J Sci Comput (2018) 75:1282–1306
https://doi.org/10.1007/s10915-017-0586-5

Higher-Order Adaptive Finite Difference Methods for Fully Nonlinear Elliptic Equations

Brittany Froese Hamfeldt[1] · **Tiago Salvador**[2]

Received: 23 June 2017 / Revised: 11 October 2017 / Accepted: 14 October 2017 /
Published online: 24 October 2017

Abstract We introduce generalised finite difference methods for solving fully nonlinear elliptic partial differential equations. Methods are based on piecewise Cartesian meshes augmented by additional points along the boundary. This allows for adaptive meshes and complicated geometries, while still ensuring consistency, monotonicity, and convergence. We describe an algorithm for efficiently computing the non-traditional finite difference stencils. We also present a strategy for computing formally higher-order convergent methods. Computational examples demonstrate the efficiency, accuracy, and flexibility of the methods.

Keywords Finite difference methods · Fully nonlinear elliptic partial differential equations · Adaptive methods · Higher-order methods

Mathematics Subject Classification 35J15 · 35J60 · 35J96 · 65N06 · 65N12 · 65N50

1 Introduction

In this article we design generalised finite difference methods for a large class of fully nonlinear degenerate elliptic partial differential equations (PDEs). The approximation schemes are almost-monotone, which allows us to exploit the Barles-Souganidis convergence frame-

The first author was partially supported by NSF DMS-1619807. The second author was partially supported by NSERC Discovery Grant RGPIN-2016-03922 and by Fundação para a Ciência e Tecnologia (FCT) Doctoral Grant (SFRH/BD/84041/2012).

✉ Brittany Froese Hamfeldt
bdfroese@njit.edu

Tiago Salvador
saldanha@umich.edu

[1] Department of Mathematical Sciences, New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA

[2] Department of Mathematics, University of Michigan, 530 Church Street, Ann Arbor, MI 48109, USA

work. A key feature of these methods is the use of piecewise Cartesian grids, augmented by additional discretisation points along the boundary. Because of the underlying structure of the grids, the methods we design overcome several hurdles that exist for current numerical methods. (1) The non-standard finite difference stencils can be constructed efficiently. (2) The monotone approximation schemes preserve consistency near the boundary. (3) Higher-order schemes are easily designed. (4) Complicated geometries are easily handled.

1.1 Background

Fully nonlinear elliptic partial differential equations (PDEs) arise in numerous applications including reflector/refractor design [22], meteorology [10], differential geometry [8], astrophysics [17], seismology [12], mesh generation [7], computer graphics [27], and mathematical finance [16]. Realistic applications often involve complicated domains and highly non-smooth data. Thus the development of robust numerical methods that are compatible with adaptive mesh refinement is a priority.

In recent years, the numerical solution of these equations has received a great deal of attention, and several new methods have been developed including finite difference methods [3,15,23,28,30], finite element methods [1,5,6,29], least squares methods [11], and methods involving fourth-order regularisation terms [14]. However, these methods are not designed to compute weak solutions. When the ellipticity of the equation is degenerate or no smooth solution exists, methods become very slow, are unstable, or converge to an incorrect solution.

Using a framework developed by Barles and Souganidis [2], provably convergent (monotone) methods have recently been constructed for several fully nonlinear equations using wide finite difference stencils [26]. Recently, the idea of wide stencil schemes has been adapted to produce monotone approximations for a large class of fully nonlinear elliptic operators posed on very general meshes or points clouds [20]. However, constructing the necessary finite difference stencils is a very expensive process, and the resulting approximations have only $\mathcal{O}(\sqrt{h})$ accuracy.

1.2 Contributions of This Work

The goal of this article is to design higher-order, adaptive, convergent finite difference methods for the solution of the degenerate elliptic PDE

$$F(x, u(x), D^2u(x)) \equiv \max_{\theta\in[0,2\pi)} F_\theta(x, u(x), u_{\theta\theta}(x)) = 0, \tag{1}$$

where $u_{\theta\theta}$ denotes the second directional derivative of u in the direction $e_\theta = (\cos\theta, \sin\theta) \in \mathbb{R}^2$. This includes a wide range of PDE operators including monotone functions of the eigenvalues of the Hessian matrix D^2u and general convex functions of the Hessian matrix [13, Proposition 5.3]. We note that the maxima can also be taken over a subset of directions by simply setting $F_\theta = -1$ for directions that are not active in the PDE operator. The methods described in this article can also be trivially adapted to include minima and other monotone functions of the operators F_θ.

We remark that the PDE operator can also include Lipschitz continuous dependence on the gradient ∇u. Monotone approximation of first-order operators is fairly well-established and does not require the wide-stencil structure that is necessary to correctly discretise second-order operators. In this article, we omit terms involving the gradient for the sake of compactness and clarity.

We take as a starting point the meshfree finite difference approximations developed in [20]. Given a set of discretisation points $\mathcal{G}$, that work introduced a generalised finite difference approximation of (1) of the form

$$\max_{\theta\in\mathcal{A}\subset[0,2\pi)} F_\theta\left(x_i, u(x_i), \sum_{j\in\mathcal{N}(i,\theta)} a_{i,j,\theta}(u(x_i)-u(x_j))\right) = 0, \quad x_i \in \mathcal{G}. \tag{2}$$

Above, the set of neighbours $\mathcal{N}(i,\theta)$ gives indices of several points that are near x_i and align closely with the e_θ direction. In the original paper, these sets of neighbours are computed through brute force, by finding and inspecting all nodes lying within a distance $\sqrt{h}$ of each other.

In this article, we describe the construction of piecewise Cartesian meshes using a quadtree structure, which is augmented by additional discretisation points along the boundary in order to preserve consistency. We demonstrate that the resulting set of discretisation points satisfies the conditions required by [20, Theorem 13]. This ensures that it is possible to build monotone (convergent) approximations. Moreover, the structure of these quadtrees allows for easy mesh adaptation, with refinement criteria that can either be specified a priori or determined automatically from the quality of the solution of (2).

By exploiting the underyling structure of the quadtree meshes, we design an efficient method for constructing the set of neighbours $\mathcal{N}(i,\theta)$ required by our generalised finite difference stencils. This leads to a much faster numerical method for approximating solutions of (1) that can easily handle singular solutions, complicated geometries, and highly non-uniform meshes.

Finally, we describe a strategy for producing higher-order almost-monotone methods using the framework of [21]. This requires combining the monotone scheme with a formally higher-order approximation. By utilising the structure and symmetry within the quadtree mesh, we obtain a simple strategy for constructing higher-order schemes in the interior of the domain. Near the boundary, where these simple schemes no longer exist, we propose a least-squares approach to constructing higher-order schemes.

1.3 Contents

In Sect. 2, we review the theory of generalised finite difference approximations for fully nonlinear elliptic equations. In Sect. 3, we describe our strategy for constructing meshes and finite difference stencils. In Sect. 4, we describe a higher-order implementation of these methods. In Sect. 5, we present several computational examples. Finally, in Sect. 6, we provide conclusions and perspectives.

2 Generalised Finite Difference Schemes

In this section we review existing results on the construction and convergence of numerical methods for fully nonlinear elliptic equations.

2.1 Weak Solutions

One of the challenges associated with the approximation of fully nonlinear PDEs is the fact that classical (smooth) solutions may not exist. It thus becomes necessary to interpret PDEs using some notion of weak solution, and the numerical methods that are used need to respect

this notion of weak solution. The most common concept of weak solution for this class of PDEs is the *viscosity solution*, which involves transferring derivatives onto smooth test functions via a maximum principle argument [9].

The PDEs we consider in this work belong to the class of degenerate elliptic equations,

$$F(x, u(x), D^2u(x)) = 0, \quad x \in \bar{\Omega} \subset \mathbb{R}^2. \tag{3}$$

Definition 1 (*Degenerate elliptic*) The operator $F : \bar{\Omega} \times \mathbb{R} \times \mathcal{S}^2 \to \mathbb{R}$ is *degenerate elliptic* if

$$F(x, u, X) \leq F(x, v, Y)$$

whenever $u \leq v$ and $X \geq Y$.

We note that the operator is also defined on the boundary $\partial\Omega$ of the domain, which allows both the PDE and the boundary conditions to be contained within Eq. (3). For example, if Dirichlet data $u(x) = g(x)$ is given on the boundary, the elliptic operator at the boundary will be defined by

$$F(x, u(x), D^2u(x)) = u(x) - g(x), \quad x \in \partial\Omega.$$

The PDE operators (1) that we consider in this work are degenerate elliptic if they are non-decreasing functions of their second argument (u) and non-increasing functions of all subsequent arguments (which involve second directional derivatives).

Since degenerate elliptic equations need not have classical solutions, solutions need to be interpreted in a weak sense. The numerical methods developed in this article are guided by the very powerful concept of the viscosity solution [9]. Checking the definition of the viscosity solution requires checking the value of the PDE operator for smooth test functions lying above or below the semi-continuous envelopes of the candidate solution.

Definition 2 (*Upper and lower semi-continuous envelopes*) The *upper and lower semi-continuous envelopes* of a function $u(x)$ are defined, respectively, by

$$u^*(x) = \limsup_{y \to x} u(y),$$
$$u_*(x) = \liminf_{y \to x} u(y).$$

Definition 3 (*Viscosity subsolution (supersolution)*) An upper (lower) semi-continuous function u is a *viscosity subsolution (supersolution)* of (1) if for every $\phi \in C^2(\bar{\Omega})$, whenever $u - \phi$ has a local maximum (minimum) at $x \in \bar{\Omega}$, then

$$F_*^{(*)}(x, u(x), D^2\phi(x)) \leq (\geq) 0.$$

Definition 4 (*Viscosity solution*) A function u is a *viscosity solution* of (1) if u^* is a subsolution and u_* a supersolution.

An important property of many elliptic equations is the comparison principle, which immediately implies uniqueness of the solution.

Definition 5 (*Comparison principle*) A PDE has a *comparison principle* if whenever u is an upper semi-continuous subsolution and v a lower semi-continuous supersolution of the equation, then $u \leq v$ on $\bar{\Omega}$.

2.2 Convergence of Elliptic Schemes

In order to construct convergent approximations of elliptic operators, we will rely on the framework provided by Barles and Souganidis [2] and further developed by Oberman [24].

We consider finite difference schemes that have the form

$$F^\epsilon(x, u(x), u(x) - u(\cdot)) = 0 \tag{4}$$

where ϵ is a small parameter.

The convergence framework requires notions of consistency and monotonicity, which we define below.

Definition 6 (*Consistency*) The scheme (4) is *consistent* with the Eq. (1) if for any smooth function ϕ and $x \in \bar{\Omega}$,

$$\limsup_{\epsilon \to 0^+, y \to x, \xi \to 0} F^\epsilon(y, \phi(y) + \xi, \phi(y) - \phi(\cdot)) \leq F^*(x, \phi(x), \nabla\phi(x), D^2\phi(x)),$$

$$\liminf_{\epsilon \to 0^+, y \to x, \xi \to 0} F^\epsilon(y, \phi(y) + \xi, \phi(y) - \phi(\cdot)) \geq F_*(x, \phi(x), \nabla\phi(x), D^2\phi(x)).$$

Definition 7 (*Monotonicity*) The scheme (4) is monotone if F^ϵ is a non-decreasing function of its final two arguments.

Schemes that satisfy these two properties respect the notion of the viscosity solution at the discrete level. In particular, these schemes preserve the maximum principle and are guaranteed to converge to the solution of the underlying PDE.

Theorem 1 (Convergence [24]) *Let u be the unique viscosity solution of the PDE (1), where F is a degenerate elliptic operator with a comparison principle. Let the finite difference approximation F^ϵ be consistent and monotone and let u^ϵ be any solution of the scheme (4), with bounds independent of ϵ. Then u^ϵ converges uniformly to u as $\epsilon \to 0$.*

The above theorem assumes existence of a bounded solution to the approximation scheme. This is typically straightforward to show for a consistent, monotone approximation of a well-posed PDE, though the precise details can vary slightly and rely on available well-posedness theory for the PDE in question.

Theorem 2 (Existence and Stability [19, Lemmas 35–36]) *Let F^ϵ be a consistent, monotone scheme that is Lipschitz in its last two arguments. Suppose also that there exist strict classical sub- and super-solutions to the PDE (3). Then for small enough $\epsilon > 0$, the scheme (4) has a solution u^ϵ. Moreover, there exists a constant $M > 0$ such that $\|u^\epsilon\|_\infty \leq M$ for sufficiently small $\epsilon > 0$.*

In many cases, simple quadratic functions will serve as the sub- and super-solutions required by Theorem 2. For more complicated PDE operators, particularly those with a non-trivial dependence on the gradient ∇u, the theory of classical solutions of the equation can often be used to show the existence of these sub- and super-solutions.

2.3 Meshfree Finite Difference Approximations

In [20], a new generalised finite difference method was introduced for approximating fully nonlinear second order elliptic operators on point clouds. We review the key results of that work, which will be foundational to the higher-order adaptive methods that will be developed in the remainder of this article.

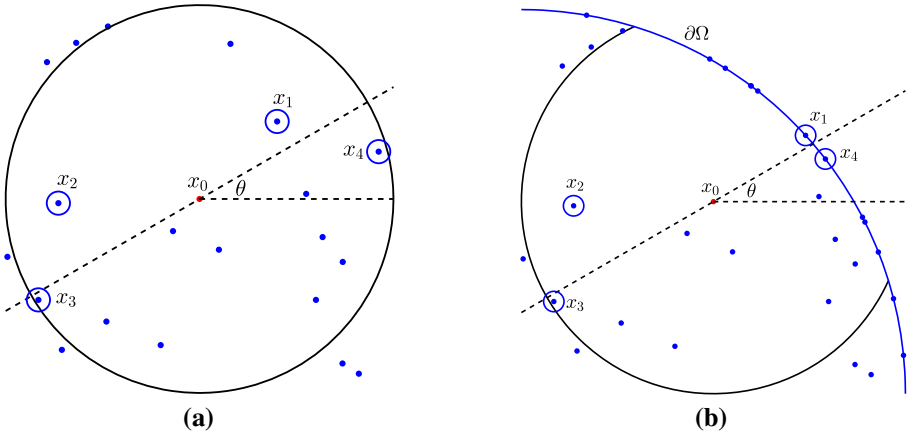

Fig. 1 A finite difference stencil chosen from a point cloud **a** in the interior and **b** near the boundary

Definition 8 (*Notation*)

(N1) $\Omega \subset \mathbb{R}^2$ is a bounded domain with Lipschitz boundary $\partial\Omega$.
(N2) $\mathcal{G} \subset \bar{\Omega}$ is a point cloud consisting of the points x_i, $i = 1, \ldots, N$.
(N3) $h = \sup_{x\in\Omega} \min_{y\in\mathcal{G}} |x - y|$ is the spatial resolution of the point cloud. In particular, every ball of radius h contained in $\bar{\Omega}$ contains at least one discretisation point x_i.
(N4) $h_B = \sup_{x\in\partial\Omega} \min_{y\in\mathcal{G}\cap\partial\Omega} |x - y|$ is the resolution of the point cloud on the boundary. In particular, every ball of radius h_B centred at a boundary point $x \in \partial\Omega$ contains at least one discretisation point $x_i \in \mathcal{G} \cap \partial\Omega$ on the boundary.
(N5) $\delta = \min_{x\in\Omega\cap\mathcal{G}} \inf_{y\in\partial\Omega} |x-y|$ is the distance between the set of interior discretisation points and the boundary. In particular, if $x_i \in \mathcal{G} \cap \Omega$ and $x_j \in \partial\Omega$, then the distance between x_i and x_j is at least δ.
(N6) $d\phi$ is the angular resolution used to approximate the second directional derivatives $u_{\theta\theta}$.
(N7) $d\theta$ is the angular resolution used to approximate the nonlinear operator.
(N8) ϵ is the search radius associated with the point cloud.

Discretising the PDE requires approximating second directional derivatives $u_{\theta\theta}$ at each interior discretisation point $x_i \in \mathcal{G}$. To accomplish this, we consider all points $x_j \in \mathcal{G} \cap B(x_i, \epsilon)$ within a search neighbourhood of radius ϵ centred at x_i. Discretisation points within this neighbourhood can be written in polar coordinates (r, ϕ) with respect to the axes defined by the lines $x_0 + t(\cos\theta, \sin\theta)$, $x_0 + t(-\sin\theta, \cos\theta)$. We seek one neighbouring discretisation point in each quadrant described by these axes, with each neighbour aligning as closely as possible with the line $x_0 + tv$, where $v = (\cos\theta, \sin\theta)$. That is, we select the neighbours

$$x_j \in \operatorname{argmin} \left\{ \sin^2\phi \mid (r, \phi) \in \mathcal{G}^h \cap B(x_0, \epsilon) \text{ is in the } j\text{th quadrant} \right\} \tag{5}$$

for $j = 1, \ldots, 4$. See Fig. 1. We say that a stencil with angular resolution $d\phi$ exists for the point cloud $\mathcal{G}$ if for all interior discretisation points, the four discretisation points $x_j \in \mathcal{G}$ defined by (5) exist and satisfy $d\phi = \max\{\phi_j\}$.

Because of the "wide-stencil" nature of these approximations (since the search radius $\epsilon \gg h$), care must be taken near the boundary. In order to preserve consistency up to

the boundary, it is necessary that the boundary be more highly resolved than the interior ($h_B \ll h$). In particular, this means that a simple Cartesian mesh (or piecewise Cartesian mesh) is *not* sufficient for producing consistent schemes up to the boundary.

Then a consistent, monotone approximation of $u_{\theta\theta}$ is

$$\mathcal{D}^h_{\theta\theta} u(x_0) = \sum_{j=1}^{4} a_j (u(x_j) - u(x_0))$$

where we use the polar coordinate characterisation of the neighbours to define

$$S_j = r_j \sin\phi_j, \quad C_j = r_j \cos\phi_j$$

and the coefficients are given by

$$\begin{aligned}
a_1 &= \frac{2S_4(C_3S_2 - C_2S_3)}{(C_3S_2 - C_2S_3)(C_1^2S_4 - C_4^2S_1) - (C_1S_4 - C_4S_1)(C_3^2S_2 - C_2^2S_3)} \\
a_2 &= \frac{2S_3(C_1S_4 - C_4S_1)}{(C_3S_2 - C_2S_3)(C_1^2S_4 - C_4^2S_1) - (C_1S_4 - C_4S_1)(C_3^2S_2 - C_2^2S_3)} \\
a_3 &= \frac{-2S_2(C_1S_4 - C_4S_1)}{(C_3S_2 - C_2S_3)(C_1^2S_4 - C_4^2S_1) - (C_1S_4 - C_4S_1)(C_3^2S_2 - C_2^2S_3)} \\
a_4 &= \frac{-2S_1(C_3S_2 - C_2S_3)}{(C_3S_2 - C_2S_3)(C_1^2S_4 - C_4^2S_1) - (C_1S_4 - C_4S_1)(C_3^2S_2 - C_2^2S_3)}.
\end{aligned}$$

In general, the PDE requires evaluating second directional derivatives in all possible directions. Instead, we consider a finite subset $\mathcal{A} = \left\{ jd\theta \mid j = 0, \ldots, \lfloor \frac{2\pi}{d\theta} \rfloor \right\} \subset [0, 2\pi)$ with a resolution $d\theta$.

Then we can substitute these coefficients into (2) to obtain the scheme:

$$F_i[u] \equiv \max_{\theta \in \mathcal{A}} F_\theta \left(x_i, u(x_i), \sum_{j \in \mathcal{N}(i,\theta)} a_{i,j,\theta}(u(x_i) - u(x_j)) \right) = 0, \quad x_i \in \mathcal{G}. \tag{6}$$

We recall the convergence result from [20, Theorem 18].

Theorem 3 (Convergence) *Let F be a degenerate elliptic operator with a comparison principle that is Lipschitz continuous in $u_{\theta\theta}$ for each $\theta \in [0, 2\pi)$ and let u be the unique viscosity solution of the PDE (1). Suppose also that (1) has a strict classical sub- and super-solution. Consider a sequence of point clouds $\mathcal{G}^n$, with parameters defined as in Definition 8, which satisfy the following conditions.*

- *The spatial resolution $h^n \to 0$ as $n \to \infty$.*
- *The boundary resolution satisfies $h_B^n/\delta^n \to 0$ as $n \to \infty$.*
- *The search radius satisfies both $\epsilon^n \to 0$ and $h^n/\epsilon^n \to 0$ as $n \to \infty$.*
- *The angular resolution $d\theta^n \to 0$ as $h^n \to 0$.*

Then for sufficiently large n, the approximation scheme (6) admits a solution u^n and u^n converges uniformly to u as $n \to \infty$.

We note that the angular resolution that emerges from the scheme satisfies $d\phi = \mathcal{O}(\max\{h/\epsilon, h_B/\delta\})$ (Fig. 2). For a uniform grid, a natural choice of parameters is $\epsilon = \mathcal{O}(\sqrt{h})$, $h_B = \mathcal{O}(h^{3/2})$, $\delta = \mathcal{O}(h)$, $d\theta = \mathcal{O}(\sqrt{h})$. This leads to a formally optimal discretisation error of $\mathcal{O}(\sqrt{h})$.

We remark also that these parameters can be defined locally instead of globally in order to accommodate highly non-uniform meshes.

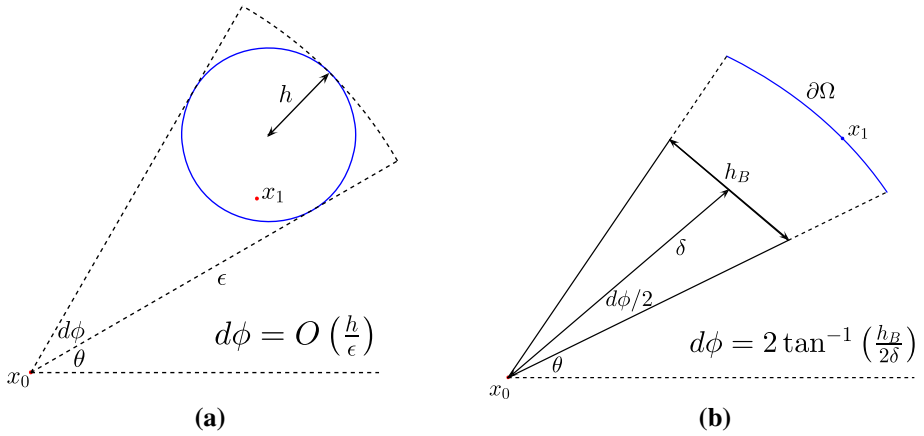

Fig. 2 The angular resolution of a generalised finite difference stencil

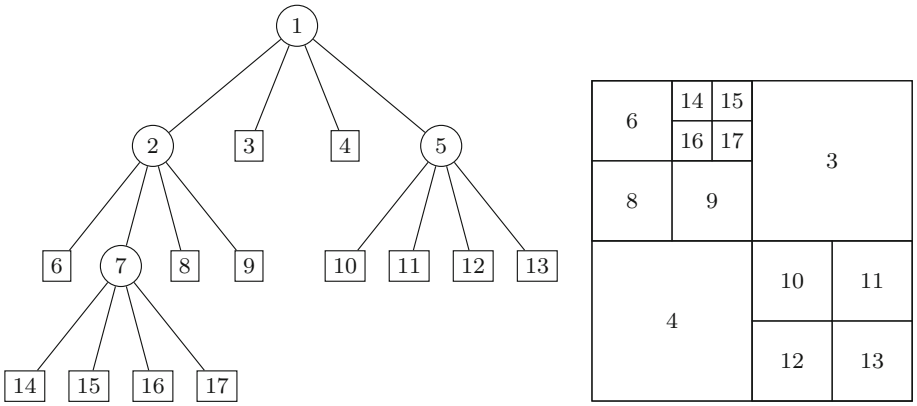

Fig. 3 A quadtree and the corresponding subdivision. The internal nodes are represented with circles and the leaves with squares

3 Construction of Meshes and Stencils

In this section, we explain how we use augmented quadtrees [4, Chapter 14] to build piecewise Cartesian meshes with additional discretisation points on the boundary. We organise this section as follows. In Sect. 3.1, we recall the basic structure of a quadtree. In Sect. 3.2 we explain how we augment the quadtree to deal with the boundary. In Sect. 3.4, we explain how the quadtrees are used to efficiently find the stencils. Finally, in Sect. 3.3 we discuss mesh adaptation.

3.1 Quadtrees

Quadtrees are based on a simple idea: a square can be divided into four smaller squares which correspond to the four quadrants of the square. A quadtree is then a rooted tree in which every internal node has four children and every node in the tree corresponds to a square. A square with no children is called a leaf square. See Fig. 3.

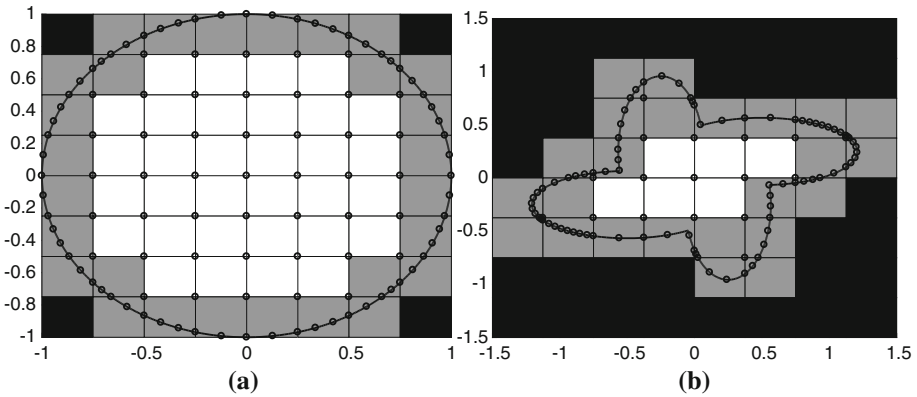

Fig. 4 Black squares are part of the quadtree but not used since they are not inside the domain. Grey squares intersect the boundary. White squares are inside the domain

Quadtrees can then easily be used to build uniform and non-uniform meshes: the squares' vertices are the mesh points. This structure is appealing because it is general enough to allow for local mesh adaptation, while still maintaining enough structure to efficiently build the finite difference stencils. Indeed, as we will see in Sect. 3.4, the quadtree structure allow us to significantly reduce the number of mesh points inspected when constructing our stencils. However, the quadtree in and of itself is not ideal for handling complicated geometries as the mesh points are restricted to be vertices of the squares. We observe that the global spatial resolution h of a quadtree corresponds to the length-scale of the largest leaf square. However, the quadtree can be highly non-uniform and the local spatial resolution near a particular point may be much less than h.

3.2 Meshing the Boundary

Quadtrees alone are not enough for the schemes proposed here: the boundary requires additional treatment. As discussed in Sect. 2, the boundary must be more highly resolved than the interior to maintain consistency of the numerical method. As a result, we cannot restrict the mesh points to be the vertices of the squares in the quadtree.

To overcome this, we build augmented quadtrees: each leaf square that intersects the boundary is marked as such and additional mesh points that lie on the boundary are added and associated with this boundary leaf square. The immediate advantage of this approach is that mesh points may lie exactly on the boundary, which allows us to handle complicated geometries with ease. In addition, by keeping track of which boundary leaf square the mesh points belong to, we preserve one of the key properties of the quadtree: knowledge of the relative position of the mesh points. This allows for efficient construction of the finite difference stencils.

We make the following general assumption: each edge of a leaf square intersects the boundary at most once. This is a reasonable assumption that simply entails that our quadtree must be sufficiently refined near the boundary. See Fig. 4, where each edge of the grey squares intersects the boundary at most once.

The only question left to address is exactly how many additional boundary mesh points one must add to guarantee the existence of a consistent stencil. This is addressed in Theorem 3, which requires that the boundary resolution go to zero more quickly than the resolution of

the "standard" quadtree, $h_B = o(h)$, and more quickly than the gap between the boundary and the interior nodes, $h_B = o(\delta)$. Note that these conditions need only be satisfied locally rather than globally.

We define a simple algorithm that enlarges a given point cloud so that the condition $h_B \leq 2\delta \tan(d\theta/2)$ is satisfied locally (see Algorithm 1). This ensures that the angular resolution of the finite difference approximations is commensurate with the angular resolution used to approximate the nonlinear operator: $d\phi \lesssim 2\tan^{-1}\left(\frac{h_B}{2\delta}\right) \leq d\theta$ (Fig. 2).

Algorithm 1 Building augmented quadtrees

1: **for** each boundary leaf square S **do**
2: Add the points in $\partial S \cap \partial\Omega$ to the point cloud $\mathcal{G}$.
3: Compute $X = \Omega \cap \mathcal{G} \cap S$.
4: Compute $\delta = \min_{x\in X} \min_{y\in\partial\Omega\cap S} |x - y|$.
5: Compute the arc length, l, of the curve $\partial\Omega \cap S$.
6: Compute the desired boundary resolution $h_B = 4\delta \tan(d\theta/2)$.
7: Select $\lceil l/h_B \rceil$ points lying on the curve $\partial\Omega \cap S$.
8: Add these points to the point cloud $\mathcal{G}$.
9: **end for**

In Fig. 4, the meshes obtained by applying Algorithm 1 to the point cloud obtained from a quadtree of depth 4 are displayed. The fan-shaped domain illustrates the advantages of the local criteria: the boundary is only highly resolved when there are interior mesh points nearby.

These augmented quadtrees enable us to construct convergent (consistent and monotone) finite difference approximations.

Lemma 1 (Approximation with augmented quadtrees) *Consider a sequence of augmented quadtrees $\mathcal{G}^n$ constructed via Algorithm 1 with spatial resolution $h_n \to 0$. Consider also a sequence of search radii $\epsilon^n = \mathcal{O}(\sqrt{h^n})$ and angular resolutions $d\theta^n = \mathcal{O}(\sqrt{h^n})$. Then $\mathcal{G}^n$ satisfies the hypotheses of Theorem 3.*

Proof We need only verify that $h_B^n/\delta^n \to 0$; the remaining conditions of Theorem 3 are trivially satisfied.

We recall that both h_B^n and δ^n can be defined locally. Indeed, for each boundary leaf square S we can let

$$h_{B,S}^n = \sup_{x\in\partial\Omega\cap S} \min_{y\in\mathcal{G}^n\cap\partial\Omega\cap S} |x - y|, \quad \delta_S^n = \min_{x\in\Omega\cap\mathcal{G}^n} \inf_{y\in\partial\Omega\cap S} |x - y|.$$

By construction, Algorithm 1 ensures that $h_{B,S}^n/\delta_S^n = \mathcal{O}(d\theta_S^n) \to 0$.

Moreover, it is sufficient to verify these conditions at boundary leaf squares; other interior squares will produce larger values of δ_S^n and smaller values of $h_{B,S}^n/\delta_S^n$. □

3.3 Refinement, Adaptivity and Balance

The use of quadtrees also provides a natural means of doing mesh adaptation. A refinement criteria can either be specified a priori or determined automatically from the quality of the solution.

In Fig. 5, we provide an example of a priori refinement: the mesh is refined near the corners of the domain.

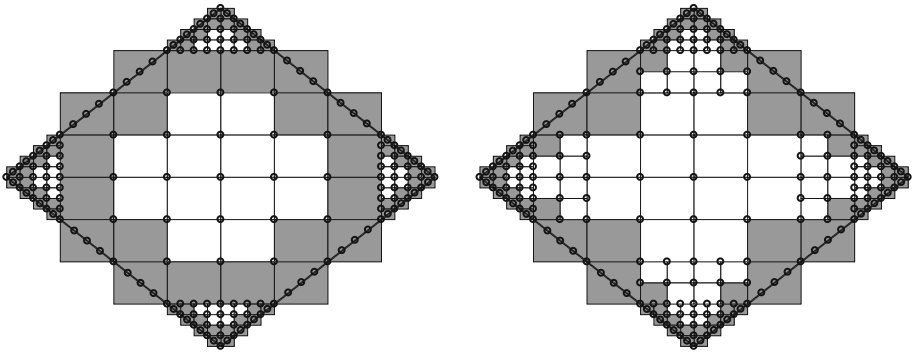

Fig. 5 A priori refinement near the corners of the domain: **a** unbalanced quadtree and **b** its balanced version

Simply refining the quadtree can lead to a very unbalanced quadtree when large squares adjoin several smaller squares. This is an undesirable property for our meshes as it makes the construction of high-order schemes significantly more difficult. Therefore, we always maintain a balanced quadtree: any two neighbouring squares differ by at most a factor of two in length scale (see Fig. 5). Balancing a quadtree can be done efficiently; we refer to [4, Theorem 14.4] for details.

3.4 Generating the Stencil

We explain how quadtrees are used to efficiently find the neighbours for each interior mesh point. The main idea is the following: given the quadtree structure we know the relative position of the mesh points and can significantly restrict the number of nodes we examine.

Consider a direction $\nu = (\cos\theta, \sin\theta)$ and the line $x_0 + t\nu$. Without loss of generality, assume the line has positive slope as in Fig. 6. We describe the procedure for finding the required neighbours of x_0 lying in the first and fourth quadrants.

Algorithm 2 Finding the neighbours of $x_0 \in \mathcal{G}$ in the first and fourth quadrant.

1: Identify the leaf square that has x_0 as its southwest vertex. This can be done efficiently since, when constructing the quadtree, a record is maintained of the (four) leaf squares that have each interior x_0 as a vertex.
2: Identify which edge(s) of this square intersect the line $x_0 + t\nu$, selecting the edge that yields the smaller value of t, t_{min} (i.e. the first edge to intersect this ray).
3: Identify the neighbouring leaf squares that share this edge, selecting the one that intersects the line $x_0 + t\nu$ at $t = t_{min}$.
4: Identify the edge of this square that intersects the line $x_0 + t\nu$ at $t = t_{min}$.
5: Consider the two endpoints y_1, y_2 of this edge as potential neighbours, one lying in the first quadrant and one in the fourth quadrant.
6: Repeat steps 2–5, continually adding nodes to the list of potential neighbours, until the ray $x_0 + t\nu$ exits the search region ($t > \epsilon$) or we encounter a boundary leaf square.
7: If the procedure terminates at a boundary leaf square, add to the list of potential neighbours all boundary nodes associated with this square.

From the list of potential neighbours in each quadrant, the precise neighbours used in the stencil are determined via (5). See Fig. 6, for a close-up of the neighbours search in a uniform (left) and non-uniform (right) mesh.

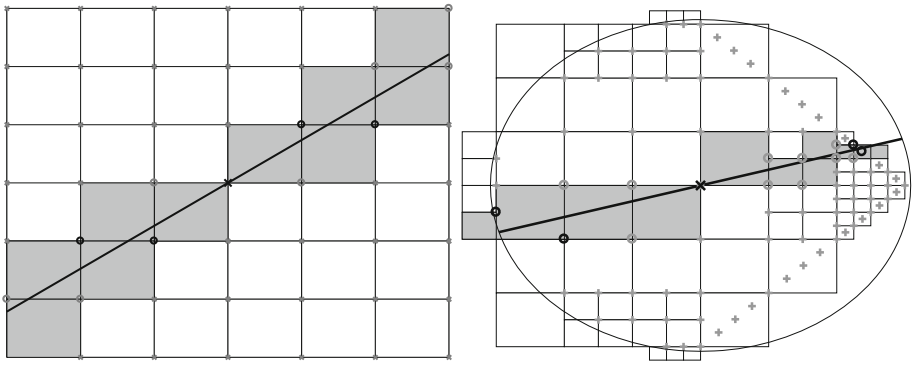

Fig. 6 Potential neighbours of $x_0 \in \mathcal{G}$ (black x-mark) as a result of Algorithm 2 are marked with a circle, with the selected neighbours in black. All remaining mesh points are marked with an x-mark. The grey squares are the ones considered in Algorithm 2

Referring to Fig. 6a, we provide a rough estimate on the improvement this algorithm yields for a uniform $N \times N$ grid with grid spacing $h = \mathcal{O}(1/N)$. Recall that the search region is a disc of radius ϵ. The brute force algorithm used in [20] examines $\mathcal{O}((\epsilon/h)^2)$ neighbours, while the algorithm proposed above using quadtrees examines only $\mathcal{O}(\epsilon/h)$ neighbours. Given the typical choice $\epsilon = \mathcal{O}(\sqrt{h})$, the cost of constructing the stencil at each point is reduced from $\mathcal{O}(N)$ to $\mathcal{O}(\sqrt{N})$. A similar speed-up is seen for the piecewise Cartesian meshes produced by the quadtree.

4 Higher-Order Methods

We also introduce a technique for building formally higher-order approximations on highly non-uniform/unstructured grids. We focus on second-order schemes, which is typically sufficient for applications, but the same ideas can be easily be extended to higher-order schemes.

4.1 Filtered Schemes

The meshfree finite difference approximation discussed in Sect. 2 is low accuracy; formally it is at best $\mathcal{O}(\sqrt{h})$. However it can be used as the foundation for higher-order convergent filtered schemes as in [21]. The main idea is to blend a monotone convergent scheme with a non-monotone accurate scheme and retain the advantages of both: stability and convergence of the former, and higher accuracy of the latter.

To accomplish this, we let $F_A[u]$ be any higher-order scheme and $F_M[u]$ be a monotone approximation scheme, both defined on the same mesh. The filtered scheme is then defined as

$$F_F[u] = F_M[u] + h^\alpha S\left(\frac{F_A[u] - F_M[u]}{h^\alpha}\right),$$

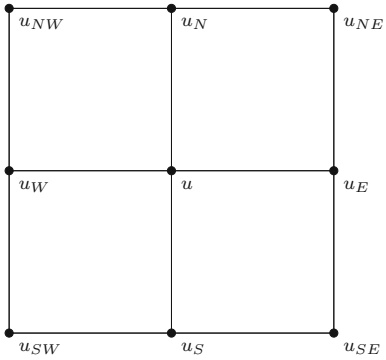

Fig. 7 A regular node and respective stencil for a uniform Cartesian grid

where the filter S is given by

$$S(x) = \begin{cases} x, & |x| \leq 1, \\ 0, & |x| \geq 2, \\ -x+2, & 1 \leq x \leq 2, \\ -x-2, & -2 \leq x \leq -1. \end{cases}$$

As long as $\alpha > 0$, this approximation converges to the viscosity solution of the PDE under the same conditions as the monotone scheme converges. The underlying reason is that this scheme is a small perturbation of a monotone scheme and the proof of the Barles-Souganidis theorem is easily modified to accommodate this. Moreover, if h^α is larger than the discretisation error of the monotone scheme, the formal accuracy of the filtered scheme is the same as the formal accuracy of the non-monotone scheme.

4.2 Higher-Order Schemes in Interior

We discuss how to build high-order schemes for the non-uniform meshes proposed in Sect. 3. In this section, we focus on interior mesh points away from the boundary.

Defining higher-order schemes for (1) reduces to defining higher-order approximations to u_{xx}, u_{yy} and u_{xy}. We will focus on building second order approximations, although the ideas are easily generalised. For a uniform Cartesian grid, such as in Fig. 7, these are widely known and are given by

$$u_{xx} \approx \frac{u_W + u_E - 2u}{h^2},$$
$$u_{yy} \approx \frac{u_N + u_S - 2u}{h^2},$$
$$u_{xy} \approx \frac{u_{NE} + u_{SW} - u_{NW} - u_{SE}}{4h^2}.$$

In a uniform cartesian grid, all the nodes are regular nodes; i.e., each node is the vertex of four different squares like the one depicted in Fig. 7. However, the meshes proposed here are non-uniform and in general not all nodes will be regular nodes. We need also consider dangling nodes, which occur midway along the shared edge of two equally-sized squares, one of which is subdivided. Thus additional work is required to define the higher-order schemes.

As explained in Sect. 3, the meshes are generated using quadtrees that are kept balanced (the lengths of neighbouring squares differ by at most a factor of two). Thus each interior

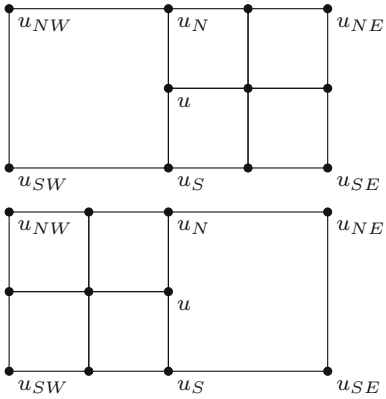

Fig. 8 A dangling node in the x variable and respective stencil for the higher-order scheme

mesh point can be associated to one of five different configurations. These are depicted in Figs. 7, 8, and 9. The generic element chosen to represent each configuration is one where each square is a leaf square. In general, one or more of the smaller squares may have children in the quadtree; i.e., they may be subdivided into smaller squares. We consider these to be redundant when constructing the high-order schemes. Considering all possible different configurations would only increase the complexity of the schemes with no additional benefits as the schemes would remain asymptotically second order; only the asymptotic error constant could be improved.

For the configurations in Fig. 8, we use the following approximations

$$\begin{aligned} u_{xx} &\approx \frac{-2u_N - 2u_S + u_{NW} + u_{NE} + u_{SE} + u_{SW}}{2h^2}, \\ u_{yy} &\approx 4\frac{u_N + u_S - 2u}{h^2}, \\ u_{xy} &\approx \frac{u_{NE} + u_{SW} - u_{NW} - u_{SE}}{2h^2}. \end{aligned}$$

As for the configurations in Fig. 9, we have

$$\begin{aligned} u_{xx} &\approx 4\frac{u_W + u_E - 2u}{h^2}, \\ u_{yy} &\approx \frac{-2u_W - 2u_E + u_{NW} + u_{NE} + u_{SE} + u_{SW}}{2h^2}, \\ u_{xy} &\approx \frac{u_{NE} + u_{SW} - u_{NW} - u_{SE}}{2h^2}. \end{aligned}$$

The standard Taylor expansion argument shows that the above expressions are second order accurate.

Finally, we explain how one can efficiently determine the configuration of each interior mesh point. As mentioned in Sect. 3.4, for each interior mesh point a record is kept of the four leaf squares that the interior mesh point as a vertex. Thus the configuration is easily determined by determining the depth of the neighbouring squares and respective parent squares in the quadtree.

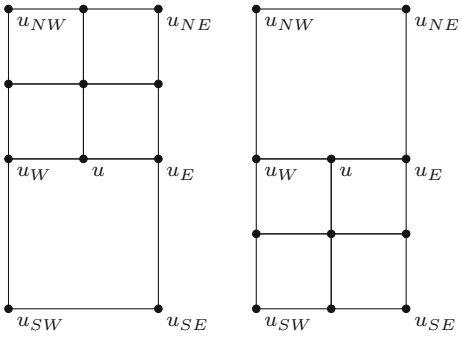

Fig. 9 A dangling node in the y variable and respective stencil for the higher-order scheme

4.3 Least-Squares Constructions Near Boundary

In this section, we discuss how to construct second order schemes at interior points near the boundary. When near the boundary, the construction of the schemes cannot reduce to the cases considered in the previous section: in general, not all the neighbouring mesh points will be the vertices of squares and some will lie on the boundary, which we allow to have a complicated geometry. Thus additional care is needed. Here we describe a general strategy for building high-order schemes.

Let $\{x_i\}_{i=1}^N$ denote neighbouring mesh points to the interior mesh point x_0 with $\|x_i - x_0\|_\infty = \mathcal{O}(h)$. Using Taylor expansion we obtain, for each $i = 1, \ldots, N$,

$$u(x_i) - u(x_0) = \sum_{0<|\alpha|\leq 3} \frac{(x_i - x_0)^\alpha}{\alpha!} (\partial^\alpha u)(x_0) + \mathcal{O}(h^4),$$

where we are using the multi-index notation. Hence

$$\sum_{i=1}^N a_i (u(x_i) - u(x_0)) = \partial^\beta u(x_0) + \mathcal{O}(h^2)$$

if the $\{a_i\}_{i=1}^N$ solve the linear system

$$\sum_{i=1}^N \frac{(x_i - x_0)^\alpha}{\alpha!} a_i = \mathbb{1}_{\{\alpha=\beta\}}$$

for $0 < |\alpha| \leq 3$.

To approximate second derivatives to second order, we expect to require $N = 9$ neighbours.

Designing second order schemes is now reduced to determining the neighbouring mesh points and solving the respective linear system. However, since we are particularly interested in the case where some of the neighbouring mesh points lie on the boundary, which may have a complicated geometry, it is hard to make any a priori claim regarding the invertability and conditioning of the linear system. It is important to point out that we are interested in obtaining any particular solution. As we saw in the previous section, depending on the derivative being approximated and the location of the neighbouring mesh points, the number of neighbouring mesh points required changes.

We are now ready to describe the strategy implemented to construct the higher-order schemes. First, we determine which configuration we are in. If all the vertices of squares

neighbouring x_0 are mesh points, we use the approximations described in the previous section. Otherwise, we build the linear system above using all mesh points x_i that lie within squares adjoining x_0 (some will lie on the boundary of the domain and will not be vertices of the squares). In general, we will have $N \geq 9$, but the linear system may still have no solution or be ill-conditioned. If that is the case, we consider additional neighbouring mesh points by adding the mesh points that belong to neighbouring squares. In general, we end up with an under-determined system and we select the least squares solution. In practice, adding additional neighbouring mesh points was not always required, and it was never required more than once. Thus the high-order scheme has stencil width $\mathcal{O}(h)$ and preserves the formal discretisation error of $\mathcal{O}(h^2)$.

5 Computational Examples

5.1 Monge–Ampère Equation

We consider the Monge–Ampère equation

$$\begin{cases} -\det(D^2u) + f = 0, & x \in \Omega \\ u = g, & x \in \partial\Omega \\ u \text{ is convex.} \end{cases}$$

The PDE is only elliptic in the space of convex functions. However, as in [18], we can use the globally elliptic extension

$$-\min_{\theta\in[0,\pi/2)} \left\{ \max\left\{ \frac{\partial^2 u}{\partial e_\theta^2}, 0 \right\} \max\left\{ \frac{\partial^2 u}{\partial e_{\theta+\pi/2}^2}, 0 \right\} + \min\left\{ \frac{\partial^2 u}{\partial e_\theta^2}, 0 \right\} + \max\left\{ \frac{\partial^2 u}{\partial e_{\theta+\pi/2}^2}, 0 \right\} \right\} + f = 0.$$

We will consider four different domains given by $\Omega = \{(x, y) \in \mathbb{R}^2 \mid \phi(x, y) < 0\}$ where ϕ is given by

(a) (circle) $\phi(x, y) = x^2 + y^2 - 1$,
(b) (ellipse) $\phi(x, y) = x^2 + 2y^2 - 1$,
(c) (diamond) $\phi(x, y) = |x| + |y| - 1$,
(d) (diamond stretched) $\phi(x, y) = |x| + |2y| - 1$.

Example 1 We consider first the following C^2 solution of the Monge–Ampère equation

$$u(x, y) = e^{\frac{x^2+y^2}{2}}, \quad f(x, y) = (1 + x^2 + y^2)e^{x^2+y^2}.$$

Results are displayed in Table 1. On each domain, the filtered implementation recovers the desired second-order accuracy even though the boundary nodes do not belong to the structured piecewise Cartesian mesh. The Monge–Ampère equation is intimately related to the computation of optimal transport maps. For these applications, it is the gradient of the solution that is of interest, rather than the potential function itself. Notice that due to the use of the quadtree structure and the way we deal we discretize the boundary, each interior node always has neighbouring nodes along the x and y directions. Therefore the gradient can be approximated with standard finite differences. In the particular case of the Monge–Ampère equation here, the mesh is uniform except near the boundary and thus the gradient

Table 1 Convergence results for the C^2 solution of the Monge–Ampère equation

N	h	Monotone		Filtered	
Max error and order, Example 1 (circle)					
101	2.500×10^{-1}	1.739×10^{-2}	–	7.008×10^{-3}	–
349	1.250×10^{-1}	5.678×10^{-3}	1.61	2.219×10^{-3}	1.66
1149	6.250×10^{-2}	2.781×10^{-3}	1.03	6.183×10^{-4}	1.84
4297	3.125×10^{-2}	1.876×10^{-3}	0.57	1.701×10^{-4}	1.86
15,741	1.563×10^{-2}	1.630×10^{-3}	0.20	4.320×10^{-5}	1.98
58,553	7.813×10^{-3}	1.566×10^{-3}	0.06	1.082×10^{-5}	2.00
230,725	3.906×10^{-3}	1.008×10^{-3}	0.64	2.704×10^{-6}	2.00
Max error and order, Example 1 (ellipse)					
79	2.500×10^{-1}	9.467×10^{-3}	–	8.905×10^{-3}	–
257	1.250×10^{-1}	3.028×10^{-3}	1.64	1.232×10^{-3}	2.85
901	6.250×10^{-2}	1.264×10^{-3}	1.26	3.627×10^{-4}	1.76
3151	3.125×10^{-2}	8.276×10^{-4}	0.61	1.012×10^{-4}	1.84
11,305	1.563×10^{-2}	7.443×10^{-4}	0.15	2.602×10^{-5}	1.96
42,947	7.813×10^{-3}	7.238×10^{-4}	0.04	6.512×10^{-6}	2.00
166,749	3.906×10^{-3}	4.554×10^{-4}	0.67	1.628×10^{-6}	2.00
Max error and order, Example 1 (diamond)					
57	2.500×10^{-1}	6.214×10^{-3}	–	7.245×10^{-3}	–
209	1.250×10^{-1}	2.677×10^{-3}	1.22	1.439×10^{-3}	2.33
673	6.250×10^{-2}	9.714×10^{-4}	1.46	3.561×10^{-4}	2.01
2497	3.125×10^{-2}	2.805×10^{-4}	1.79	8.881×10^{-5}	2.00
9345	1.563×10^{-2}	3.029×10^{-4}	− 0.11	2.218×10^{-5}	2.00
36,097	7.813×10^{-3}	1.903×10^{-4}	0.67	5.544×10^{-6}	2.00
139,777	3.906×10^{-3}	1.571×10^{-4}	0.28	1.386×10^{-6}	2.00
Max error and order, Example 1 (diamond stretched)					
61	2.500×10^{-1}	2.598×10^{-3}	–	1.892×10^{-3}	–
153	1.250×10^{-1}	8.006×10^{-4}	1.70	5.214×10^{-4}	1.86
497	6.250×10^{-2}	2.797×10^{-4}	1.52	1.356×10^{-4}	1.94
1633	3.125×10^{-2}	1.523×10^{-4}	0.88	3.428×10^{-5}	1.98
5569	1.563×10^{-2}	7.681×10^{-5}	0.99	8.597×10^{-6}	2.00
20,609	7.813×10^{-3}	2.360×10^{-5}	1.70	2.151×10^{-6}	2.00
76,033	3.906×10^{-3}	2.756×10^{-5}	− 0.22	5.379×10^{-7}	2.00

is approximated with second order accurate centred finite differences. In Table 2, we display the accuracy of the gradient and observe a superconvergence phenomenon. Despite the fact that we used a second-order scheme to solve the Monge–Ampère equation, this second-order accuracy is observed in the computed gradient of the solution as well as in the solution itself.

Example 2 We consider also a C^1 solution of the Monge–Ampère equation, for which the ellipticity is degenerate in an open set.

Table 2 Convergence results for the gradient of C^2 solution of the Monge–Ampère equation

N	h	Monotone		Filtered	
Max error in ∇u and order, Example 1 (circle)					
101	2.500×10^{-1}	1.032×10^{-2}	–	3.430×10^{-2}	–
349	1.250×10^{-1}	7.244×10^{-3}	0.51	1.213×10^{-2}	1.50
1149	6.250×10^{-2}	1.079×10^{-2}	− 0.58	6.447×10^{-3}	0.91
4297	3.125×10^{-2}	1.331×10^{-2}	− 0.30	1.223×10^{-3}	2.40
15,741	1.562×10^{-2}	1.393×10^{-2}	− 0.07	1.524×10^{-4}	3.00
58,553	7.812×10^{-3}	1.435×10^{-2}	− 0.04	3.587×10^{-5}	2.09
230,725	3.906×10^{-3}	7.137×10^{-3}	1.01	9.112×10^{-6}	1.98
Max error in ∇u and order, Example 1 (ellipse)					
79	2.500×10^{-1}	2.289×10^{-2}	–	3.247×10^{-2}	–
257	1.250×10^{-1}	6.595×10^{-3}	1.79	9.866×10^{-3}	1.72
901	6.250×10^{-2}	5.930×10^{-3}	0.15	6.864×10^{-3}	0.52
3151	3.125×10^{-2}	7.975×10^{-3}	− 0.43	9.158×10^{-4}	2.91
11,305	1.562×10^{-2}	8.658×10^{-3}	− 0.12	1.884×10^{-4}	2.28
42,947	7.812×10^{-3}	8.718×10^{-3}	− 0.01	4.586×10^{-5}	2.04
166,903	3.906×10^{-3}	4.819×10^{-3}	0.86	1.163×10^{-5}	1.98
Max error in ∇u and order, Example 1 (diamond)					
57	2.500×10^{-1}	2.598×10^{-2}	–	2.751×10^{-2}	–
177	1.250×10^{-1}	1.028×10^{-2}	1.34	1.082×10^{-2}	1.35
673	6.250×10^{-2}	3.029×10^{-3}	1.76	3.386×10^{-3}	1.68
2369	3.125×10^{-2}	2.908×10^{-3}	0.06	9.483×10^{-4}	1.84
8833	1.562×10^{-2}	3.253×10^{-3}	− 0.16	2.517×10^{-4}	1.91
34,561	7.812×10^{-3}	3.462×10^{-3}	− 0.09	6.491×10^{-5}	1.95
135,681	3.906×10^{-3}	2.130×10^{-3}	0.70	1.649×10^{-5}	1.98
Max error in ∇u and order, Example 1 (diamond stretched)					
45	2.500×10^{-1}	3.417×10^{-2}	–	3.501×10^{-2}	–
153	1.250×10^{-1}	1.228×10^{-2}	1.48	1.222×10^{-2}	1.52
433	6.250×10^{-2}	3.600×10^{-3}	1.77	3.618×10^{-3}	1.76
1377	3.125×10^{-2}	9.821×10^{-4}	1.87	9.852×10^{-4}	1.88
5057	1.562×10^{-2}	8.367×10^{-4}	0.23	2.571×10^{-4}	1.94
18,561	7.812×10^{-3}	8.703×10^{-4}	− 0.06	6.566×10^{-5}	1.97
70,401	3.906×10^{-3}	6.430×10^{-4}	0.44	1.659×10^{-5}	1.98

$$u(x, y) = \frac{1}{2} \max\left\{\sqrt{x^2 + y^2} - 0.2, 0\right\}^2, \quad f(x, y) = \max\left\{1 - \frac{0.2}{\sqrt{x^2 + y^2)}}, 0\right\}.$$

This solution is singular, and there is thus no realistic hope of attaining the formal second-order discretisation error obtained from Taylor's Theorem. Nevertheless, the method converges and we observe roughly first-order accuracy (Table 3). The reduced accuracy of the estimated gradients (Table 4) is to be expected since the solution itself does not have

Table 3 Convergence results for the C^1 solution of the Monge–Ampère equation

N	h	Monotone		Filtered	
Max error and order, Example 2 (circle)					
101	2.500×10^{-1}	1.606×10^{-2}	–	4.143×10^{-3}	–
349	1.250×10^{-1}	9.103×10^{-3}	0.82	2.384×10^{-3}	0.80
1149	6.250×10^{-2}	5.706×10^{-3}	0.67	1.733×10^{-3}	0.46
4297	3.125×10^{-2}	4.827×10^{-3}	0.24	7.823×10^{-4}	1.15
15,741	1.563×10^{-2}	4.330×10^{-3}	0.16	3.702×10^{-4}	1.08
58,553	7.813×10^{-3}	4.300×10^{-3}	0.01	1.838×10^{-4}	1.01
230,725	3.906×10^{-3}	2.207×10^{-3}	0.96	9.571×10^{-5}	0.94
Max error and order, Example 2 (ellipse)					
79	2.500×10^{-1}	1.193×10^{-2}	–	4.672×10^{-3}	–
257	1.250×10^{-1}	7.126×10^{-3}	0.74	3.180×10^{-3}	0.56
901	6.250×10^{-2}	4.569×10^{-3}	0.64	1.579×10^{-3}	1.01
3151	3.125×10^{-2}	3.924×10^{-3}	0.22	8.300×10^{-4}	0.93
11,305	1.563×10^{-2}	3.519×10^{-3}	0.16	3.747×10^{-4}	1.15
42,947	7.813×10^{-3}	3.497×10^{-3}	0.01	1.741×10^{-4}	1.11
166,749	3.906×10^{-3}	1.774×10^{-3}	0.98	8.052×10^{-5}	1.11
Max error and order, Example 2 (diamond)					
57	2.500×10^{-1}	8.503×10^{-3}	–	6.576×10^{-3}	–
209	1.250×10^{-1}	6.706×10^{-3}	0.34	2.168×10^{-3}	1.60
673	6.250×10^{-2}	4.265×10^{-3}	0.65	1.555×10^{-3}	0.48
2497	3.125×10^{-2}	3.441×10^{-3}	0.31	7.661×10^{-4}	1.02
9345	1.563×10^{-2}	2.303×10^{-3}	0.58	3.622×10^{-4}	1.08
36,097	7.813×10^{-3}	1.109×10^{-3}	1.05	2.296×10^{-4}	0.66
139,777	3.906×10^{-3}	1.007×10^{-3}	0.14	7.695×10^{-5}	1.58
Max error and order, Example 2 (diamond stretched)					
61	2.500×10^{-1}	2.696×10^{-3}	–	4.108×10^{-3}	–
153	1.250×10^{-1}	3.781×10^{-3}	− 0.49	3.026×10^{-3}	0.44
497	6.250×10^{-2}	2.426×10^{-3}	0.64	1.213×10^{-3}	1.32
1633	3.125×10^{-2}	1.820×10^{-3}	0.41	1.233×10^{-3}	− 0.02
5569	1.563×10^{-2}	1.203×10^{-3}	0.60	3.663×10^{-4}	1.75
20,609	7.813×10^{-3}	6.089×10^{-4}	0.98	2.028×10^{-4}	0.85
76,033	3.906×10^{-3}	5.692×10^{-4}	0.10	9.177×10^{-5}	1.14

sufficient accuracy to justify the Taylor expansions used in either the discretisation of the PDE or the approximation of the gradient.

5.2 Pucci Equation

We also demonstrate that our method works on more complicated domains. To do this, we consider the Pucci equation

Table 4 Convergence results for the gradient of C^1 solution of the Monge–Ampère equation

N	h	Monotone		Filtered	
Max error in ∇u and order, Example 2 (circle)					
101	2.500×10^{-1}	2.861×10^{-2}	–	3.465×10^{-2}	–
349	1.250×10^{-1}	1.097×10^{-2}	1.38	1.514×10^{-2}	1.19
1149	6.250×10^{-2}	9.979×10^{-3}	0.14	9.984×10^{-3}	0.60
4297	3.125×10^{-2}	7.942×10^{-3}	0.33	4.185×10^{-3}	1.25
15,741	1.562×10^{-2}	7.232×10^{-3}	0.14	3.058×10^{-3}	0.45
58,553	7.812×10^{-3}	7.305×10^{-3}	− 0.01	3.135×10^{-3}	− 0.04
230,725	3.906×10^{-3}	3.635×10^{-3}	1.01	2.229×10^{-3}	0.49
Max error in ∇u and order, Example 2 (ellipse)					
79	2.500×10^{-1}	3.029×10^{-2}	–	3.716×10^{-2}	–
257	1.250×10^{-1}	1.327×10^{-2}	1.19	9.992×10^{-3}	1.89
901	6.250×10^{-2}	1.038×10^{-2}	0.35	1.123×10^{-2}	− 0.17
3151	3.125×10^{-2}	8.408×10^{-3}	0.30	4.787×10^{-3}	1.23
11,305	1.562×10^{-2}	7.688×10^{-3}	0.13	4.179×10^{-3}	0.20
42,947	7.812×10^{-3}	7.724×10^{-3}	− 0.01	3.081×10^{-3}	0.44
166,903	3.906×10^{-3}	3.859×10^{-3}	1.00	2.142×10^{-3}	0.52
Max error in ∇u and order, Example 2 (diamond)					
57	2.500×10^{-1}	2.907×10^{-2}	–	3.419×10^{-2}	–
177	1.250×10^{-1}	1.097×10^{-2}	1.41	1.513×10^{-2}	1.18
673	6.250×10^{-2}	9.985×10^{-3}	0.14	9.960×10^{-3}	0.60
2369	3.125×10^{-2}	7.907×10^{-3}	0.34	7.322×10^{-3}	0.44
8833	1.562×10^{-2}	7.206×10^{-3}	0.13	3.465×10^{-3}	1.08
34,561	7.812×10^{-3}	7.287×10^{-3}	− 0.02	3.747×10^{-3}	− 0.11
135,681	3.906×10^{-3}	3.617×10^{-3}	1.01	3.234×10^{-3}	0.21
Max error in ∇u and order, Example 2 (diamond stretched)					
45	2.500×10^{-1}	3.364×10^{-2}	–	3.364×10^{-2}	–
153	1.250×10^{-1}	1.135×10^{-2}	1.57	1.063×10^{-2}	1.66
433	6.250×10^{-2}	1.090×10^{-2}	0.06	1.134×10^{-2}	− 0.09
1377	3.125×10^{-2}	9.230×10^{-3}	0.24	5.322×10^{-3}	1.09
5057	1.562×10^{-2}	8.918×10^{-3}	0.05	6.393×10^{-3}	− 0.26
18,561	7.812×10^{-3}	8.971×10^{-3}	− 0.01	4.360×10^{-3}	0.55
70,401	3.906×10^{-3}	4.238×10^{-3}	1.08	2.733×10^{-3}	0.67

$$\begin{cases} \alpha\lambda_+(D^2u) + \lambda_-(D^2u) = 0, & x \in \Omega, \\ u = g, & x \in \partial\Omega, \end{cases}$$

where

$$\lambda_+(D^2u) = \max_{\theta\in[0,2\pi]} \frac{\partial^2 u}{\partial e_\theta^2} \quad \text{and} \quad \lambda_-(D^2u) = \min_{\theta\in[0,2\pi]} \frac{\partial^2 u}{\partial e_\theta^2}.$$

For $\alpha > 0$, the PDE is elliptic.

We will solve this in the irregular, non-convex, non-smooth domain in Fig. 4b, which can be defined as $\Omega = \{(x, y) \in \mathbb{R}^2 \mid \phi(x, y) < 0\}$ where ϕ is given by

$$\phi(x, y) = \min\left\{c_+^{(1)}(x, y), c_-^{(1)}(x, y), c_+^{(2)}(x, y), c_-^{(2)}(x, y)\right\}$$

with

$$c_\pm^{(1)}(x, y) = \left(x \pm \frac{1}{2}\right)^2 + 5\left(y \pm \frac{1}{4}\right)^2 - \frac{1}{2} \quad \text{and}$$
$$c_\pm^{(2)}(x, y) = 5\left(x \pm \frac{1}{4}\right)^2 + \left(y \mp \frac{1}{4}\right)^2 - \frac{1}{2}.$$

In order to define a filtered scheme for the Pucci equation, we first need to define an accurate scheme for it. Notice that, based on the work discussed in Sect. 4, it is enough to rewrite the Pucci equation in terms of the Hessian entries. This can be accomplished in the following way. For $\theta \in [0, 2\pi]$,

$$u_{\theta\theta} = \cos^2(\theta)u_{xx} + \sin^2(\theta)u_{yy} + 2\sin(\theta)\cos(\theta)u_{xy}$$

and so

$$\frac{d}{d\theta}u_{\theta\theta} = -\sin(2\theta)(u_{xx} - u_{yy}) + 2\cos(2\theta)u_{xy}.$$

Hence

$$\frac{d}{d\theta}u_{\theta\theta} = 0 \iff \tan(2\theta^*) = \frac{2u_{xy}}{u_{xx} - u_{yy}} \implies \theta^* = \frac{1}{2}\arctan\left(\frac{2u_{xy}}{u_{xx} - u_{yy}}\right).$$

Then

$$\lambda_+(D^2u) = \max\left\{\frac{\partial^2 u}{\partial e_{\theta^*}^2}, \frac{\partial^2 u}{\partial e_{\theta^*+\frac{\pi}{2}}^2}\right\} \quad \text{and} \quad \lambda_-(D^2u) = \min\left\{\frac{\partial^2 u}{\partial e_{\theta^*}^2}, \frac{\partial^2 u}{\partial e_{\theta^*+\frac{\pi}{2}}^2}\right\}.$$

Example 3 We consider the radial solution of the Pucci equation

$$u(x, y) = -\rho^{1-\alpha}, \quad \text{where } \rho(x, y) = \sqrt{(x+2)^2 + (y+2)^2}$$

and $\alpha = 3$.

Results are displayed in Table 5. The results are similar to Example 1 of the Monge–Ampère equation: despite the irregular, non-convex, non-smooth domain, we observe second-order accuracy for the filtered scheme.

5.3 Computation Time

One of the reasons to use quadtrees to build non-uniform meshes was to efficiently find the neighbours for the finite difference schemes. Here we compare the implementation with quadtrees, discussed in depth in Sect. 3, with the simple but inefficient brute force approach of [20]. Reduced CPU time is the ultimate goal, but not necessarily a robust measure of efficiency as it depends on both hardware and software implementations. Here both implementations were vectorised whenever possible in order to optimise the code for MATLAB.

Table 5 Convergence results for the radial solution of the Pucci equation

N	h	Monotone		Filtered	
Max error and order, Example 3 (fan-shape)					
93	3.750×10^{-1}	7.007×10^{-3}	–	2.158×10^{-3}	–
435	1.875×10^{-1}	2.456×10^{-3}	1.51	1.673×10^{-4}	3.69
711	9.375×10^{-2}	2.939×10^{-3}	− 0.26	8.033×10^{-4}	− 2.26
1853	4.688×10^{-2}	5.189×10^{-4}	2.50	9.827×10^{-6}	6.35
6759	2.344×10^{-2}	1.010×10^{-3}	− 0.96	2.336×10^{-6}	2.07
30,215	1.172×10^{-2}	7.123×10^{-4}	0.50	5.658×10^{-7}	2.05
84,951	5.859×10^{-3}	2.212×10^{-4}	1.69	1.415×10^{-7}	2.00

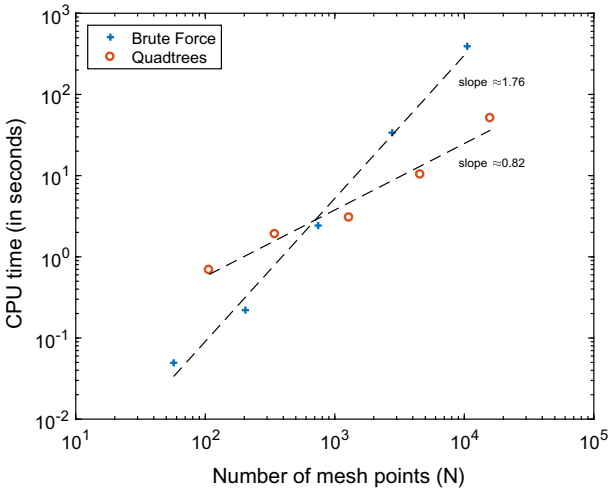

Fig. 10 Number of mesh points versus CPU time in seconds to generate a mesh and find the respective stencil for an ellipsoidal domain

Although it is still possible to further optimise the code, the CPU time should be a fair indication of the improvement gained by using quadtrees.

In Fig. 10, we compare the number of mesh points versus the CPU time required to generate the mesh and find the stencil. We let the domain Ω be the ellipse given by $\Omega = \{(x, y) \in \mathbb{R}^2 \mid x^2 + 2y^2 < 1\}$. The results indicate that our new approach is optimal, with the computation time required to construct the stencils being roughly proportional to the number of mesh points. This represents an improvement of roughly one order over the original brute force approach. In practice, we can generate a mesh and find the stencil based on a uniform 256×256 grid in roughly 50 s with the use of quadtrees, instead of over 6 minutes with the previous brute force approach.

5.4 Adaptivity

In our next example, we demonstrate the improvements possible with adaptivity using our generalised finite difference approximations.

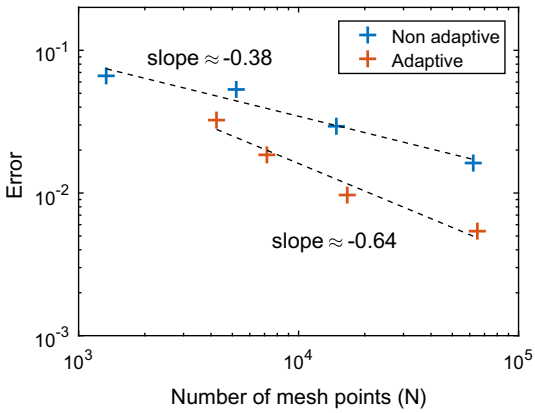

Fig. 11 Number of mesh points vs error for Example 4

Example 4 We consider the fully nonlinear convex envelope equation

$$\begin{cases} \max\left\{-\lambda_-(D^2u), u-g\right\} = 0, & x \in \Omega \\ u = 0.5, & x \in \partial\Omega, \end{cases}$$

where

$$\lambda_-(D^2u) = \min_{\theta\in[0,2\pi]} \frac{\partial^2 u}{\partial e_\theta^2}.$$

The equation is posed on an ellipse with semi-major axis equal to one and semi-minor axis equal to one-half, which is rotated through an angle of $\phi = \pi/6$. The obstacle g consists of two cones,

$$\begin{aligned} g_1(x,y) &= (x\cos\phi + y\sin\phi + 0.5)^2 + (-x\sin\phi + y\cos\phi)^2 \\ g_2(x,y) &= (x\cos\phi + y\sin\phi - 0.5)^2 + (-x\sin\phi + y\cos\phi)^2 \\ g(x,y) &= \min\{g_1(x,y), g_2(x,y), 0.5\} \end{aligned}$$

and the exact solution is

$$u(x,y) = \begin{cases} \min\{g_1(x,y), g_2(x,y)\}, & |x\cos\phi + y\sin\phi| \geq 0.5 \\ |-x\sin\phi + y\cos\phi|, & |x\cos\phi + y\sin\phi| < 0.5. \end{cases}$$

We note that this solution is only Lipschitz continuous, and the equation must be understood in a weak sense.

We defined the following refinement strategy. Given a solution a solution u^h,

1. Compute numerically $\left\|D^2u\right\| = \sqrt{u_{xx}^2 + 2u_{xy}^2 + u_{yy}^2}$.
2. Refine twice every square with a vertex such that $h\left\|D^2u\right\| > 0.5$.

This refinement strategy is inspired by the hybrid scheme proposed in [25].

A convergence plot is displayed in Fig. 11. We observe an improvement in accuracy from roughly $\mathcal{O}(N^{-0.38})$ to $\mathcal{O}(N^{-0.64})$. Moreover, the use of adaptivity leads to a clear qualitative improvement in the computed solutions. This is evident in Fig. 12, which shows that the solution obtained with the uniform mesh is visibly non-convex along the singularity, which does not align with the grid. By resolving this singularity, the adaptive method produces a solution that has a dramatically better quality.

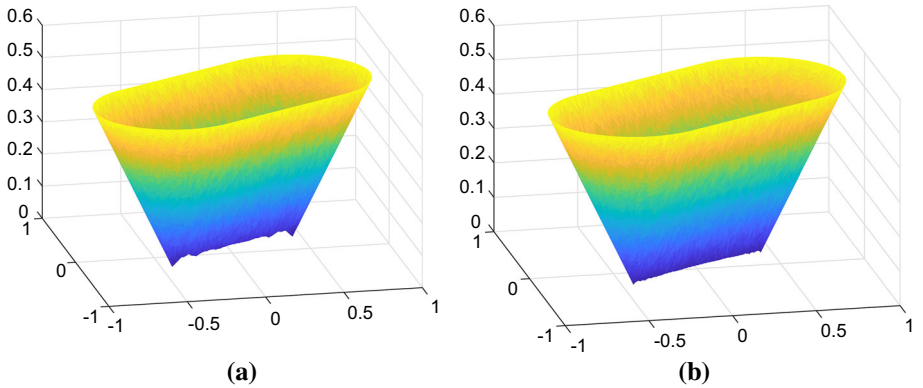

Fig. 12 Solutions of the convex envelope equation computed with the non-adaptive (left) and adaptive approaches (right)

6 Conclusions

In this article, we described generalised finite difference methods for solving a large class of fully nonlinear elliptic partial differential equations. These methods were inspired by the meshfree methods described in [20], which are flexible and convergent, but very expensive to implement.

Our meshes used a modified quadtree structure that used piecewise Cartesian grids in the interior of the domain, augmented by a set of points lying exactly on the boundary. The inclusion of the additional boundary points was needed to ensure convergence of the numerical methods. This type of mesh also allows us to deal easily with complicated boundary geometries.

By relying on the underlying quadtree structure, we developed an algorithm for efficiently constructing the mesh and finite difference stencils. This led to a dramatic improvement in computational efficiency as compared to the original brute force approach.

We also described a strategy for constructing higher-order approximations, which still fit within the convergence framework. In the interior of the domain, the quadtree structure allows us to explicitly write out the higher-order finite difference schemes. This strategy fails near the boundary, which can have a very complicated geometry. At these points, we employed a least-squares approach to construct higher-order schemes.

We also used these methods to perform automatic mesh adaptation, which refines the mesh near singularities in the computed solutions. This led to a dramatic qualitative and quantitative improvement in computed solutions.

Acknowledgements We thank Adam Oberman for helpful discussions and support of this project.

References

1. Awanou, G.: Standard finite elements for the numerical resolution of the elliptic Monge–Ampère equation: classical solutions. IMA J. Numer. Anal. **35**(3), 1150–1166 (2015)
2. Barles, G., Souganidis, P.E.: Convergence of approximation schemes for fully nonlinear second order equations. Asymptot. Anal. **4**(3), 271–283 (1991)

3. Benamou, J.D., Froese, B.D., Oberman, A.M.: Two numerical methods for the elliptic Monge–Ampère equation. M2AN Math. Model. Numer. Anal. **44**(4), 737–758 (2010). doi:10.1051/m2an/2010017
4. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry, 3rd edn. Springer, Berlin (2008). doi:10.1007/978-3-540-77974-2. **(Algorithms and applications)**
5. Böhmer, K.: On finite element methods for fully nonlinear elliptic equations of second order. SIAM J. Numer. Anal. **46**(3), 1212–1249 (2008)
6. Brenner, S.C., Gudi, T., Neilan, M., Sung, L.Y.: C^0 penalty methods for the fully nonlinear Monge–Ampère equation. Math. Comput. **80**(276), 1979–1995 (2011)
7. Budd, C.J., Williams, J.F.: Moving mesh generation using the parabolic Monge–Ampère equation. SIAM J. Sci. Comput. **31**(5), 3438–3465 (2009). doi:10.1137/080716773
8. Caffarelli, L.A., Milman, M.: Monge Ampère Equation: Applications to Geometry and Optimization: NSF-CBMS Conference on the Monge Ampère Equation, Applications to Geometry and Optimization, July 9–13, 1997, Florida Atlantic University, vol. 226. American Mathematical Society (1999)
9. Crandall, M.G., Ishii, H., Lions, P.L.: User's guide to viscosity solutions of second order partial differential equations. Bull. Am. Math. Soc. (N.S.) **27**(1), 1–67 (1992)
10. Cullen, M.J.P., Norbury, J., Purser, R.J.: Generalised Lagrangian solutions for atmospheric and oceanic flows. SIAM J. Appl. Math. **51**(1), 20–31 (1991). doi:10.1137/0151002
11. Dean, E.J., Glowinski, R.: Numerical methods for fully nonlinear elliptic equations of the Monge–Ampère type. Comput. Methods Appl. Mech. Eng. **195**(13–16), 1344–1386 (2006)
12. Engquist, B., Froese, B.D.: Application of the Wasserstein metric to seismic signals. Commun. Math. Sci. **12**(5), 979–988 (2014)
13. Evans, L.C.: Classical solutions of fully nonlinear, convex, second-order elliptic equations. Commun. Pure Appl. Math. **35**(3), 333–363 (1982)
14. Feng, X., Neilan, M.: Vanishing moment method and moment solutions for fully nonlinear second order partial differential equations. J. Sci. Comput. **38**(1), 74–98 (2009)
15. Finn, J.M., Delzanno, G.L., Chacón, L.: Grid generation and adaptation by Monge–Kantorovich optimization in two and three dimensions. In: Proceedings of the 17th International Meshing Roundtable, pp. 551–568 (2008). doi:10.1007/978-3-540-87921-3_33
16. Fleming, W.H., Soner, H.M.: Controlled Markov Processes and Viscosity Solutions, vol. 25. Springer, Berlin (2006)
17. Frisch, U., Matarrese, S., Mohayaee, R., Sobolevski, A.: A reconstruction of the initial conditions of the universe by optimal mass transportation. Nature **417**, 260–262 (2002). doi:10.1038/417260a
18. Froese, B.D.: A numerical method for the elliptic Monge–Ampère equation with transport boundary conditions. SIAM J. Sci. Comput. **34**(3), A1432–A1459 (2012)
19. Froese, B.D.: Convergence approximation of non-continuous surfaces of prescribed Gaussian curvature. Communications on Pure and Applied Analysis. https://arxiv.org/pdf/1601.06315.pdf **(to appear)**
20. Froese, B.D.: Meshfree finite difference approximations for functions of the eigenvalues of the Hessian. Numer. Math. (2017). doi:10.1007/s00211-017-0898-2
21. Froese, B.D., Oberman, A.M.: Convergent filtered schemes for the Monge–Ampère partial differential equation. SIAM J. Numer. Anal. **51**(1), 423–444 (2013)
22. Glimm, T., Oliker, V.: Optical design of single reflector systems and the Monge–Kantorovich mass transfer problem. J. Math. Sci. (N. Y.) **117**(3), 4096–4108 (2003). doi:10.1023/A:1024856201493. **(Nonlinear problems and function theory)**
23. Loeper, G., Rapetti, F.: Numerical solution of the Monge–Ampère equation by a Newton's algorithm. C. R. Math. Acad. Sci. Paris **340**(4), 319–324 (2005). doi:10.1016/j.crma.2004.12.018
24. Oberman, A.M.: Convergent difference schemes for degenerate elliptic and parabolic equations: Hamilton–Jacobi equations and free boundary problems. SIAM J. Numer. Anal. **44**(2), 879–895 (2006)
25. Oberman, A.M.: Computing the convex envelope using a nonlinear partial differential equation. Math. Models Methods Appl. Sci. **18**(5), 759–780 (2008)
26. Oberman, A.M.: 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**(1), 221–238 (2008). doi:10.3934/dcdsb.2008.10.221
27. Osher, S., Paragios, N.: Geometric Level Set Methods in Imaging, Vision, and Graphics. Springer, Berlin (2003)
28. Saumier, L.P., Agueh, M., Khouider, B.: An efficient numerical algorithm for the L2 optimal transport problem with periodic densities. IMA J. Appl. Math. **80**(1), 135–157 (2015)
29. Smears, I., Suli, E.: Discontinuous Galerkin finite element approximation of Hamilton–Jacobi–Bellman equations with Cordes coefficients. SIAM J. Numer. Anal. **52**(2), 993–1016 (2014)
30. Sulman, M., Williams, J.F., Russell, R.D.: Optimal mass transport for higher dimensional adaptive grid generation. J. Comput. Phys. **230**(9), 3302–3330 (2011). doi:10.1016/j.jcp.2011.01.025