3}$ is denoted by k_*, and the number of inner product equivalents is denoted by N-flops. In addition the table shows the error $\|f - z^{k_*}\|_h$ between the analytic solution f and the final iterate z^{k_*}.

6. CONCLUDING REMARKS

This paper discusses the application of preconditioned conjugate gradient methods as well as a stationary and an instationary ADI method to linear algebraic equations stemming from a finite difference discretization of the fourth-order elliptic partial differential equation $u_{xxxx} + u_{yyyy} = b$. This equation is solved as part of a scattered data approximation scheme, where it is used as a smoothing technique to fill in unknown values in a partially determined grid. The particular form, not involving mixed derivatives, is used to make it easy to handle discontinuity conditions connected to the scattered data problem.

The results illustrate that, for the problem considered here, the instationary ADI method is superior to the other methods, both with respect to the work involved and with respect to accuracy of the solution. For the instationary ADI method the number of iterations is essentially independent of the grid size, a property not shared by the other solvers where the number of iterations grows very fast as a function of the grid size. One needs to perform more numerical experiments to confirm this for a wider class of model problems.

We remark that the use of the ADI methods is suggested by the fact that the stencil (9) does not contain mixed derivatives. If such terms are included to enforce smoothness of the surface also in other than the coordinate directions, the grid points could not be ordered any more in such a way that the matrices H and V are banded without many zeroes in between. One alternative to compute the Cholesky decomposition of H or V would then be to use an iterative method like the conjugate gradient method instead for the updates of z within an ADI method. Another approach would be to split P into several components, e.g. by introducing ordering also along diagonals in the grid. Or, one could use one iteration of the ADI method with P given by (9) as a preconditioner for the PCG method applied to the operator with mixed derivatives.

In general, it is fair to say that the fast solution of large fourth-order problems is not an easy question. To avoid the requirement on the high regularity of the solution when discretizing the problem (6) with a difference method, one usually rather employs a variational approach. Thus, an alternative would be to use finite elements in a Galerkin method for (4) combined with some approach to incorporate the known grid function values into the problem formulation. Because of the dramatic effect of the condition number for fourth order problems, a multilevel (see [9]) or multigrid preconditioner (see, e.g., [12]) which exists for these types of variational problems yielding a $O(1)$ growth-rate in a PCG method is worth considering. The given data could be handled by appending them as side conditions by Lagrange multipliers as in [15] or by using a least-squares approach like in [7]. However, in addition to the fact that these approaches are typically realized for second order problems only, the construction of several grids of different grid spacing h with a consistent treatment of the given data which a multilevel method could be based upon is not that straightforward.

ACKNOWLEDGMENTS

The authors thank Professor James H. Bramble for pointing out the ADI method for the present application and Professor Wolfgang Dahmen for valuable discussions. Also, we thank the unknown referees for their suggestions.

REFERENCES

1. W. F. Ames, *Numerical Methods for Partial Differential Equations,* 3rd ed. (Academic Press, San Diego, 1992).

2. E. Arge, M. Dæhlen, and A. Tveito, Box spline interpolation; a computational study, *J. Comput. Appl. Math.* **44**, 303 (1992).
3. E. Arge, M. Dæhlen, and A. Tveito, Approximation of scattered data using smooth grid functions, *J. Comput. Appl. Math.* **59**, 191 (1995).
4. E. Arge and M. Floater, Approximating scattered data with discontinuities, *Numer. Algorithms* **8**, 149 (1994).
5. E. Arge and Ø. Hjelle, Software tools for modelling scattered data, in *Numerical Methods and Software Tools in Industrial Mathematics*, edited by M. Dæhlen and A. Tveito (Birkhäuser, Boston, 1997), p. 45.
6. G. Birkhoff, R. S. Varga, and D. Young, Alternating direction implicit methods, *Adv. Comput.* **3**, 189 (1962).
7. J. Bramble, R. Lazarov, and J. Pasciak, Least-squares for second order elliptic problems, Technical Report ISC-97-05-MATH, Texas A&M University, January 1997.
8. A. M. Bruaset, *Preconditioners for Discretized Elliptic Problems*, Ph.D. thesis, Dept. of Informatics, University of Oslo, Norway, December 1992.
9. W. Dahmen and A. Kunoth, Multilevel preconditioning, *Numer. Math*. **63**, 315 (1992).
10. J. Duchon, Splines minimizing rotation-invariant semi-norms in Sobolev spaces, in *Constructive Theory of Functions of Several Variables*, edited by W. Schempp and K. Zeller (Springer-Verlag, New York/Berlin, 1977), p. 85. [Lecture Notes in Math., Vol. 571]
11. G. Golub and C. F. Van Loan, *Matrix Computations* (John Hopkins Univ. Press, Baltimore, MD, 1989).
12. W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations* (Springer, 1994).
13. Ø. Hammer, *Extensions of the Minimum Curvature Method for Geological Horizon Modelling*, Master's thesis, Dept. of Informatics, University of Oslo, Norway, April 1996.
14. O. Kloster, *En trestegs-metode for approksimasjon av spredte data*, Master's thesis, Faculty for Physics and Mathematics, The Norwegian Technical University (NTH), Norway, 1995.
15. A. Kunoth, Multilevel preconditioning—Appending boundary conditions by Lagrange multipliers, *Adv. Comput. Math.* **4**, 145 (1995).
16. M. J. D. Powell, The theory of radial basis function approximation in 1990, in *Advances in Numerical Analysis, Vol. II*, edited by W. Light (Oxford Science Publications, 1992), p. 105.
17. D. Shepard, A two-dimensional interpolating function for irregularly spaced data, *Proc. ACM. Nat. Conf.* 517 (1968).
18. *SISCAT Scattered Data Library*, `http://www.nobjects.com.`
19. G. Starke, Optimal alternating direction implicit parameters for nonsymmetric systems of linear equations, *SIAM J. Numer. Anal.* **28**, 1431 (1991).
20. Ch. Tarrou, private communication.