DOI: 10.1051/m2an/2010017

ESAIM: M2AN 44 (2010) 737-758

www.esaim-m2an.org

# TWO NUMERICAL METHODS FOR THE ELLIPTIC MONGE-AMPÈRE EQUATION

Jean-David Benamou<sup>1</sup>, Brittany D. Froese<sup>2</sup> and Adam M. Oberman<sup>2</sup>

Abstract. The numerical solution of the elliptic Monge-Ampère Partial Differential Equation has been a subject of increasing interest recently [Glowinski, in 6th International Congress on Industrial and Applied Mathematics, ICIAM 07, Invited Lectures (2009) 155-192; Oliker and Prussner, Numer. Math. 54 (1988) 271–293; Oberman, Discrete Contin. Dyn. Syst. Ser. B 10 (2008) 221–238; Dean and Glowinski, in Partial differential equations, Comput. Methods Appl. Sci. 16 (2008) 43–63; Glowinski et al., Japan J. Indust. Appl. Math. 25 (2008) 1-63; Dean and Glowinski, Electron. Trans. Numer. Anal. 22 (2006) 71–96; Dean and Glowinski, Comput. Methods Appl. Mech. Engrg. 195 (2006) 1344– 1386; Dean et al., in Control and boundary analysis, Lect. Notes Pure Appl. Math. 240 (2005) 1–27; Feng and Neilan, SIAM J. Numer. Anal. 47 (2009) 1226–1250; Feng and Neilan, J. Sci. Comput. 38 (2009) 74-98; Feng and Neilan, http://arxiv.org/abs/0712.1240v1; G. Loeper and F. Rapetti, C. R. Math. Acad. Sci. Paris 340 (2005) 319-324]. There are already two methods available [Oliker and Prussner, Numer. Math. 54 (1988) 271–293; Oberman, Discrete Contin. Dyn. Syst. Ser. B 10 (2008) 221-238] which converge even for singular solutions. However, many of the newly proposed methods lack numerical evidence of convergence on singular solutions, or are known to break down in this case. In this article we present and study the performance of two methods. The first method, which is simply the natural finite difference discretization of the equation, is demonstrated to be the best performing method (in terms of convergence and solution time) currently available for generic (possibly singular) problems, in particular when the right hand side touches zero. The second method, which involves the iterative solution of a Poisson equation involving the Hessian of the solution, is demonstrated to be the best performing (in terms of solution time) when the solution is regular, which occurs when the right hand side is strictly positive.

Received September 16, 2008. Revised June 11, 2009 and November 12, 2009. Published online February 23, 2010.

Keywords and phrases. Finite difference schemes, partial differential equations, viscosity solutions, Monge-Ampère equation.

 $<sup>^1</sup>$  INRIA, Domaine de Voluceau, B.P. 105, 78153 Rocquencourt, France.  ${\tt jean-david.benamou@inria.fr}$ 

 $<sup>^2</sup>$  Department of Mathematics, Simon Fraser University, 8888 University Drive, BC V5A 1S6 Burnaby, Canada. bdf1@sfu.ca; aoberman@sfu.ca

## 1. Introduction

The Monge-Ampère equation is a geometric fully nonlinear elliptic Partial Differential Equation (PDE) [17]. Applications of the Monge-Ampère equation appear in the classical problem of prescribed Gauss curvature and in the problem of optimal mass transportation (with quadratic cost), among others [4,11].

The numerical solution of the elliptic Monge-Ampère PDE has been a subject of increasing interest [7–10,12–14,16,18,21,22], including an invited lecture at 2007 ICIAM [15].

Two cases must be distinguished: when the right hand side is strictly positive (and bounded) the operator is uniformly elliptic and solutions are regular (assuming the domain is strictly convex, see below). When the right hand side touches zero, the operator is degenerate elliptic and solutions can be singular. Naturally, it is more challenging to solve the equation in the degenerate elliptic case.

Many of the newly proposed methods lack numerical evidence of convergence on singular solutions, or are known to break down in this case.

While one of the authors has previously introduced a convergent finite difference method [21], the increased attention by the numerical PDE community has prompted us to write this paper. This work has two main goals:

- (1) To present a simple (nine point stencil) finite difference method which performs just as well as the provably convergent (wide stencil) method of [21] in robust numerical experiments.
- (2) To present a simple method which models (and outperforms) the other methods in the regular/uniformly elliptic case, and demonstrates the dramatically decreasing performance on singular solutions.

The PDE we study is

$$\det(D^2u(x)) = f(x),$$

where  $D^2u$  is the Hessian of the function u. Since we will be restricting to domains  $\Omega \subset \mathbb{R}^2$ , we can rewrite the PDE as

$$\left(\frac{\partial^2 u}{\partial x^2} \frac{\partial^2 u}{\partial y^2} - \left(\frac{\partial^2 u}{\partial x \partial y}\right)^2\right)(x, y) = f(x, y) \text{ in } \Omega \subset \mathbb{R}^2.$$
(MA)

The PDE comes with Dirichlet boundary conditions

$$u = g, \quad \text{on } \partial\Omega$$
 (D)

and the additional convexity constraint

$$u$$
 is convex,  $(C)$ 

which is required for the equation to be elliptic. Here  $\Omega \subset \mathbb{R}^n$  is a bounded domain with boundary  $\partial\Omega$  and the source term  $f:\Omega \to \mathbb{R}$  is a non-negative function (or, later, measure). The PDE (MA) (along with the conditions (D, C)) is challenging to solve numerically for the following reasons:

- The equation is fully nonlinear, so the correct notion of weak solutions is either geometric solutions [1] or viscosity solutions [6]. While the two-dimensional equation can be written in divergence form (see below), it still involves the Hessian of the solution. This restricts the applicability of the Finite Element Method (FEM) or general Galerkin projection methods to more regular solutions.
- The weak solutions of the equation can be quite singular. Specifically, if the source function f is not strictly positive, then solutions need not be in  $\mathbf{H}^2(\Omega)$  (see, for example, Sect. 5.4). When f is a Dirac measure, solutions can be only Lipschitz continuous (see Sect. 6.1).
- The convexity constraint: The solution u must be convex in order for the equation to be elliptic. Without the convexity constraint, this equation does not have a unique solution. (For example, taking boundary data g = 0, if u is a solution then -u is also a solution, in  $\mathbb{R}^2$ .)

The equation can be written in divergence form in two dimensions:

$$\det(D^2 u) = \frac{1}{2} \operatorname{div} \left( \begin{pmatrix} u_{yy} & -u_{xy} \\ -u_{xy} & u_{xx} \end{pmatrix} (u_x, u_y)^T \right).$$

## 1.1. Regularity

Regularity results can be used to determine when solutions are smooth, and thereby determine the method of choice. For strictly convex domains and smooth, strictly positive data, the solution is smooth [5]. However, as soon as f approaches zero, regularity breaks down; see [17] for  $W^{2,p}$  estimates on solutions.

Since we work in the rectangle, which is not strictly convex, problems can occur at the boundary (see Sect. 5.3). While smooth, strictly convex boundary data can improve this, an additional condition is needed to avoid singularities at the corner.

## 1.2. Related work: provably convergent methods

Two provably convergent methods have already been presented for (MA).

The first, by Oliker and Prussner [22], uses a discretization based on the geometric interpretation of the solutions [1]. In two dimensions this method converges to the geometric solution.

The second, by one of authors of this work [21], converges to the viscosity solution of (MA). While it is not difficult to build consistent, stable numerical methods for (MA), this alone is not enough to prove convergence to the viscosity solution. For convergence, another ingredient is needed, namely monotonicity of the methods [2]. Monotone methods respect the comparison principle at the discrete level [19]. However, building monotone methods for this equation requires the use of wide stencils, which come with an additional consistency error related to the directional resolution of the stencil.

## 1.3. Related work: other methods

There have been a number of other recently proposed methods.

Dean and Glowinski et al. [7–10,15,16] have investigated Lagrangian and least squares methods for the numerical solution of Monge-Ampère. The authors are careful to be clear about the convergence behavior of their methods. In [8] they give an example of a solution that is not in  $\mathbf{H}^2(\Omega)$ , for which their method diverges. They point out in [7] "if, despite the smoothness of its data, the above problem has no solution in  $\mathbf{H}^2(\Omega)$ , the above augmented Lagrangian method solves it in a least-squares sense," and in [8], "in cases where the Monge-Ampère equation under consideration has no smooth solution, [the methods] provide a quasi-solution".

Böhmer [3] performs consistency and stability analysis of finite element methods for fully nonlinear equations. However, as mentioned before, consistency and stability is not enough to prove convergence to viscosity solutions.

Loeper and Rapetti [18] solve the equation (with periodic boundary conditions) by linearization and iterate using Newton's method. They prove convergence of the Newton algorithm for the linearization of the continuous problem to the solution of (MA). However, they do not address the issue of the convergence of the discretized solution to the solution of the equation in the limit of the discretization parameter going to zero.

Feng and Neilan [12–14] solve second order equations (including the Monge-Ampère equation) by adding a small multiple of the bilaplacian. The bilaplacian term introduces an additional discretization error and additional boundary conditions, which may not be compatible with the solution. A proof of convergence to viscosity solutions has been cited, but is not yet available. In [14] they compute an example where f is sometimes negative, which they describe to be convex.

## 1.4. Discussion of contents

We present in Section 2 two different numerical methods for solving (MA). These methods have the advantages of *simplicity* and *performance*. The methods are simple in the sense that the algorithms are fully explained in about a page, and can be implemented in MATLAB using only basic explicit matrix operations (M1,M2) and a linear solve (M2).

The overall and relative performance of the methods is discussed in Section 3. This performance of (M2) relative to (M1) is superior when the solutions are smooth but substantially inferior when the solutions

are singular. In fact, (M2) appears to diverge on singular solutions. We study in detail the performance of the two methods on solutions ranging from smooth Section 4 to moderate Section 5 to very singular Section 6.

Extensions are discussed in Section 7. Conclusions are made in Section 8.

### 2. The methods

In this section we present the two methods for solving the two-dimensional Monge-Ampère equation. The first is an explicit, Gauss-Seidel iteration method that is obtained by solving a quadratic equation at each grid point and choosing the smaller root to ensure selection of the convex solution.

The second method is an iterative method that requires the solution of a Poisson equation at each iteration. The source term in the Poisson equation involves the source function for (MA), f, and the Hessian of the current iterate.

## 2.1. Method 1: An explicit finite difference method

The first method involves simply discretizing the second derivatives in (MA) using standard central differences on a uniform Cartesian grid. The result is

$$(\mathcal{D}_{xx}^2 u_{ij})(\mathcal{D}_{yy}^2 u_{ij}) - (\mathcal{D}_{xy}^2 u_{ij})^2 = f_{i,j}$$
(MA)<sup>h</sup>

where

$$\mathcal{D}_{xx}^{2}u_{ij} = \frac{1}{h^{2}} (u_{i+1,j} + u_{i-1,j} - 2u_{i,j})$$

$$\mathcal{D}_{yy}^{2}u_{ij} = \frac{1}{h^{2}} (u_{i,j+1} + u_{i,j-1} - 2u_{i,j})$$

$$\mathcal{D}_{xy}^{2}u_{ij} = \frac{1}{4h^{2}} (u_{i+1,j+1} + u_{i-1,j-1} - u_{i-1,j+1} - u_{i+1,j-1}).$$

Since we are using centered differences, this discretization is consistent with (MA), and it is second order accurate if the solution is smooth.

Remark 2.1. The above discretization is valid in the interior of the domain. Of course, in a general (non-rectangular) domain it may be necessary to interpolate the boundary values to obtain a first order scheme at the boundary.

Next, introduce the notation

$$a_{1} = (u_{i+1,j} + u_{i-1,j})/2$$

$$a_{2} = (u_{i,j+1} + u_{i,j-1})/2$$

$$a_{3} = (u_{i+1,j+1} + u_{i-1,j-1})/2$$

$$a_{4} = (u_{i-1,j+1} + u_{i+1,j-1})/2$$
(2.1)

and rewrite  $(MA)^h$  as a quadratic equation for  $u_{i,j}$ 

$$4(a_1 - u_{i,j})(a_2 - u_{i,j}) - \frac{1}{4}(a_3 - a_4)^2 = h^4 f_{i,j}.$$

Solving for  $u_{i,j}$  and selecting the smaller root (in order to select the locally convex solution), we obtain Method 1:

$$u_{i,j} = \frac{1}{2}(a_1 + a_2) - \frac{1}{2}\sqrt{(a_1 - a_2)^2 + \frac{1}{4}(a_3 - a_4)^2 + h^4 f_{i,j}}.$$
 (M1)

We can now use Gauss-Seidel iteration to find the fixed point of (M1). The Dirichlet boundary conditions (D) are enforced at boundary grid points. The convexity constraint (C) is not enforced beyond the selection of the positive root in (M1).

## 2.2. Method 2: Iterative solution of a Poisson equation

The second method we study involves successively solving the Poisson equation until a fixed point is reached. This method is a model for the non-convergent methods which require that the solution be in  $\mathbf{H}^2(\Omega)$ .

The method is based on an algebraic identity satisfied by the two-dimensional Monge-Ampère equation, used by Dean and Glowinski [7–10,16]. We generalize this idea below to produce a fixed point operator in higher dimensions.

Expanding  $(\Delta u)^2 = (u_{xx} + u_{yy})^2$  and applying (MA), we obtain the identity

$$(u_{xx} + u_{yy})^2 = u_{xx}^2 + u_{yy}^2 + 2u_{xy}^2 + 2f$$

for solutions u(x, y) of (MA) which are smooth enough.

This algebraic identity leads to the following definition, where the positive square root is selected in order to select the locally convex solution.

**Definition 2.1.** Define the operator  $T: \mathbf{H}^2(\Omega) \to \mathbf{H}^2(\Omega)$  for  $\Omega \subset \mathbb{R}^n$  by

$$T[u] = \Delta^{-1} \left( \sqrt{(\Delta u)^2 + 2(f - \det(D^2 u))} \right).$$

In particular, in  $\mathbb{R}^2$ ,

$$T[u] = \Delta^{-1} \left( \sqrt{u_{xx}^2 + u_{yy}^2 + 2u_{xy}^2 + 2f} \right). \tag{2.2}$$

The solution operator incorporates the Dirichlet boundary conditions (D).

**Remark.** While it is possible to define the domain of the operator more carefully, for our purposes it is sufficient to observe that it is well-posed on  $\mathbf{H}^2(\Omega)$ .

**Lemma 2.1.** Provided the solution, u, of (MA) is in  $\mathbf{H}^2(\Omega)$ , it is a fixed point of the operator T.

*Proof.* Let v be the solution of (MA, D, C). Since v is in  $H^2(\Omega)$ ,  $\Delta v$  is in  $L^2(\Omega)$  and the Poisson equation

$$\Delta u = |\Delta v|$$
,

augmented by the Dirichlet boundary conditions (D), is well-posed. Inserting the solution v into Definition 2.1 we obtain

$$T[v] = \Delta^{-1}\left(\sqrt{(\Delta v)^2}\right) = \Delta^{-1}(|\Delta v|).$$

Since v is convex, by (C),  $\Delta v > 0$ , and consequently,

$$T[v] = \Delta^{-1}(\Delta v) = v.$$

Therefore, v is a fixed point of (M2).

Method 2 consists of iterating  $u^{n+1} = T[u^n]$  by solving

$$\Delta u^{n+1} = \sqrt{(u_{xx}^n)^2 + (u_{yy}^n)^2 + 2(u_{xy}^n)^2 + 2f}$$
(M2)

along with the prescribed Dirichlet boundary conditions (D).

#### 2.3. Discussion of Method 1

In order to prove convergence of the finite difference method to viscosity solution of (MA), it is sufficient to prove that the method is monotone. When written in the form (M1), it would be enough to show [19] that the value  $u_{i,j}$  is a non-decreasing function of the neighboring values. However, this is not the case for (M1). In addition, based on the discussion in [20,21], we don't expect that any narrow stencil method will be monotone and consistent for the equation (MA).

However, we can study the convexity requirement. Directional convexity along grid directions is the requirement that the solution satisfy

$$u(x) \le \frac{u(x+v) + u(x-v)}{2}$$
 (2.3)

for all lattice (grid) directions v. For more details about enforcing convexity of solutions, using finite difference methods, we refer to [20].

The directional convexity along the axes is satisfied by (M1).

**Lemma 2.2.** The fixed point of (M1) satisfies the inequalities (2.3) for the grid directions

$$v \in \{(1,0), (0,1)\}.$$

*Proof.* Using the notation of (2.1), assume without loss of generality that  $a_2 \leq a_1$ . Next, estimate  $u_{i,j}$  from (M1), using the fact that f is non-negative,

$$u_{i,j} \le \frac{a_1 + a_2}{2} - \frac{a_1 - a_2}{2} = a_2 = \frac{u_{i,j+1} + u_{i,j-1}}{2}$$

to obtain convexity in the direction v = (0, 1). Using the first assumption in the proof, we also have

$$u_{i,j} \le \frac{u_{i+1,j} + u_{i-1,j}}{2},$$

which gives the convexity in the direction v = (1, 0).

## 2.4. Discussion of Method 2

The method is convenient to implement because it simply involves evaluating derivatives and solving the Poisson equation, which can be accomplished using the method of choice. However, the Poisson equation is solved with a source involving the norm of the Hessian of the previous solution. When the solution is not in  $\mathbf{H}^2(\Omega)$ , this function can be unbounded.

## 3. Numerical examples: Overview

In this section we discuss the performance of the methods on a range of computational examples. We are concerned with two measures of performance.

- (1) Convergence rate. Both methods are discretized using standard centered finite differences, which is formally second order accurate. For a fixed resolution, the numerical solution, and consequently the numerical accuracy, of both methods is the same. In practice, the observed convergence rate varies depending on the regularity of the solution, from second order (for smooth solutions) to order one half (for singular solutions).
- (2) Solution time. The difference in performance for the two methods is observed in the number of iterations and the CPU time required to solve the discretized equation.

The performance of (M2) relative to (M1) is faster when the solutions are smooth but substantially slower when the solutions are singular. In fact, we have found strongly singular solutions for which (M2) appears to diverge.

These examples fall into three categories depending on the regularity of the solution.

- Smooth or mildly singular solutions. These solutions are the mildest: we expect any reasonable methods for Monge-Ampère will perform well.
- Moderately singular solutions. These examples that are at the edge of the capabilities of many of the existing methods.
- Strongly singular solutions. In this case many of the methods currently available have do not present numerical or analytical evidence of convergence. In particular, in the case of [7–10,16], the methods are known to fail. In the case of [12–14] we are not aware of computational results.

## 3.1. Comparison with previous works

There are four main examples computed by Dean and Glowinski. Feng and Neilan also give computational examples for these problems. The first two, which we call mild, are studied in Sections 4.1 and 4.2. The third, for which no exact solution is available, though the solution is known to be singular, is studied in Section 5.3. The fourth solution is not in  $\mathbf{H}^2(\Omega)$ . For this solution, the method of Dean and Glowinski diverges [8], p. 1361. While the convergence rate is slower than for the smoother examples, both our methods (M1) and (M2) are observed to converge for this example, Section 5.4.

The solutions above we categorize as mild or moderate. In addition, we present computational results for more singular examples, Section 6.

We compare the results of Dean and Glowinski [7-10,16] to (M2), which is the model for these methods. For the first three examples, these require roughly the same number of iterations to converge. Although computation times were not given, presumably each iteration of our method is more efficient since it involves only a single Poisson solve. Each iteration of [7-10,16] involves a Poisson solve and additional calculations.

Feng and Neilan [12–14] do not give details about the computation time or iteration count of their numerical method.

We also compared (M1), (M2) to the monotone method of [21]. The discretizations are different and so the accuracy can differ. As expected, the accuracy for the monotone method was generally lower, but improved as the stencil was widened. The only case where the monotone method was more accurate than (M1) was for the most singular example computed, Section 6.1. A minor modification (implemented in Sect. 7.1) improves the accuracy of the method by an order of magnitude while maintaining a narrow stencil.

## 3.2. Comparison of Method 1 and Method 2

The computation time for (M1) was roughly independent of the regularity of the solution. On the other hand, the computation times for (M2) were faster (by more than one order of magnitude) on very smooth solutions, about equal on mildly singular solutions, and much slower (by a couple of orders of magnitude) on the more singular solutions.

Naturally, for any finite resolution, the discretized solutions are not singular, so we cannot conclude that the method diverges. But since the relative computational costs increase dramatically as the solutions become less regular, this behavior is consistent with divergence.

We find (M2) performs extremely rapidly when solutions are smooth, but is very slow when solutions are not smooth or not strictly convex. The explicit method (M1), on the other hand, requires approximately the same amount of CPU time to converge without regard for the regularity of the solutions.

To show this more concretely, we consider a representative example from each category: the smooth radial solution of Section 4.1, the example with noisy data of Section 5.1, and the non-smooth cone of Section 6.1. We compare the absolute computation time required for these three examples to demonstrate our observation that the speed of (M1) is independent of regularity while the speed of (M2) is very sensitive to regularity; see Figure 1. We compare the ratio of CPU times required by the two methods for a representative example, see Table 1.

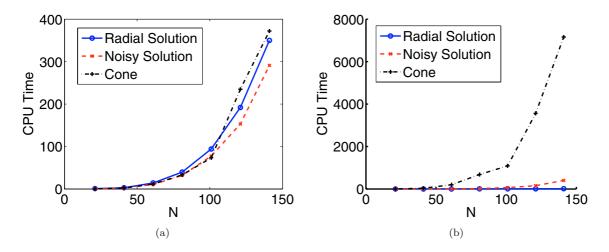


FIGURE 1. CPU time required to converge for the smooth radial solution of Section 4.1, the noisy solution of Section 5.1, and the non-smooth cone of Section 6.1. (a) Results for (M1). (b) Results for (M2).

TABLE 1. Ratio of computation times required for (M1) and (M2) for three representative examples: the smooth radial solution of Section 4.1, the noisy solution of Section 5.1, and the non-smooth cone of Section 6.1.

$\overline{N}$	Section 4.1	Section 5.1	Section 6.1
21	6.3	1.3	0.12
41	4.1	2.1	0.07
61	7.7	1.9	0.06
81	11.7	1.5	0.05
101	14.1	1.2	0.04
121	23.5	1.0	0.07
141	29.9	0.7	0.05

# 3.3. Computational details

The solutions are computed on an  $N \times N$  grid and iteration is continued until the maximum difference between two subsequent iterates is less than  $10^{-14}$ . By convention, h is the spatial discretization parameter h = L/N where L is the length of one side of the square domain.

A reasonable choice for the initial data is the solution of  $u_{xx} + u_{yy} = \sqrt{2f}$ , which was used when measuring the solution time and iteration count. However, the methods are robust: the iterations converge to a fixed point for all initial data attempted. The initial data attempted included: a constant (which didn't respect the boundary conditions), and both non-convex and random initial data.

The implementation of (M2) can be performed using the method of choice. We focus on the simple finite difference method that arises from discretizing (2.2) using central differences (as with the first method). We also briefly discuss in Section 7.2 the use of an adaptive finite element method.

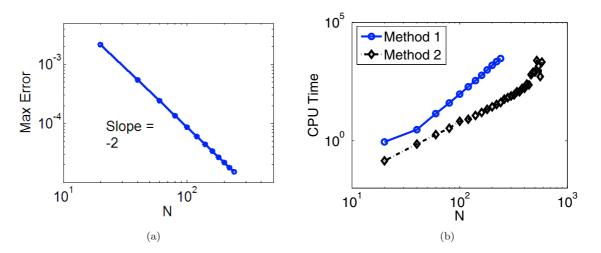


FIGURE 2. Convergence results for  $u(x,y) = \exp\left(\frac{x^2 + y^2}{2}\right)$  on an  $N \times N$  grid. (a) Error versus N. (b) Total CPU time.

# 4. Smooth or mildly singular examples

In the following examples, the solutions are smooth, or mildly singular, and (M2) is much faster than (M1). We observed convergence rates consistent with  $O(h^2)$  for the smooth solution, and  $O(h^{1.5})$  for the mildly singular solution.

### 4.1. An exact radial solution

An exact radial solution is

$$u(x,y) = \exp\left(\frac{x^2 + y^2}{2}\right), \qquad f(x,y) = (1 + x^2 + y^2)\exp(x^2 + y^2).$$

Both methods converge to the same numerical solution, which is accurate to second order in h,  $O(h^2)$ , see Figure 2(a). For this smooth example, (M2) is by far the fastest method: see Table 2 and Figure 2(b).

## 4.2. Blow-up at boundary

Another exact solution is

$$u(x,y) = \frac{2\sqrt{2}}{3}(x^2 + y^2)^{3/4}, \qquad f(x,y) = \frac{1}{\sqrt{x^2 + y^2}}$$

on the square  $[0,1] \times [0,1]$ ; the function f blows up at the boundary. Again, both methods converge to the exact solution, although the order of convergence is now only 1.5. For this example (M2) was significantly faster than (M1). See Tables 3 and 4 and Figure 3.

Table 2. Computation times for the exact solution  $u(x,y) = \exp\left(\frac{x^2+y^2}{2}\right)$  on an  $N \times N$  grid.

$\overline{N}$	Iterat	Iterations		ime (s)
	(M1)	(M2)	(M1)	(M2)
21	1014	54	0.9	0.1
81	14621	59	39.8	3.4
141	42746	60	350.0	11.7
201	84611	60	1540.0	27.2
261	_	65	_	57.8
:		:		:
341	_	65	_	118.0
421		78	_	249.0
501		151		804.0
581		284	_	2120.0

Table 3. Errors for the exact solution  $u(x,y) = \frac{2\sqrt{2}}{3}(x^2 + y^2)^{3/4}$  on an  $N \times N$  grid. Results are the same for both methods.

N	21	61	101	141	181	221
$  u-u^{(N)}  _{\infty}$	$5.5 \times 10^{-4}$	$1.1 \times 10^{-4}$	$5.0 \times 10^{-5}$	$3.0 \times 10^{-5}$	$2.1\times10^{-5}$	$1.5 \times 10^{-5}$

Table 4. Computation times for  $u(x,y) = \frac{2\sqrt{2}}{3}(x^2 + y^2)^{3/4}$  on an  $N \times N$  grid.

$\overline{N}$	Iterati	Iterations		me (s)
	(M1)	(M2)	(M1)	(M2)
21	1083	58	0.5	0.2
61	8967	59	28.7	1.9
101	23849	59	206.0	6.9
141	45358	59	770.0	11.4
181	73254	59	2050.0	21.2
221	107388	60	4500.0	34.6

# 5. Moderately singular examples

In the following examples, the solutions are moderately singular, and the two methods perform about equally.

## 5.1. Noisy data

We now return to the first example we considered, with exact solution

$$u(x,y) = \exp\left(\frac{x^2 + y^2}{2}\right), \qquad f(x,y) = (1 + x^2 + y^2)\exp(x^2 + y^2),$$

and study the effects of adding high frequency sinusoidal noise to the data (f and the boundary conditions). Results are shown in Figure 4. Although the data is not convex, all methods find a solution that is convex except at the boundary. The noise does not seem to have any effect on the rate of convergence for (M1). However, (M2) becomes much slower when dealing with the noisy, non-convex data so that the time required for the two methods is comparable; compare Tables 2 and 5.

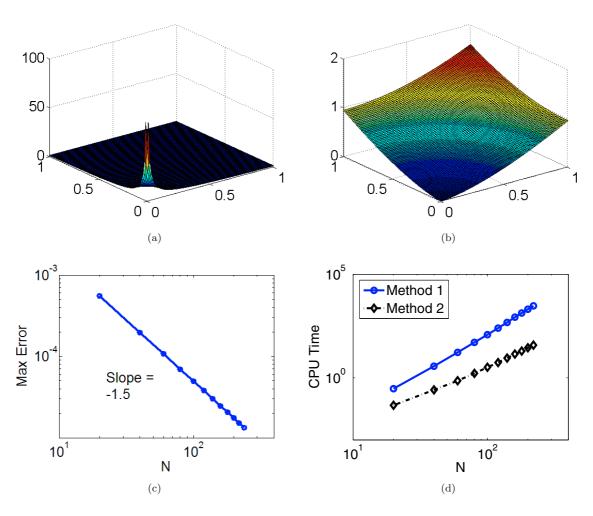


FIGURE 3. Results for  $u(x,y)=\frac{2\sqrt{2}}{3}(x^2+y^2)^{3/4}$  on an  $N\times N$  grid. (a) f blows up at the boundary. (b) Surface plot of the solution. (c) Error versus N. (d) Total CPU time.

Table 5. Computation times for  $u(x,y) = \exp\left(\frac{x^2 + y^2}{2}\right)$  with noisy data on an  $N \times N$  grid.

$\overline{N}$	Iterat	Iterations		ime (s)
	(M1)	(M2)	(M1)	(M2)
21	1016	58	0.2	0.1
41	3866	99	2.4	1.2
61	8432	185	10.7	5.6
81	14672	409	32.0	21.0
101	22516	695	77.6	63.4
121	31965	1205	153.0	156.0
141	43008	2162	291.0	401.0

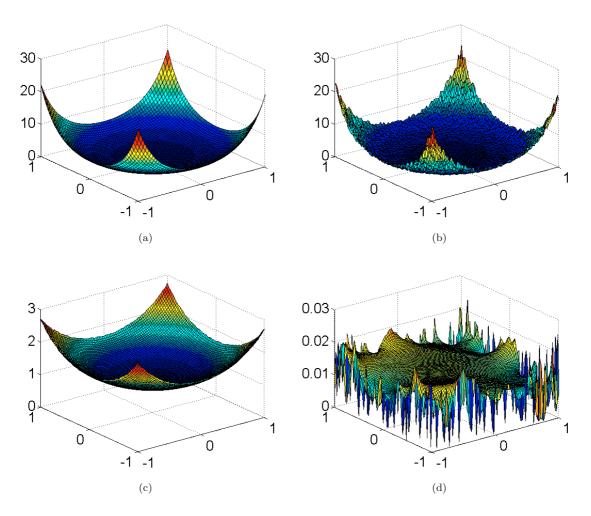


FIGURE 4. Results for  $u(x,y) = \exp\left(\frac{x^2+y^2}{2}\right)$  with noisy, non-convex data on an  $81 \times 81$  grid. (a) f without noise. (b) f with noise. (c) Surface plot of the solution with noise. (d)  $|u-u_{\text{noisy}}|$ .

## 5.2. Discontinuous source term

We next consider an example source term f that is discontinuous on a circle

$$f = \begin{cases} 1 & (x - 0.5)^2 + (y - 0.5)^2 > 0.16^2 \\ 0 & \text{otherwise.} \end{cases}$$

The solution is

$$u = \max\left\{\frac{(x-0.5)^2 + (y-0.5)^2}{2}, 0.08\right\}.$$

The solution is flat where f vanishes and parabolic elsewhere. See the surface plot in Figure 5(a). Both methods require about the same amount of time for this example as is seen in Figure 5(b) and Table 6.

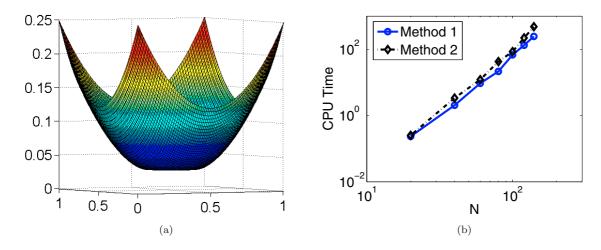


FIGURE 5. Results for Section 5.2. (a) Surface plot of solution. (b) Total CPU time.

Table 6. Computation times for Section 5.2 on an  $N \times N$  grid.

$\overline{N}$	Iterations		CPU time (s)	
	(M1)	(M2)	(M1)	(M2)
21	950	90	0.2	0.3
41	3651	259	2.0	3.4
61	7944	363	9.4	12.1
81	13810	783	21.5	42.5
101	21162	893	68.0	85.0
121	29966	1678	132.0	221.0
141	40196	2076	244.0	480.0

## 5.3. An example which is singular at the boundary

This example was taken from [8]. We consider the solution of Monge-Ampère when f=1 with constant boundary values 1 on the square  $[-1,1] \times [-1,1]$ . There is no exact solution available for this example, however the solution is not smooth: since  $u_{xx}=0$  on part of the boundary,  $u_{yy}\to\infty$  there. Method (M1) converges to a solution that is convex except near the corners of the domain (Fig. 6(b)). This is comparable to the solutions presented in [8]. Method (M2) converges to a slightly different, more convex, solution.

As no exact solution is available for comparison, following [21], we focus on the minimum value of the solution produced by each method. We also consider the minimum values obtained using the monotone method in [21]; see Table 7. The solutions obtained with our methods are lower than the solutions obtained with the monotone, wide-stencil method. The monotone method is known to converge to a supersolution of (MA), so we do expect these values to be higher than the true values. This is evident in the numerics as the solution obtained with the monotone method becomes lower (and closer to the solutions obtained with our methods) as the stencil width is increased.

Method (M2) performs very slowly in this case. It is also much slower than (M1), which requires about the same number of iterations to converge as it did for the smooth examples. See Figure 7 and Table 8.

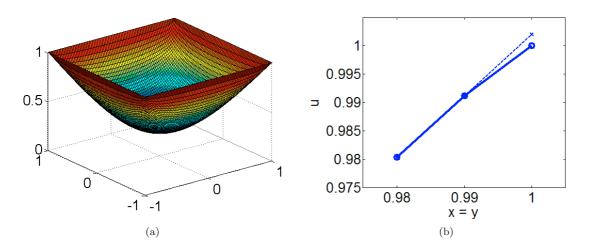


FIGURE 6. Solutions when f = 1 with constant boundary values 1. (a) Surface plot of solution. (b) The methods produces a solution with slight negative curvature along the line y = x.

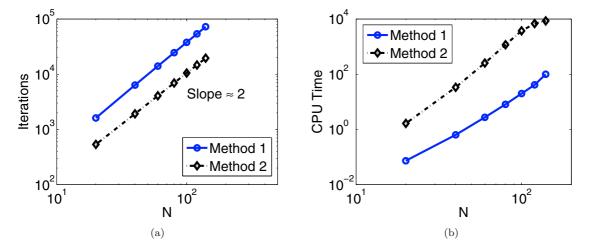


FIGURE 7. Convergence results when f=1 with constant boundary values 1 on an  $N \times N$  grid. (a) Total number of iterations required versus N. (b) Total CPU time versus N.

## 5.4. Surface of ball

Next we consider another example from [8], which is the exact solution

$$u(x,y) = -\sqrt{2 - x^2 - y^2}, \qquad f = \frac{2}{(2 - x^2 - y^2)^2}$$

on the domain  $[0,1] \times [0,1]$ . The method of [8] is known to diverge for this example, however both (M1) and (M2) successfully capture this non-smooth solution, although we no longer have second order accuracy as we did with the smooth solution of Section 4.1, instead the convergence rate is about .5. See Figure 8 and Table 9. In this case (M2) outperforms (M1) by an order of magnitude with regard to computation time.

TABLE 7. Minimum value of u when f = 1 with constant boundary values 1 on an  $N \times N$  grid. We include results from the wide stencil methods of [21] on nine, seventeen, and thirty-three point stencils.

$\overline{N}$		,	$u_{\min}^{(N)}$	
	(M1),	9-Point	17-Point	33-Point
	(M2)	stencil	stencil	stencil
21	0.2892	0.3115	0.2815	0.2839
41	0.2734	0.3090	0.2807	0.2732
61	0.2682	0.3082	0.2803	0.2711
81	0.2655	0.3078	0.2802	0.2704
101	0.2639	0.3076	0.2800	0.2700
121	0.2629	0.3075	0.2798	0.2697
141	0.2621	0.3074	0.2796	0.2695

Table 8. Computation times when f = 1 with constant boundary values 1 on an  $N \times N$  grid.

$\overline{N}$	Itera	tions	CPU 1	time (s)
	(M1)	(M2)	(M1)	(M2)
21	1615	535	0.1	1.7
41	6366	1917	0.6	33.6
61	14049	4070	2.8	254.0
81	24560	6944	8.1	1170.0
101	37824	10503	19.9	3730.0
121	53778	14711	41.2	6820.0
141	72377	19571	99.7	8560.0

Table 9. Computation times when  $u = -\sqrt{2 - x^2 - y^2}$  on an  $N \times N$  grid.

$\overline{N}$	Iterations		CPU time (s)	
	(M1)	(M2)	(M1)	(M2)
21	1009	67	0.2	0.2
41	3833	79	1.9	1.0
61	8343	89	8.5	2.7
81	14480	99	26.3	5.2
101	22189	106	60.2	10.0
121	31445	113	120.1	15.0
141	42202	119	218.3	21.3

## 6. Strongly singular examples

In the following examples, the solutions are strongly singular (they are not in  $\mathbf{H}^2(\Omega)$ ), and (M2) is expected to diverge. The increasingly large number of iterations required to locate a fixed point of (M2) (both relative to (M1) on the same solutions, and relative to (M2) on smoother solutions) is consistent with divergence.

For the cone  $u(x,y) = \sqrt{x^2 + y^2}$ , the convergence rate decreases to  $O(h^{\cdot 8})$ , and the solution time is about one order of magnitude slower for (M2).

Another example is the absolute value function u(x,y) = |x| in Section 6.2. Here the computation time for (M1) is two orders of magnitude faster than for (M2). Since the solution is piecewise linear, the discretization

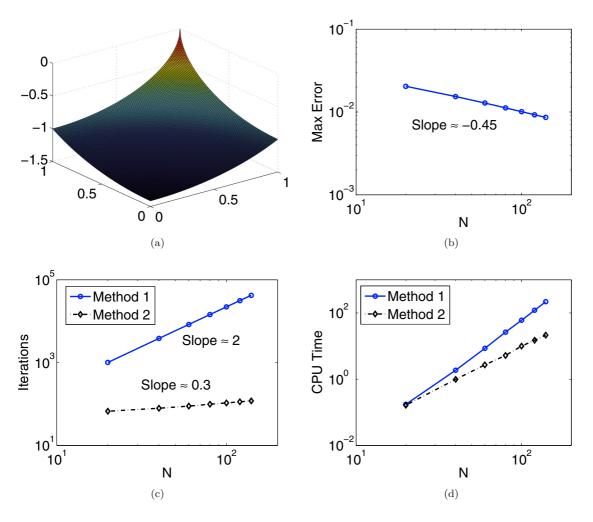


FIGURE 8. Results for  $u(x,y) = -\sqrt{2-x^2-y^2}$  on an  $N \times N$  grid. (a) Surface plot of the solution. (b) Error versus N. (c) Number of iterations. (d) Total CPU time.

error is zero. However, the rate of convergence for iterations to a fixed point is  $O(N^{-5})$  for (M1) and the  $O(N^{-2})$  for (M2).

## 6.1. **Cone**

Next we consider the cone  $u(x,y) = \sqrt{x^2 + y^2}$ . To obtain this solution with our methods we set f to be the appropriately weighted delta distribution at the origin, which discretizes to

$$f_{i,j} = \begin{cases} 4/h^2 & x_i = y_j = 0\\ 0 & \text{otherwise,} \end{cases}$$

(centering the grid so that the origin is a grid point), so that (MA) is satisfied in the sense of distributions. Despite the fact that this solution is only Lipschitz continuous, both still converge to the exact solution; see Table 10 and Figure 9, however the rate of convergence is approximately  $O(h^{.8})$  and the computational time for (M2) is about two orders of magnitude longer than for (M1) (Tab. 11).

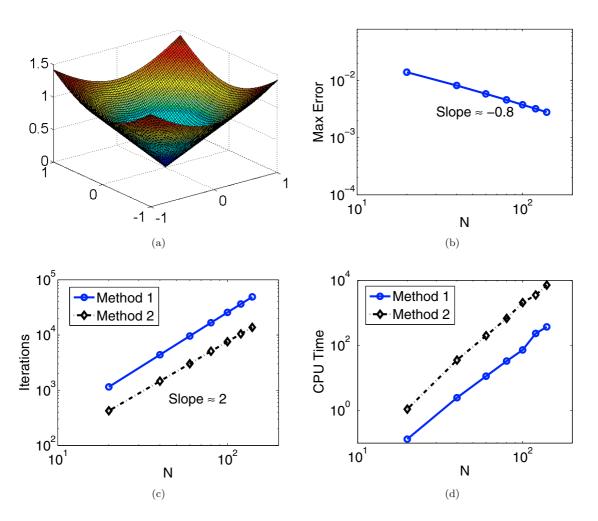


FIGURE 9. Results for  $u(x,y) = \sqrt{x^2 + y^2}$  on an  $N \times N$  grid. (a) Surface plot of the solution. (b) Error versus N. (c) Total number of iterations. (d) Total CPU time.

Table 10. Errors for the exact solution  $u(x,y) = \sqrt{x^2 + y^2}$  on an  $N \times N$  grid.

N	21	61	101	141
$  u-u^{(N)}  _{\infty}$	$1.4 \times 10^{-2}$	$5.9\times10^{-3}$	$3.8\times10^{-3}$	$2.8\times10^{-3}$

## 6.2. Absolute value function

Next we consider the function u(x,y) = |x|, which is Lipschitz continuous but not differentiable. Since even at the edge one eigenvalue is zero,  $\det(D^2u) = 0$  in the sense of distributions. Therefore setting f = 0, all methods correctly find this solution. Moreover, it appears that regardless of the value of N, the error in the approximations can be made arbitrarily small by iterating sufficiently many times (since the solution is piecewise linear).

Table 11. Computation times for  $u(x,y) = \sqrt{x^2 + y^2}$  on an  $N \times N$  grid.

			~	. / \	
Ν	Itera	$_{ m tions}$	CPU t	CPU time (s)	
	(M1)	(M2)	(M1)	(M2)	
21	1148	424	0.1	1.1	
41	4386	1458	2.5	35.6	
61	9588	3017	11.4	200.0	
81	16686	5082	33.2	681.0	
101	25635	7561	73.3	2090.0	
121	36396	10486	235.0	3560.0	
141	48946	13750	372.0	7160.0	

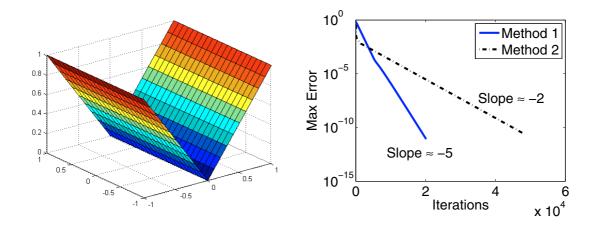


FIGURE 10. Results for u(x) = |x| on an  $N \times N$  grid. (a) Surface Plot. (b) Error versus number of iterations for N = 81.

Table 12. Computation times for u(x,y) = |x| on an  $N \times N$  grid.

$\overline{N}$	Itera	Iterations		ime (s)
	(M1)	(M2)	(M1)	(M2)
21	1417	3604	0.2	9
41	5366	13252	1.9	164
61	11636	28233	12.0	878
81	20135	48116	37.3	2440
101	30795	72727	83.2	6650

Although both methods appear to be exact for this example, (M1) performs much more quickly for this degenerate example. We plot maximum error against number of iterations in Figure 10(b). From this plot we observe that the error decreases like the fifth power of the number of iterations for (M1), but only decreases quadratically for (M2). The difference in computational time is even more pronounced; see Table 12.

Table 13. Errors for the exact solution  $u(x,y) = \sqrt{x^2 + y^2}$  on an  $N \times N$  grid.

$\overline{N}$	$\ u-u^{(N)}\ _{\infty}$	
	(M1)	(M1*)
21	$1.4 \times 10^{-2}$	$2.5 \times 10^{-3}$
41	$8.2 \times 10^{-3}$	$1.5 \times 10^{-3}$
61	$5.9 \times 10^{-3}$	$1.1 \times 10^{-3}$
81	$4.6 \times 10^{-3}$	$8.4 \times 10^{-4}$
101	$3.8 \times 10^{-3}$	$7.0 \times 10^{-4}$
121	$3.2 \times 10^{-3}$	$6.0 \times 10^{-4}$
141	$2.8\times10^{-3}$	$5.3\times10^{-4}$

# 7. Extensions to other methods

Several modifications to our methods can be used to improve the results in some of the non-smooth examples.

## 7.1. Additional directional convexity

The convergent finite difference method of [21] enforced directional convexity in many directions by using a wide stencil method. The purpose of this section is to explore whether increasing the directions of convexity, while maintaining a narrow stencil improves the methods.

Along the lines of [20,21] (where convexity along several directions ensures convergence), we can also build more convexity requirements into our method. This is accomplished by modifying (M1) as follows:

$$u_{i,j} = \min \left\{ \frac{a_1 + a_2}{2} - \sqrt{\left(\frac{a_1 - a_2}{2}\right)^2 + \left(\frac{a_3 - a_4}{4}\right)^2 + \frac{1}{4}f_{i,j}h^4}, \ a_3, \ a_4 \right\}.$$
 (M1\*)

**Lemma 7.1.** The fixed point of (M1\*) satisfies the inequality (2.3) for the grid directions

$$v \in \{(1,0), (0,1), (-1,1), (1,1)\}.$$

*Proof.* The proof of the first part of this lemma is the same as the proof of the first part of Lemma 2.2. The second half of this lemma is built directly into  $(M1^*)$ .

In the majority of cases, this modified method performs comparably (or identically) to (M1), with both methods converging to the same solution. One notable exception to this trend is the cone. Although the convergence time is the same for both (M1) and (M1\*), the results obtained with (M1\*) are roughly one order of magnitude more accurate; see Table 13.

## 7.2. Finite element implementation

Method (M2) involves iteratively solving Poisson's equation. The Poisson solver can be implemented using the method of choice. We tried a finite element solver which took advantage of mesh adaptation, to see if the performance of the method improved.

We implemented the Poisson solver of (M2) using a Finite Element Lagrangian P2 Poisson solver, FreeFem++. This software incorporates an anisotropic mesh adaptative algorithm. At each iteration of (M2) the mesh is adapted based on the magnitude of the norm of the Hessian of the solution.

In most examples, this finite element implementation is not significantly faster than the finite difference implementation. The one significant difference was observed for the example of Section 5.3. The mesh appears to have lined up well with the singularities of the solution, see Figure 11. In this case, the method, initialized

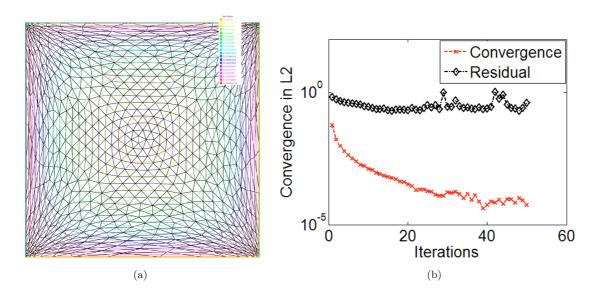


FIGURE 11. Results with f=1 and homogeneous Dirichlet boundary conditions initialized on a uniform  $40 \times 40$  mesh. (a) Final mesh and level curves of the solution. (b) Convergence and residual against number of iterations.

on a uniform  $40 \times 40$  mesh, converged in about 40 iterations whereas the finite difference implementation requires thousands of iterations (compare to Tab. 8).

## 7.3. Preconditioning

It is natural to look for other methods of solving the discretized equations  $(MA)^h$  that might be faster than either (M1), (M2).

Since (M2) is faster than (M1) when solutions are regular, which, by the regularity theory, occurs when the source term f is bounded away from zero, we suggest the following:

- (1) Replace f with  $f^h = \max(f, h^2)$ . With this source term, the solution of (MA) is smooth (unless there are singularities on the boundary, as in Sect. 5.3).
- (2) Approximately solve (MA) with  $f^h$ , for  $u^h$ , using (M2).
- (3) Solve (MA) with f using (M1), using  $u^h$  as initial data.

While we didn't pursue this, we offer it as a starting point for accelerating convergence to the solution of the equation  $(MA)^h$ .

## 7.4. Higher dimensions

One challenge in extending these methods to higher dimensions is enforcing convexity of the solution. Method (M1), for instance, involves discretizing (MA) and solving a quadratic equation in two dimensions; this method does not immediately generalize to higher dimensions. Method (M2) can, of course, be used in higher dimensions, although additional work may be needed to enforce convexity. We leave the higher dimensional case to future work.

# 8. Conclusions

Two simple numerical methods (M1), (M2) were introduced for solving (MA). Computations were performed on a number of solutions whose regularity ranged from smooth to strongly singular. Performance comparisons were made between the two methods and between results available for existing methods.

The first method, (M1), is a simple explicit finite difference method. While it is second order accurate, it is not monotone, so there is no convergence proof available. However, the method appeared to converge for all the numerical examples computed. These examples ranged from very smooth to singular (not in  $\mathbf{H}^2(\Omega)$ , Lipschitz continuous). The observed convergence rate went from two (second order) for smooth solutions, down to one half for singular solutions. The discrete equation was solved by a simple iterative method, with the number of iterations (for fixed problem size) roughly independent of the regularity of solution.

We concluded that (M1), which is simply the natural finite difference discretization of the equation, is the *simplest* and *best performing* method currently available, for generic problems. Only in the most singular case studied did the provably convergent method of [21] perform better. However, in this case we introduced a simple modification of (M1) which improved accuracy by an order of magnitude, Section 7.1.

The second method, (M2) solved the same discrete equation  $(MA)^h$  using a different iterative method. In this case, an identity satisfied by (MA) was used to write a Poisson equation involving the Hessian of the solution and the source term (2.2). This method is *simple* to implement and demonstrably *superior* to other methods that require solutions to be in  $\mathbf{H}^2(\Omega)$ . It converges on a wider range of solutions, and each iteration is simply a Poisson solve.

Comparing the two methods, Method (M2) was faster (by at least an order of magnitude) on smooth solutions. However, strongly singular solutions led to a poorly conditioned Poisson equation and, for the most singular solutions, (M2) was slower (by more than an order of magnitude) than (M1).

In summary, Method (M2) is the method of choice for solutions which are known to be in  $\mathbf{H}^2(\Omega)$  or better, while Method (M1) is the method of choice for generic problems where the regularity is not known. In particular, if the source term, f, is sometimes zero, solutions may be singular, and (M1) should be used.

# References

- [1] I. Bakelman, Convex analysis and nonlinear geometric elliptic equations. Springer-Verlag, Germany (1994).
- [2] G. Barles and P.E. Souganidis, Convergence of approximation schemes for fully nonlinear second order equations. *Asymptotic Anal.* 4 (1991) 271–283.
- [3] K. Böhmer, On finite element methods for fully nonlinear elliptic equations of second order. SIAM J. Numer. Anal. 46 (2008) 1212–1249.
- [4] L.A. Caffarelli and M. Milman, Eds., Monge Ampère equation: applications to geometry and optimization, Contemporary Mathematics 226. American Mathematical Society, Providence, USA (1999).
- [5] L. Caffarelli, L. Nirenberg and J. Spruck, The Dirichlet problem for nonlinear second-order elliptic equations. I. Monge-Ampère equation. Comm. Pure Appl. Math. 37 (1984) 369–402.
- [6] M.G. Crandall, H. Ishii and P.-L. Lions, User's guide to viscosity solutions of second order partial differential equations. Bull. Amer. Math. Soc. (N.S.) 27 (1992) 1–67.
- [7] E.J. Dean and R. Glowinski, An augmented Lagrangian approach to the numerical solution of the Dirichlet problem for the elliptic Monge-Ampère equation in two dimensions. *Electron. Trans. Numer. Anal.* **22** (2006) 71–96.
- [8] E.J. Dean and R. Glowinski, Numerical methods for fully nonlinear elliptic equations of the Monge-Ampère type. Comput. Methods Appl. Mech. Engrg. 195 (2006) 1344–1386.
- [9] E.J. Dean and R. Glowinski, On the numerical solution of the elliptic Monge-Ampère equation in dimension two: a least-squares approach, in *Partial differential equations, Comput. Methods Appl. Sci.* 16, Springer, Dordrecht, The Netherlands (2008) 43–63.
- [10] E.J. Dean, R. Glowinski and T.-W. Pan, Operator-splitting methods and applications to the direct numerical simulation of particulate flow and to the solution of the elliptic Monge-Ampère equation, in *Control and boundary analysis*, *Lect. Notes* Pure Appl. Math. 240, Chapman & Hall/CRC, Boca Raton, USA (2005) 1–27.
- [11] L.C. Evans, Partial differential equations and Monge-Kantorovich mass transfer, in *Current developments in mathematics*, 1997 (Cambridge, MA), Int. Press, Boston, USA (1999) 65–126.
- [12] X. Feng and M. Neilan, Galerkin methods for the fully nonlinear Monge-Ampère equation. http://arxiv.org/abs/0712.1240v1 (2007).

- [13] X. Feng and M. Neilan, Mixed finite element methods for the fully nonlinear Monge-Ampère equation based on the vanishing moment method. SIAM J. Numer. Anal. 47 (2009) 1226–1250.
- [14] X. Feng and M. Neilan, Vanishing moment method and moment solutions for fully nonlinear second order partial differential equations. J. Sci. Comput. 38 (2009) 74–98.
- [15] R. Glowinski, Numerical methods for fully nonlinear elliptic equations, in 6th International Congress on Industrial and Applied Mathematics, ICIAM 07, Invited Lectures, R. Jeltsch and G. Wanner Eds. (2009) 155–192.
- [16] R. Glowinski, E.J. Dean, G. Guidoboni, L.H. Juárez and T.-W. Pan, Applications of operator-splitting methods to the direct numerical simulation of particulate and free-surface flows and to the numerical solution of the two-dimensional elliptic Monge-Ampère equation. *Japan J. Indust. Appl. Math.* 25 (2008) 1–63.
- [17] C.E. Gutiérrez, The Monge-Ampère equation, Progress in Nonlinear Differential Equations and their Applications 44. Birkhäuser Boston Inc., Boston, USA (2001).
- [18] G. Loeper and F. Rapetti, Numerical solution of the Monge-Ampère equation by a Newton's algorithm. C. R. Math. Acad. Sci. Paris 340 (2005) 319–324.
- [19] A.M. Oberman, Convergent difference schemes for degenerate elliptic and parabolic equations: Hamilton-Jacobi equations and free boundary problems. SIAM J. Numer. Anal. 44 (2006) 879–895.
- [20] A.M. Oberman, Computing the convex envelope using a nonlinear partial differential equation. Math. Models Methods Appl. Sci. 18 (2008) 759–780.
- [21] A.M. Oberman, Wide stencil finite difference schemes for the elliptic Monge-Ampère equation and functions of the eigenvalues of the Hessian. *Discrete Contin. Dyn. Syst. Ser. B* 10 (2008) 221–238.
- [22] V.I. Oliker and L.D. Prussner, On the numerical solution of the equation  $(\partial^2 z/\partial x^2)(\partial^2 z/\partial y^2) (\partial^2 z/\partial x\partial y)^2 = f$  and its discretizations, I. Numer. Math. **54** (1988) 271–293.