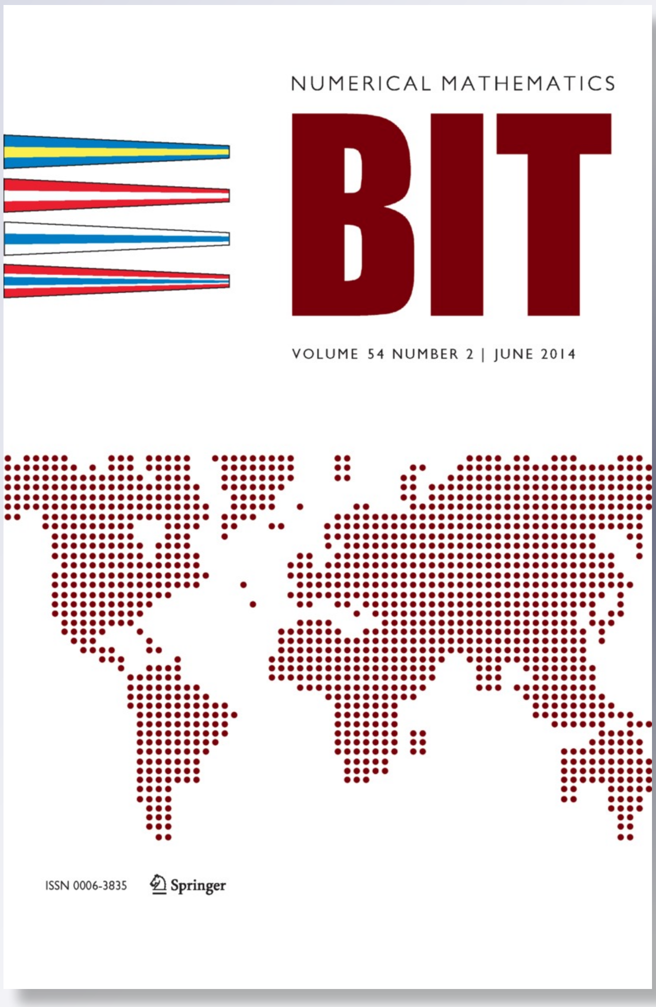
# A new iterative algorithm for mean curvature-based variational image denoising

Li Sun & Ke Chen

**BIT Numerical Mathematics**

ISSN 0006-3835
Volume 54
Number 2

Bit Numer Math (2014) 54:523-553
DOI 10.1007/s10543-013-0448-y



BIT Numer Math (2014) 54:523 – 553
DOI 10.1007/s10543-013-0448-y

# A new iterative algorithm for mean curvature-based variational image denoising

**Li Sun · Ke Chen**

Received: 11 September 2012 / Accepted: 21 September 2013 / Published online: 10 October 2013

**Abstract** The total variation semi-norm based model by Rudin-Osher-Fatemi (in Physica D 60, 259–268, 1992) has been widely used for image denoising due to its ability to preserve sharp edges. One drawback of this model is the so-called staircasing effect that is seen in restoration of smooth images. Recently several models have been proposed to overcome the problem. The mean curvature-based model by Zhu and Chan (in SIAM J. Imaging Sci. 5(1), 1–32, 2012) is one such model which is known to be effective for restoring both smooth and nonsmooth images. It is, however, extremely challenging to solve efficiently, and the existing methods are slow or become efficient only with strong assumptions on the formulation; the latter includes Brito-Chen (SIAM J. Imaging Sci. 3(3), 363–389, 2010) and Tai et al. (SIAM J. Imaging Sci. 4(1), 313–344, 2011).

Here we propose a new and general numerical algorithm for solving the mean curvature model which is based on an augmented Lagrangian formulation with a special linearised fixed point iteration and a nonlinear multigrid method. The algorithm improves on Brito-Chen (SIAM J. Imaging Sci. 3(3), 363–389, 2010) and Tai et al. (SIAM J. Imaging Sci. 4(1), 313–344, 2011). Although the idea of an augmented Lagrange method has been used in other contexts, both the treatment of the boundary conditions and the subsequent algorithms require careful analysis as standard

Communicated by Rosemary Renaut.

This work is supported by the Fundamental Research Funds for the Central Universities (lzujbky-2013-15).

L. Sun
School of Mathematics and Statistics, Lanzhou University, Lanzhou, China
e-mail: sunli@lzu.edu.cn

K. Chen (✉)
Center for Mathematical Imaging Techniques and Department of Mathematical Sciences, The University of Liverpool, Liverpool L69 7ZL, UK
e-mail: k.chen@liv.ac.uk
url: http://www.liv.ac.uk/cmit

BIT Numer Math (2014) 54:523 – 553
DOI 10.1007/s10543-013-0448-y

approaches do not work well. After constructing two fixed point methods, we analyze their smoothing properties and use them for developing a converging multigrid method. Finally numerical experiments are conducted to illustrate the advantages by comparing with other related algorithms and to test the effectiveness of the proposed algorithms.

**Keywords** Staircasing effect · Denoising · Mean curvature-based model · Fixed point iteration method · Nonlinear multigrid

**Mathematics Subject Classification (2010)** 68U10 · 65F10 · 65K10

## 1 Introduction

The presence of noise in an image is common, often arising from the image formation process such as image recording or transmission. Denoising is perhaps the most fundamental image processing task which has been deeply investigated for many years. In this paper, we mainly consider a model for removing additive, zero-mean Gaussian type noise and focus on effective numerical algorithms. Let $\Omega$ be a bounded and open domain in $\mathbb{R}^2$ (without loss of generality we assume $\Omega$ is a square), $f$ the observed image and $u$ the true image. The degradation model can be expressed as:

$$f(x,y) = u(x,y) + n(x,y), \quad (x,y) \in \Omega \tag{1.1}$$

which $n$ is the unknown additive noise (see [8] for an example of other noise models). It is well-known that $u$ cannot be obtained uniquely from $f$. Rather, regularisation is required in order to find an approximation to $u$, yielding the model

$$\min_u \left\{ J(u) = \frac{1}{2}\int_\Omega (u-f)^2 dxdy + \lambda R(u) \right\}, \tag{1.2}$$

where the first fitting term ensures that $u$ is close to $f$, the second term $R(u)$ is a regularizer designed based on a priori information about $u$, and $\lambda > 0$ is a regularisation parameter. The Total Variation (TV) regularizer $R(u) = \int_\Omega |\nabla u|\, dxdy$ proposed by Rudin, Osher and Fatemi [18] leads to the TV model. The Euler-Lagrange (EL) equation for (1.2) is

$$-\lambda \nabla \cdot \frac{\nabla u}{|\nabla u|} + (u - f) = 0, \tag{1.3}$$

where the Neumann boundary condition $\nabla u \cdot \boldsymbol{\nu} = 0$ is imposed with $\boldsymbol{\nu}$ the unit outward normal vector. In practice the term $|\nabla u|$ is replaced by $|\nabla u|_\beta = \sqrt{|\nabla u|^2 + \beta}$ to avoid division by zero, where $\beta > 0$ is a small parameter. This model can preserve shape edges and contours; for smooth images, however, the TV model produces undesirable staircasing effects in transforming a smooth function into a piecewise constant function.

In order to remedy this drawback, Blomgren et al. proposed to use a different regularizer $R(u) = \int_\Omega |\nabla u|^{P(|\nabla u|)}\, dxdy$ [1] (with $1 \le P \le 2$) instead of the TV seminorm where $P = 1$. The split Bregman (SB) iteration has attracted a lot of attention in image denoising and deblurring [12, 17]. The fundamental idea is to transform a constrained optimization problem to a series of unconstrained problems by introducing auxiliary variables. In each unconstrained problem, the objective function is defined by the Bregman distance for a convex functional. This method converges rapidly and visually reduces the effects of staircasing. The non-local means model by [11, 15] also provides effective staircase reduction by defining a non-local TV (involving all first order variations) in a neighborhood of each pixel. Recently many researchers have turned to higher order models. The total generalized variation (TGV) model [2] uses a semi-norm defined by a nontrivial mixing of first and second order variations to improve on the TV model. Here we are concerned with the particular model using mean curvature as a regularizer, shown to be effective for denoising by Zhu and Chan [28] and Brito and Chen [4]. The resulting EL partial differential equation (PDE) is highly nonlinear and of fourth order. The method uses a gradient descent algorithm and is consequently slow to converge. The stabilization method adopted by [4] was a major improvement but it has severe limitation in regularising the nonlinearity. Below we first review the mean curvature-based model and the existing numerical algorithms. Then we present our new algorithm.

The contributions of the paper include the following: (i) our work improves on Brito-Chen [4] in better treatment of the nonlinearity; (ii) it improves on the staircasing effect of the TV model; (iii) it provides a fast realisation of the mean curvature-based model; and (iv) it improves on the augmented Lagrangian implementation (AL) following Tai et al. [20] in restoration quality. Although SB, AL and TGV all outperform the TV, the improvements by AL and TGV are more than by the SB while the TGV is the best of the three methods. The proposed algorithm will perform similarly to the TGV and slightly better than it in some cases.

*Mean curvature-based model* The mean curvature-based model of Zhu and Chan [28] and Lysaker-Osher-Tai [16]

$$\min_u \left\{ J(u) = \frac{1}{2}\int_\Omega (u-f)^2\,dxdy + \lambda \int_\Omega \Phi(\kappa)\,dxdy \right\}, \tag{1.4}$$

is known to be better than the TV model in image denoising [4, 16, 28], where $\kappa(u)$ is the mean curvature of the image defined as $\kappa(u) = \nabla \cdot (\nabla u/|\nabla u|)$. Here $\Phi(\kappa) = \kappa^2$ or $\Phi(\kappa) = |\kappa|$ or is a combination of both. Below we take $\Phi(\kappa) = \kappa^2$, so $\Phi'(\kappa) = 2\kappa$. Note that in [28] an image is understood as a surface represented by $(x, y, z)$ where $z = u(x, y)$. In that case $\kappa(u) = \nabla \cdot (\nabla u/\sqrt{|\nabla u|^2 + 1})$. Here the more common form $\kappa(u) = \kappa_\beta(u) = \nabla \cdot (\nabla u/\sqrt{|\nabla u|^2 + \beta})$ is adopted [16]. $\beta$ is a small parameter introduced to mitigate potential instability as $|\nabla u|$ becomes small. In fact smaller $\beta$ corresponds to stronger nonlinearity and slows down the convergence of many numerical methods. We are interested in obtaining a numerical algorithm that converges even for small $\beta$; note that previous methods simply do not work for small $\beta$.

The EL equation for (1.4) is the following

$$\lambda \nabla \cdot \left( \frac{\nabla \Phi'(\kappa)}{|\nabla u|} - \frac{\nabla u \cdot \nabla \Phi'(\kappa)}{(|\nabla u|)^3} \nabla u \right) + u - f = 0 \quad \text{in } \Omega \tag{1.5}$$

with boundary conditions $\nabla u \cdot \boldsymbol{\nu} = 0$, $\kappa = 0$ on $\partial\Omega$ and $\boldsymbol{\nu}$ is the unit outward normal vector. Model (1.4) is equivalent to finding a piecewise smooth surface to approximate the image surface [28]. It can remove noise efficiently and, at the same time, maintains corners, edges and grey scale intensity contrast. A time marching (gradient descent) method [28] can be applied to the parabolic form of (1.5):

$$\frac{\partial u}{\partial t} = \lambda \nabla \cdot \left( \frac{\nabla \Phi'(\kappa)}{|\nabla u|} - \frac{\nabla u \cdot \nabla \Phi'(\kappa)}{(|\nabla u|)^3} \nabla u \right) + u - f \tag{1.6}$$

with initial condition $u(x, y, 0) = f(x, y)$. This method converges very slowly due to the stability restrictions on the time step, $\Delta t \sim O((\Delta x)^4)$. An alternative Augmented Lagrangian method for (1.4) with $\beta = 1$ was suggested in [29].

Brito-Chen [4] developed two new algorithms for solving (1.5): a stabilized fixed point method and, based upon this, an efficient nonlinear multigrid (MG) method. In developing their fixed point method, they found that various fixed point schemes do not converge. However after a stabilizing term $\gamma N$ is added to both sides of (1.5), they obtained a converging fixed point method

$$-\gamma N_{i,j}^{(k+1)} - \lambda \nabla \cdot \left( D_2(u)_{i,j}^{(k+1)} (\nabla u)_{i,j}^{(k+1)} \right) + u_{i,j}^{(k+1)} = g_{i,j}^{(k)}, \tag{1.7}$$

where

$$D_1 = \frac{1}{|\nabla u|_\beta}, \qquad D_2 = \frac{\nabla u \cdot \nabla \Phi'}{|\nabla u|_\beta^3},$$

$$g_{i,j}^{(k)} = f_{i,j} - \lambda \nabla \cdot \left( D_1(u)_{i,j}^{(k)} (\nabla \Phi')_{i,j}^{(k)} \right) - \gamma N_{i,j}^{(k)},$$

and $N = \mathrm{TV}(u) = \nabla \cdot (\nabla u / |\nabla u|)$ provided that $\beta$ is large enough (e.g. $\beta \geq 10^{-2}$). This method is called the stabilized fixed point (SFP) method. It has been proven to be very efficient as a smoother for a nonlinear MG by local Fourier analysis, as long as $\beta$ is not too small. It is our intention to develop a new algorithm that does not impose such a strong assumption on $\beta$.

The rest of the paper is organised as follows. Section 2 introduces our iterative method for solving (1.4) on a single grid. Section 3 uses the developed iterative method as a smoother for a nonlinear multigrid method. Since previous methods do not converge for small $\beta$, our tests will focus on cases of small $\beta$. Section 4 gives numerical experiments, contrasting the performance of the proposed algorithms as compared to other approaches in the literature. As expected, our method does converge for the mean curvature model with small parameter $\beta$ and it shows some degree of robustness with respect to the choice of $\beta$.

## 2 A new iterative method for a mean curvature model

In this section, we present a new augmented Lagrangian method related to the mean curvature-based model (1.4). As we shall see, though simple and efficient, our new algorithm will be as accurate as [4, 28] and more importantly more robust with respect to $\beta$ than [4]. In order to solve (1.4), we change it into a constrained minimization problem by introducing a new auxiliary variable $\omega = (\omega_1, \omega_2)$ and by using an operator splitting technique. The auxiliary variable permits the derivation of a system of second order PDEs in contrast to the fourth order PDE obtained without the auxiliary variable.

Then with $\omega$, the constrained minimization problem can be expressed as:

$$\begin{aligned} &\min_{u,\omega}\left\{ E(u,\omega) = \int_\Omega (u-f)^2 dxdy + \lambda \int_\Omega (\nabla\cdot\omega)^2 dxdy \right\} \\ &\text{s.t.,}\quad \omega = \frac{\nabla u}{|\nabla u|_\beta}. \end{aligned} \tag{2.1}$$

The constraint condition can be reformulated as $\nabla u - \omega|\nabla u|_\beta = 0$, yielding the quadratic penalty method for (1.4)

$$\begin{aligned} \min_{u,\omega}\Big\{ E(u,\omega) = \int_\Omega (u-f)^2 dxdy + \lambda \int_\Omega (\nabla\cdot\omega)^2 dxdy \\ + \gamma \int_\Omega \left\| \nabla u - \omega |\nabla u|_\beta \right\|^2 dxdy \Big\}, \end{aligned} \tag{2.2}$$

where $\|\cdot\|$ denotes $L_2$ norm, and $\lambda$ and $\gamma$ are positive regularisation parameters. We first derive the formal EL equations for (2.2).

## 2.1 The EL equations

We shall assume the image to be reconstructed is smooth enough, such that $u, \omega \in W^{1,1}(\Omega) = \{u \in L^1(\Omega) : D^1 u \in L^1(\Omega)\}$, and $\nabla\cdot\omega$ and $\nabla u$ are well defined.

**Lemma 1** *The Euler-Lagrange equations for the functional* (2.2) *are*:

$$\begin{cases} u - f - \gamma\nabla\cdot\left(\nabla u - |\nabla u|_\beta\omega - \dfrac{\nabla u\cdot\omega}{|\nabla u|_\beta}\nabla u + (\omega\cdot\omega)\nabla u\right) = 0 \\ -\gamma|\nabla u|_\beta \nabla u - \lambda\nabla(\nabla\cdot\omega) + \gamma|\nabla u|_\beta^2\omega = 0, \end{cases} \tag{2.3}$$

*where the boundary conditions are* $\nabla\cdot\omega = 0$, $\nabla u\cdot\boldsymbol{\nu} = 0$, *and* $\omega\cdot\boldsymbol{\nu} = 0$.

*Proof* Let $v, \phi_1, \phi_2 : \mathbb{R}^2 \longrightarrow \mathbb{R} \in \mathbf{C}_c^1(\Omega)$ (first-order continuous functions). Taking $\phi = (\phi_1, \phi_2)$, from (2.2), we have

$$\begin{aligned} \delta E(u,\omega) &= \frac{d}{d\epsilon} E(u+\epsilon v, \omega+\epsilon\phi)\Big|_{\epsilon=0} \\ &= \frac{d}{d\epsilon} \underbrace{\left( \int_\Omega (u+\epsilon v - f)^2 dxdy + \lambda \int_\Omega (\nabla\cdot\omega + \epsilon\nabla\cdot\phi)^2 dxdy \right)\Big|_{\epsilon=0}}_{I_1} \\ &\quad + \frac{d}{d\epsilon} \underbrace{\gamma \int_\Omega \left\| (\nabla u + \epsilon\nabla v) - (\omega+\epsilon\phi)|(\nabla u + \epsilon\nabla v)|_\beta \right\|^2 dxdy \Big|_{\epsilon=0}}_{I_2} = 0. \end{aligned}$$

We now simplify $I_1, I_2$ respectively as

$$\begin{aligned}I_1 &= 2\int_\Omega (u-f)v\,dxdy + 2\lambda\int_\Omega \nabla\cdot\omega\nabla\cdot\phi\,dxdy\\ &= 2\int_\Omega (u-f)v\,dxdy - 2\lambda\int_\Omega \nabla(\nabla\cdot\omega)\cdot\phi\,dxdy + 2\lambda\int_\Gamma (\nabla\cdot\omega)(\boldsymbol{\nu}\cdot\phi)\,dS\\ &= 2\int_\Omega (u-f)v\,dxdy - 2\lambda\int_\Omega \nabla(\nabla\cdot\omega)\cdot\phi\,dxdy\end{aligned}$$

(after we impose the first boundary condition $\nabla\cdot\omega = 0$) and

$$\begin{aligned}I_2 = 2\gamma&\int_\Omega (\nabla u\nabla v - |\nabla u|_\beta\omega\cdot\nabla v - |\nabla u|_\beta\nabla u\cdot\phi)dxdy\\ &+ 2\gamma\int_\Omega\left(|\nabla u|_\beta^2\omega\cdot\phi - (\nabla u\cdot\omega)\frac{\nabla u\cdot\nabla v}{|\nabla u|_\beta} + (\omega\cdot\omega)\nabla u\cdot\nabla v\right)dxdy.\end{aligned}$$

Imposing boundary conditions $\nabla u\cdot\boldsymbol{\nu} = 0$ and $\omega\cdot\boldsymbol{\nu} = 0$, Green's first formula simplifies $I_2$ as

$$\begin{aligned}I_2 = -2\gamma&\int_\Omega \nabla\cdot\left(\nabla u - |\nabla u|_\beta\omega - \frac{\nabla u\cdot\omega}{|\nabla u|_\beta}\nabla u + (\omega\cdot\omega)\nabla u\right)v\,dxdy\\ &+ 2\gamma\int_\Omega\left(|\nabla u|_\beta^2\omega - |\nabla u|_\beta\nabla u\right)\cdot\phi\,dxdy.\end{aligned}$$

Then from $I_1 + I_2 = 0$, we obtain the EL equations (2.3). □

Since $\omega = (\omega_1, \omega_2)$, we can rewrite (2.3) as three coupled partial differential equations

$$\begin{cases} u - f - \gamma\nabla\cdot\left(\nabla u - |\nabla u|_\beta\omega - \dfrac{\nabla u\cdot\omega}{|\nabla u|_\beta}\nabla u + (\omega\cdot\omega)\nabla u\right) = 0,\\ -\gamma|\nabla u|_\beta u_x - \lambda\partial_x(\nabla\cdot\omega) + \gamma|\nabla u|_\beta^2\omega_1 = 0,\\ -\gamma|\nabla u|_\beta u_y - \lambda\partial_y(\nabla\cdot\omega) + \gamma|\nabla u|_\beta^2\omega_2 = 0.\end{cases} \tag{2.4}$$

Next we briefly discuss the discretisation of (2.4) and then present an efficient solver.

## 2.2 Numerical discretization

Without loss of generality, we assume $\Omega$ is approximated by a discrete domain $\Omega^h = \{(x_i, y_j)\in\Omega\ |x_i = ih_x - 1/2,\ y_j = jh_y - 1/2,\ i = 1,2,\ldots,m, j = 1,2,\ldots,n\}$, where $h_x,\ h_y$ are the mesh sizes in $x, y$ directions respectively, and $u^h = u^h_{i,j} = u^h(x_i, y_j)$ is the discrete form of $u$ defined on $\Omega^h$. For simplicity we will take $m = n$,

$h_x = h_y = h$. To approximate the term $\nabla \cdot V = (V_1)_x + (V_2)_y$ at pixel $(i, j)$ for any $V = (V_1, V_2)$, central differences can be used as follows:

$$(\nabla \cdot V)_{i,j} = \frac{(V_1)_{i+\frac{1}{2},j} - (V_1)_{i-\frac{1}{2},j}}{h} + \frac{(V_2)_{i,j+\frac{1}{2}} - (V_2)_{i,j-\frac{1}{2}}}{h}. \tag{2.5}$$

*Partial derivatives* in $x$ are given by the central differencing of two adjacent *whole* pixels as

$$\begin{aligned}
(u_x)_{i+\frac{1}{2},j} &= (u_{i+1,j} - u_{i,j})/h, \\
(u_x)_{i-\frac{1}{2},j} &= (u_{i,j} - u_{i-1,j})/h, \\
\partial_x(\nabla \cdot \omega)_{i+\frac{1}{2},j} &= \frac{(\omega_1)_{i-\frac{1}{2},j} + (\omega_1)_{i+\frac{1}{2}+1,j} - 2(\omega_1)_{i+\frac{1}{2},j}}{h^2} \\
&\quad + \frac{(\omega_2)_{i+1,j+\frac{1}{2}} + (\omega_2)_{i,j-\frac{1}{2}} - (\omega_2)_{i,j+\frac{1}{2}} - (\omega_2)_{i+1,j-\frac{1}{2}}}{h^2}.
\end{aligned}$$

*Partial derivatives* in $y$ are given by the min-mod at two adjacent whole pixels defined by

$$\begin{aligned}
(u_y)_{i+\frac{1}{2},j} &= \text{min-mod}\left(\frac{1}{2h}(u_{i+1,j+1} - u_{i+1,j-1}), \frac{1}{2h}(u_{i,j+1} - u_{i,j-1})\right), \\
(u_y)_{i-\frac{1}{2},j} &= \text{min-mod}\left(\frac{1}{2h}(u_{i,j+1} - u_{i,j-1}), \frac{1}{2h}(u_{i-1,j+1} - u_{i-1,j-1})\right).
\end{aligned}$$

Here, as noted in [18], the min-mod functions defined as follows help with recovery of sharp edges

$$\text{min-mod}(a, b) = \frac{\text{sgn}(a) + \text{sgn}(b)}{2}\min\bigl(|a|, |b|\bigr), \quad \text{sgn}(x) = \begin{cases} -1 & x < 0, \\ 0 & x = 0, \\ 1 & x > 0. \end{cases} \tag{2.6}$$

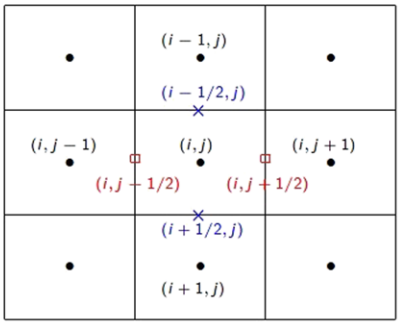
The values for $(u_y)_{i,j+1/2}$, $(u_y)_{i,j+1/2}$ are obtained similarly and

$$\begin{aligned}
\partial_y(\nabla \cdot \omega)_{i,j+\frac{1}{2}} &= \frac{(\omega_2)_{i,j-\frac{1}{2}} + (\omega_2)_{i,j+\frac{1}{2}+1} - 2(\omega_2)_{i,j+\frac{1}{2}}}{h^2} \\
&\quad + \frac{(\omega_1)_{i+\frac{1}{2},j+1} + (\omega_1)_{i-\frac{1}{2},j} - (\omega_1)_{i+\frac{1}{2},j} - (\omega_1)_{i-\frac{1}{2},j+1}}{h^2}.
\end{aligned}$$

Then

$$\begin{aligned}
|\nabla u|_{i+\frac{1}{2},j} &= \sqrt{(u_x)^2_{i+\frac{1}{2},j} + (u_y)^2_{i+\frac{1}{2},j} + \beta}, \\
|\nabla u|_{i,j+\frac{1}{2}} &= \sqrt{(u_x)^2_{i,j+\frac{1}{2}} + (u_y)^2_{i,j+\frac{1}{2}} + \beta},
\end{aligned}$$

**Fig. 1** This is a staggered grid, $u_{i,j}$ are defined on $\bullet$-nodes, $(\omega_1)_{i+\frac{1}{2},j}$ are defined on $\times$-nodes and $(\omega_2)_{i,j+\frac{1}{2}}$ are defined on $\square$-nodes



and the Neumann boundary condition on $\partial\Omega$ leads to

$$u_{i,0}=u_{i,1},\qquad u_{i,n+1}=u_{i,n},\qquad u_{0,j}=u_{1,j},\qquad u_{m+1,j}=u_{m,j}.$$

Finally, the discretization of (2.4) is obtained as

$$\begin{cases} u_{i,j}-\gamma(\nabla\cdot V)_{i,j}+\gamma\nabla\cdot\left(|\nabla u|_\beta\omega\right)_{i,j}=f_{i,j}, & (2.7a)\\ -\gamma\left(|\nabla u|_\beta u_x\right)_{i+\frac{1}{2},j}-\lambda\partial_x(\nabla\cdot\omega)_{i+\frac{1}{2},j}+\gamma\left(|\nabla u|_\beta^2\omega_1\right)_{i+\frac{1}{2},j}=0, & (2.7b)\\ -\gamma\left(|\nabla u|_\beta u_y\right)_{i,j+\frac{1}{2}}-\lambda\partial_y(\nabla\cdot\omega)_{i,j+\frac{1}{2}}+\gamma\left(|\nabla u|_\beta^2\omega_2\right)_{i,j+\frac{1}{2}}=0, & (2.7c)\end{cases}$$

where $V=\nabla u-(\nabla u\cdot\omega/|\nabla u|_\beta)\nabla u+(\omega\cdot\omega)\nabla u$.

In the above equations, we note that at each pixel $(i,j)$, not all of the three unknowns are available. In particular, as depicted in Fig. 1, unknowns $u_{i,j}$ are defined on $\bullet$-nodes, but $(\omega_1)_{i+1/2,j}$ and $(\omega_2)_{i,j+1/2}$ are defined on $\times$-nodes and $\square$-nodes, respectively. Although average operators can be used to calculate values of $\omega_1$ at $\square$-nodes and $\omega_2$ at $\times$-nodes, that is:

$$\begin{cases} (\omega_1^{\square})_{i,j+\frac{1}{2}}=\dfrac{(\omega_1)_{i+\frac{1}{2},j}+(\omega_1)_{i+\frac{1}{2},j+1}+(\omega_1)_{i-\frac{1}{2},j}+(\omega_1)_{i-\frac{1}{2},j+1}}{4}, & (2.8a)\\ (\omega_2^{\times})_{i+\frac{1}{2},j}=\dfrac{(\omega_2)_{i,j+\frac{1}{2}}+(\omega_2)_{i,j-\frac{1}{2}}+(\omega_2)_{i+1,j+\frac{1}{2}}+(\omega_2)_{i+1,j-\frac{1}{2}}}{4}, & (2.8b)\end{cases}$$

these will introduce additional errors and more importantly satisfying the boundary conditions will be less straightforward. An alternative is to use a staggered grid.

## 2.3 Discretization on a staggered grid

To maintain coupling between $u$ and $\omega$, it is convenient to use a staggered grid system as shown in Fig. 1. Then in (2.7a)–(2.7c), all three unknown quantities in a computational box are directly available. Now we consider how to handle the boundary conditions on the staggered grid.

On the boundaries we have $(\omega_1)_{1-1/2,j}=(\omega_1)_{m+1/2,j}=0$, $j=1,2,\dots,n$ and $(\omega_2)_{i,1-1/2}=(\omega_2)_{i,n+1/2}=0$, $i=1,2,\dots,m$. The values of $(\omega_2)_{1-1/2,j}$ and

$(\omega_2)_{m+1/2,j}$ can be obtained from (2.7c) discretised on points $(1-1/2, j)$ and $(m+1/2, j)$, that is

$$(\omega_2)_{i+\frac{1}{2},j} = \left(\frac{u_y}{|\nabla u|}\right)_{i+\frac{1}{2},j} = \left(\frac{u_y}{|u_y|}\right)_{i+\frac{1}{2},j} = \pm 1$$

from

$$\begin{aligned} &-\gamma\big(|\nabla u| u_y\big)_{i+\frac{1}{2},j} - \lambda\big(\partial_y(\nabla\cdot\omega)\big)_{i+\frac{1}{2},j} + \gamma\big(|\nabla u|^2\omega_2\big)_{i+\frac{1}{2},j} \\ &\quad = -\gamma\big(|\nabla u| u_y\big)_{i+\frac{1}{2},j} + \gamma\big(|\nabla u|^2\omega_2\big)_{i+\frac{1}{2},j} = 0, \quad i = 0, m \end{aligned}$$

(when $|\nabla u| = 0$, $|\nabla u|_\beta \neq 0$, so $(\omega_2)_{i+1/2,j} = 0$). Similarly,

$$(\omega_1)_{i,1-\frac{1}{2}} = (\omega_1)_{i,n+\frac{1}{2}} = \pm 1.$$

To find points near the boundaries we apply the exact boundary condition $\nabla\cdot\omega = 0$. At the top boundary ($i = 1$) and bottom boundary ($i = m$), $(\omega)_{i,j+1/2}$ is obtained from (2.7c) using $i = 1, m$, respectively, instead of from (2.8a). Likewise at the left and right boundaries, $j = 1, n$, respectively, $(\omega)_{i+1/2,j}$ is obtained from (2.7b) instead of from (2.8b).

Therefore, the staggered grid system provides us a convenient way of satisfying the boundary conditions for $u, \omega$.

## 2.4 Numerical algorithms on a single grid

We now present two algorithms for solving (2.7a)–(2.7c) on a single grid before describing their use in the context of a nonlinear multigrid method.

Newton's method is a natural algorithm to solve nonlinear equations such as (2.7a)–(2.7c), but convergence is not assured if starting from the initial guess $u^{(0)} = f$. Unfortunately, experiments demonstrate that a simple fixed point scheme such as

$$\begin{cases} u_{i,j}^{(k+1)} - \gamma\big(\nabla\cdot V^{(k+1)}\big)_{i,j} + \gamma\big(\nabla\cdot\big|\nabla u^{(k+1)}\big|_\beta\omega^{(k+1)}\big)_{i,j} = f_{i,j}, \\ -\gamma\big(\big|\nabla u^{(k+1)}\big|_\beta u_x^{(k+1)}\big)_{i+\frac{1}{2},j} - \lambda\partial_x\big(\nabla\cdot\omega^{(k+1)}\big)_{i+\frac{1}{2},j} \\ \quad + \gamma\big(|\nabla u^{(k+1)}|_\beta^2\omega_1^{(k+1)}\big)_{i+\frac{1}{2},j} = 0, \\ -\gamma\big(\big|\nabla u^{(k+1)}\big|_\beta u_y^{(k+1)}\big)_{i,j+\frac{1}{2}} - \lambda\partial_y\big(\nabla\cdot\omega^{(k+1)}\big)_{i,j+\frac{1}{2}} \\ \quad + \gamma\big(|\nabla u^{(k+1)}|_\beta^2\omega_2^{(k+1)}\big)_{i,j+\frac{1}{2}} = 0, \end{cases} \tag{2.9}$$

is neither stable nor convergent (unless $\beta$ is set large to reduce singularities). Based on a convexity-splitting idea by Yuille and Rangarajan [26] and Eyre [9, 10], we propose instead a stable numerical method for (2.7a)–(2.7c) given by

$$\begin{cases} u_{i,j} - \gamma\big(\nabla\cdot D(\omega)\nabla u\big)_{i,j} = (f_1)_{i,j}, \\ -\gamma\big(|\nabla u|_\beta u_x\big)_{i+\frac{1}{2},j} - \lambda\partial_x(\nabla\cdot\omega)_{i+\frac{1}{2},j} + \gamma\big(|\nabla u|_\beta^2\omega_1\big)_{i+\frac{1}{2},j} = 0, \\ -\gamma\big(|\nabla u|_\beta u_y\big)_{i,j+\frac{1}{2}} - \lambda\partial_y(\nabla\cdot\omega)_{i,j+\frac{1}{2}} + \gamma\big(|\nabla u|_\beta^2\omega_2\big)_{i,j+\frac{1}{2}} = 0, \end{cases} \tag{2.10}$$

where $D(\omega) = 1 + (\omega\cdot\omega)$, and $(f_1)_{i,j} = f_{i,j} - \gamma(\nabla\cdot(\nabla u\cdot\omega/|\nabla u|_\beta)\nabla u + |\nabla u|_\beta\omega)_{i,j}$. In order to address linearisation in the nonlinear system (2.10), we consider the following two methods:

(1) *The global Gauss-Seidel.* By 'global', we mean to linearise (2.10) by freezing all coefficients and right hand side terms at all pixels for outer iteration $k$, yielding the linear equations for update $k+1$ given by

$$\begin{cases} u_{i,j}^{(k+1)} - \gamma\big(\nabla\cdot D\big(\omega^{(k)}\big)\nabla u^{(k+1)}\big)_{i,j} = \big(f_1^k\big)_{i,j}, \\ -\gamma\big(\big|\nabla u^{(k)}\big|_\beta u_x^{(k+1)}\big)_{i+\frac{1}{2},j} - \lambda\partial_x\big(\nabla\cdot\omega^{(k+1)}\big)_{i+\frac{1}{2},j} \\ \quad + \gamma\big(\big|\nabla u^{(k)}\big|_\beta^2\omega_1^{(k+1)}\big)_{i+\frac{1}{2},j} = 0, \\ -\gamma\big(\big|\nabla u^{(k)}\big|_\beta u_y^{(k+1)}\big)_{i,j+\frac{1}{2}} - \lambda\partial_y\big(\nabla\cdot\omega^{(k+1)}\big)_{i,j+\frac{1}{2}} \\ \quad + \gamma\big(\big|\nabla u^{(k)}\big|_\beta^2\omega_2^{(k+1)}\big)_{i,j+\frac{1}{2}} = 0. \end{cases} \tag{2.11}$$

The linear equations (2.11) can be solved efficiently by a wide range of linear iterative methods, such as the Jacobi or Gauss-Seidel methods. Here we use the Gauss-Seidel (GS) iteration, hence our terminology that this is a 'global GS' (GGS) method. Let $p$ denote the inner iteration of the GS solver, then superscript $(k, p)$ denotes outer iteration $k$ and inner iteration $p$. The inner iteration update to level $p+1$ is given in matrix form as

$$\begin{aligned} &\begin{pmatrix} S_{i,j}^{(k)} & 0 & 0 \\ \frac{\gamma}{h}(|\nabla u^{(k)}|_\beta)_{i+\frac{1}{2},j} & \frac{2\lambda}{h^2} + \gamma(|\nabla u^{(k)}|_\beta^2)_{i+\frac{1}{2},j} & \frac{\lambda}{h^2} \\ \frac{\gamma}{h}(|\nabla u^{(k)}|_\beta)_{i,j+\frac{1}{2}} & \frac{\lambda}{h^2} & \frac{2\lambda}{h^2} + \gamma(|\nabla u^{(k)}|_\beta^2)_{i,j+\frac{1}{2}} \end{pmatrix} \\ &\quad\times \begin{pmatrix} u_{i,j}^{(k,p+1)} \\ (\omega_1)_{i+\frac{1}{2},j}^{(k,p+1)} \\ (\omega_2)_{i,j+\frac{1}{2}}^{(k,p+1)} \end{pmatrix} \\ &\quad= \begin{pmatrix} (g_1)_{i,j} \\ (g_2)_{i+\frac{1}{2},j} \\ (g_3)_{i,j+\frac{1}{2}} \end{pmatrix} \end{aligned} \tag{2.12}$$

where

$$\begin{cases} (g_1)_{i,j} = \left(f_1^k\right)_{i,j} + C^{(k)}_{i+\frac{1}{2},j} u^{(k,p)}_{i+1,j} + C^{(k)}_{i-\frac{1}{2},j} u^{(k,p+1)}_{i-1,j} + C^{(k)}_{i,j+\frac{1}{2}} u^{(k,p)}_{i,j+1} \\ \qquad + C^{(k)}_{i,j-\frac{1}{2}} u^{(k,p+1)}_{i,j-1} \\ (g_2)_{i+\frac{1}{2},j} = \frac{\gamma}{h} |\nabla u^{(k)}|_{i+\frac{1}{2},j} u^{(k,p)}_{i+1,j} + \frac{\lambda}{h^2}\big((\omega_1)^{(k,p+1)}_{i-\frac{1}{2},j} + (\omega_1)^{(k,p)}_{i+1+\frac{1}{2},j} \\ \qquad + (\omega_2)^{(k,p)}_{i+1,j+\frac{1}{2}} + (\omega_2)^{(k,p+1)}_{i,j-\frac{1}{2}} - (\omega_2)^{(k,p+1)}_{i+1,j-\frac{1}{2}}\big) \\ (g_3)_{i,j+\frac{1}{2}} = \frac{\gamma}{h} |\nabla u^{(k)}|_{i,j+\frac{1}{2}} u^{(k,p)}_{i,j+1} + \frac{\lambda}{h^2}\big((\omega_2)^{(k,p+1)}_{i,j-\frac{1}{2}} + (\omega_2)^{(k,p)}_{i,j+1+\frac{1}{2}} \\ \qquad + (\omega_1)^{(k,p)}_{i+\frac{1}{2},j+1} + (\omega_1)^{(k,p+1)}_{i-\frac{1}{2},j} - (\omega_1)^{(k,p)}_{i-\frac{1}{2},j+1}\big). \end{cases} \tag{2.13}$$

Here $S^{(k)}_{i,j} = 1 + C^{(k)}_{i+1/2,j} + C^{(k)}_{i-1/2,j} + C^{(k)}_{i,j+1/2} + C^{(k)}_{i,j-1/2}$, $C^{(k)}_{i+1/2,j} = D(\omega^{(k)})_{i+1/2,j}/h^2$, and $C^{(k)}_{i-1/2,j}$, $C^{(k)}_{i,j+1/2}$ and $C^{(k)}_{i,j-1/2}$ are defined similarly. Note that when calculating $u^{(k,p+1)}_{i,j}$, $(\omega_1)^{(k,p+1)}_{i+1/2,j}$, $(\omega_2)^{(k,p+1)}_{i,j+1/2}$, the updated $u^{(k,p+1)}_{i-1,j}$, $u^{(k,p+1)}_{i,j-1}$, $(\omega_1)^{(k,p+1)}_{i-1/2,j}$, $(\omega_2)^{(k,p+1)}_{i,j-1/2}$ and $(\omega_2)^{(k,p+1)}_{i+1,j-1/2}$ are used, hence the use of GS instead of the Jacobi to describe the process. Algorithm 1 describes the overall GGS solver.

*Remark* An alternative approach could be to use a global Jacobi method where the nonlinear coefficients are frozen globally followed by local Jacobi iterations. Our tests have shown that while this method works equally well for large $\beta$ (e.g. $\beta \geq 10^{-2}$), it fails to converge for $\beta < 10^{-2}$. Therefore if one is content with large $\beta$ and parallel implementation is necessary, this global Jacobi method is feasible. Since our purpose in this paper is to demonstrate the advantages of being able to

**Algorithm 1** $\mathbf{z}_h \leftarrow$ GGS ($u, \omega, \lambda, \gamma, \beta, \xi$ and $\zeta$), where $u$ and $\omega$ are initial values, $\lambda$ and $\gamma$ are regularisation parameters, $\beta$ is the stabilizing parameter, $\xi$ is the number of outer iterations and $\zeta$ is the number of inner iterations

```
1: Initialization: $u^0$, $\omega^0$, $\lambda$, $\gamma$, $\beta$, $\xi$ and $\zeta$.
2: For $k = 1$ to $\xi$
3:     compute $C^{(k)}_{i+\frac{1}{2},j}$, $C^{(k)}_{i-\frac{1}{2},j}$, $C^{(k)}_{i,j+\frac{1}{2}}$, $C^{(k)}_{i,j-\frac{1}{2}}$ and $(f_1^k)_{i,j}$,
4:     For $p = 1$ to $\zeta$
5:        For $i = 1, \ldots m$, $j = 1, \ldots n$
6:           compute $u^{(k,p+1)}_{i,j}$, $(\omega_1)^{(k,p+1)}_{i+\frac{1}{2},j}$ and $(\omega_2)^{(k,p+1)}_{i,j+\frac{1}{2}}$ from (2.11).
7:        end, end
8:     $u^{(k+1)}_{i,j} = u^{(k,\zeta)}_{i,j}$, $(\omega_1)^{(k+1)}_{i+\frac{1}{2},j} = (\omega_1)^{(k,\zeta)}_{i+\frac{1}{2},j}$, $(\omega_2)^{(k+1)}_{i,j+\frac{1}{2}} = (\omega_2)^{(k,\zeta)}_{i,j+\frac{1}{2}}$
9: end
10: end
```

solve the mean curvature model with a small $\beta$, we shall not discuss this method further.

(2) *Local Gauss-Seidel method*. As an alternative to the above GGS, we now consider linearising the nonlinear system (2.10) only locally at each pixel. So the frozen coefficients will be updated in a Gauss-Seidel fashion. We can rewrite (2.7a)–(2.7c) as

$$\begin{cases} N_1\big(\dots, u_{i-1,j}^{(k+1)}, (\omega_1)_{i-\frac{1}{2},j}^{(k+1)}, (\omega_2)_{i-1,j+\frac{1}{2}}^{(k+1)}, u_{i,j}^{(k+1)}, (\omega_1)_{i+\frac{1}{2},j}^{(k+1)}, \\ \quad (\omega_2)_{i,j+\frac{1}{2}}^{(k+1)}, u_{i+1,j}^{(k)}, (\omega_1)_{i+1+\frac{1}{2},j}^{(k)}, (\omega_2)_{i+1,j+\frac{1}{2}}^{(k)}, \dots\big) = 0, \\ N_2\big(\dots, u_{i-1,j}^{(k+1)}, (\omega_1)_{i-\frac{1}{2},j}^{(k+1)}, (\omega_2)_{i-1,j+\frac{1}{2}}^{(k+1)}, u_{i,j}^{(k+1)}, (\omega_1)_{i+\frac{1}{2},j}^{(k+1)}, \\ \quad (\omega_2)_{i,j+\frac{1}{2}}^{(k+1)}, u_{i+1,j}^{(k)}, (\omega_1)_{i+1+\frac{1}{2},j}^{(k)}, (\omega_2)_{i+1,j+\frac{1}{2}}^{(k)}, \dots\big) = 0, \\ N_3\big(\dots, u_{i-1,j}^{(k+1)}, (\omega_1)_{i-\frac{1}{2},j}^{(k+1)}, (\omega_2)_{i-1,j+\frac{1}{2}}^{(k+1)}, u_{i,j}^{(k+1)}, (\omega_1)_{i+\frac{1}{2},j}^{(k+1)}, \\ \quad (\omega_2)_{i,j+\frac{1}{2}}^{(k+1)}, u_{i+1,j}^{(k)}, (\omega_1)_{i+1+\frac{1}{2},j}^{(k)}, (\omega_2)_{i+1,j+\frac{1}{2}}^{(k)}, \dots\big) = 0. \end{cases} \tag{2.14}$$

Then to obtain the three local quantities $u_{i,j}^{(k+1)}$, $(\omega_1^{(k+1)})_{i+1/2,j}$ and $(\omega_2^{(k+1)})_{i,j+1/2}$ in the above nonlinear system

$$\begin{cases} N_1\big(\dots, u_{i,j}^{(k+1,l+1)}, (\omega_1)_{i+\frac{1}{2},j}^{(k+1,l+1)}, (\omega_2)_{i,j+\frac{1}{2}}^{(k+1,l+1)}, \dots\big) = 0, \\ N_2\big(\dots, u_{i,j}^{(k+1,l+1)}, (\omega_1)_{i+\frac{1}{2},j}^{(k+1,l+1)}, (\omega_2)_{i,j+\frac{1}{2}}^{(k+1,l+1)}, \dots\big) = 0, \\ N_3\big(\dots, u_{i,j}^{(k+1,l+1)}, (\omega_1)_{i+\frac{1}{2},j}^{(k+1,l+1)}, (\omega_2)_{i,j+\frac{1}{2}}^{(k+1,l+1)}, \dots\big) = 0, \end{cases} \tag{2.15}$$

we use a local fixed point method (after a local linearisation), yielding

$$\begin{pmatrix} S_{i,j} & 0 & 0 \\ \frac{\gamma}{h}(|\nabla u^{(k,l,p)}|_\beta)_{i+\frac{1}{2},j} & d_2 & \frac{\lambda}{h^2} \\ \frac{\gamma}{h}(|\nabla u^{(k,l,p)}|_\beta)_{i,j+\frac{1}{2}} & \frac{\lambda}{h^2} & d_3 \end{pmatrix} \begin{pmatrix} u_{i,j}^{(k,l,p+1)} \\ (\omega_1)_{i+\frac{1}{2},j}^{(k,l,p+1)} \\ (\omega_2)_{i,j+\frac{1}{2}}^{(k,l,p+1)} \end{pmatrix} = \begin{pmatrix} (g_1)_{i,j} \\ (g_2)_{i+\frac{1}{2},j} \\ (g_3)_{i,j+\frac{1}{2}} \end{pmatrix} \tag{2.16}$$

where $d_2 = \frac{2\lambda}{h^2} + \gamma(|\nabla u^{(k,l,p)}|_\beta^2)_{i+\frac{1}{2},j}$, $d_3 = \frac{2\lambda}{h^2} + \gamma(|\nabla u^{(k,l,p)}|_\beta^2)_{i,j+\frac{1}{2}}$,

$$\begin{cases} (g_1)_{i,j} = \left(f_1^{(k,l)}\right)_{i,j} + C^{(k,l,p)}_{i+\frac{1}{2},j} u^{(k)}_{i+1,j} + C^{(k,l,p)}_{i-\frac{1}{2},j} u^{(k+1)}_{i-1,j} + C^{(k,l,p)}_{i,j+\frac{1}{2}} u^{(k)}_{i,j+1} \\ \qquad + C^{(k,l,p)}_{i,j-\frac{1}{2}} u^{(k+1)}_{i,j-1}, \\ (g_2)_{i+\frac{1}{2},j} = \frac{\gamma}{h} \left|\nabla u^{(k,l,p)}\right|_{i+\frac{1}{2},j} u^{(k+1)}_{i+1,j} + \frac{\lambda}{h^2}\Big((\omega_1)^{(k+1)}_{i-\frac{1}{2},j} + (\omega_1)^{(k)}_{i+1+\frac{1}{2},j} \\ \qquad + (\omega_2)^{(k)}_{i+1,j+\frac{1}{2}} + (\omega_2)^{(k+1)}_{i,j-\frac{1}{2}} - (\omega_2)^{(k)}_{i+1,j-\frac{1}{2}}\Big), \\ (g_3)_{i,j+\frac{1}{2}} = \frac{\gamma}{h} \left|\nabla u^{(k,l,p)}\right|_{i,j+\frac{1}{2}} u^{(k)}_{i,j+1} + \frac{\lambda}{h^2}\Big((\omega_2)^{(k)}_{i,j-\frac{1}{2}} + (\omega_2)^{(k)}_{i,j+1+\frac{1}{2}} \\ \qquad + (\omega_1)^{(k)}_{i+\frac{1}{2},j+1} + (\omega_1)^{(k+1)}_{i-\frac{1}{2},j} - (\omega_1)^{(k+1)}_{i-\frac{1}{2},j+1}\Big), \end{cases} \tag{2.17}$$

and $S_{i,j} = 1 + C^{(k,l,p)}_{i+1/2,j} + C^{(k+1)}_{i-1/2,j} + C^{(k,l,p)}_{i,j+1/2} + C^{(k+1)}_{i,j-1/2}$. This Local Gauss-Seidel (LGS) method is described in Algorithm 2.

It is shown later that LGS and GGS are convergent on a single grid. We now consider accelerating the solution of (2.7a)–(2.7c) by using a nonlinear multigrid method with GGS and LGS as smoothers.

**Algorithm 2** $\mathbf{z}_h \leftarrow$ LGS $(u, \omega, \lambda, \gamma, \beta, \xi, \zeta$ and $\eta)$, where $u$ and $\omega$ are initial values, $\lambda$ and $\gamma$ are regularisation parameters, $\beta$ is the stabilizing parameter, $\xi$ is the number of outer iterations, $\zeta$ is the number of middle iterations (for operator splitting) and $\eta$ is the number of inner iterations

```
1:  Initialization: $u^0$, $\omega^0$, $\lambda$, $\gamma$, $\beta$, $\xi$, $\xi$ and $\eta$.
2:  For $k = 1$ to $\xi$
3:  For $i = 1, \dots m$, $j = 1, \dots n$
4:  For $l = 1$ to $\zeta$
5:      compute $(f_1^{(k,l)})_{i,j}$,
6:      For $p = 1$ to $\eta$
7:          compute all coefficients,
8:          compute $(u^{(k,l,p+1)}_{i,j}$, $(\omega_1)^{(k,l,p+1)}_{i+\frac{1}{2},j}$ and $(\omega_2)^{(k,l,p+1)}_{i,j+\frac{1}{2}}$ from (2.16).
9:      end
10:         $u^{(k,l+1)}_{i,j} = u^{(k,l,\eta)}_{i,j}$, $(\omega_1^{(k,l+1)})_{i+\frac{1}{2},j} = (\omega^{(k,l,\eta)})_{i+\frac{1}{2},j}$, $(\omega_2)^{(k,l+1)}_{i,j+\frac{1}{2}} = (\omega^{(k,l,\eta)})_{i,j+\frac{1}{2}}$
11: end
12: $u^{(k+1)}_{i,j} = u^{(k,\zeta)}_{i,j}$, $(\omega_1)^{(k+1)}_{i+\frac{1}{2},j} = (\omega_1)^{(k,\zeta)}_{i+\frac{1}{2},j}$, $(\omega_2)^{(k+1)}_{i,j+\frac{1}{2}} = (\omega_2)^{(k,\zeta)}_{i,j+\frac{1}{2}}$
13: end, end
14: end.
```

## 3 Solution by a nonlinear multigrid algorithm

The aim of this section is to develop a nonlinear multigrid (NMG) algorithm for the solution of (2.7a)–(2.7c). Multigrid (MG) methods [3, 5, 21, 22] have been applied to many fields of image processing such as restoration, segmentation and registration. These include high order models [4, 7, 13]. However for a nonlinear problem, convergence is not automatic and it is essential to construct a suitable smoother.

### 3.1 The nonlinear multigrid algorithm

A nonlinear multigrid (NMG) algorithm [3, 6, 21, 22] contains three ingredients: an iterative relaxation method as the smoother, a restriction operator to transfer solutions and residuals from a fine grid to a coarse grid, and an interpolation operator to interpolate the correction vectors back to a finer grid.

Here denote our coupled system of three nonlinear PDEs from (2.7a)–(2.7c) by $N^h(Z^h) = g^h$ on a fine grid $\Omega^h$ with size $h$, that is,

$$N_1(Z^h) = g_1^h, \qquad N_2(Z^h) = g_2^h, \qquad N_3(Z^h) = g_3^h, \tag{3.1}$$

where $Z^h = (u^h, \omega_1^h, \omega_2^h)$ and on the finest grid $g^h = (f_1^h, 0, 0)$. Assume $Z^h$ is an iteration obtained from applying a suitable smoother. The basic idea is to design an effective smoother (technique) which efficiently reduces the high frequency components of the fine grid errors. Then the coarse grid equation is $N^{2h}(Z^{2h}) = g^{2h}$. A two-grid full approximation algorithm is shown below in Algorithm 3 (repeated use of the idea leads to the full NMG). Here $R_h^{2h}$ and $I_{2h}^h$ [19] are to be discussed below. The relationship between the fine grid and the coarse grid is shown as in Fig. 2. There are three equations $u, \omega_1, \omega_2$ that are applied with $R_h^{2h}$ and $I_{2h}^h$ as transfer operations.

Here we consider GGS and LGS as smoothers. Transfer operations for $u$ on a standard coarsening of grids are straightforward [3, 4, 21] as shown below.

The restriction operator for a grid function $(u^h)_{i,j} = u^h(x_i, y_j)$ at a coarse grid point $(i, j) = (x_i, y_j) \in \Omega^{2h}$ is expressed as:

$$\begin{pmatrix} u^{2h} \\ \omega_1^{2h} \\ \omega_2^{2h} \end{pmatrix} = R_h^{2h} \begin{pmatrix} u^h \\ \omega_1^h \\ \omega_2^h \end{pmatrix},$$

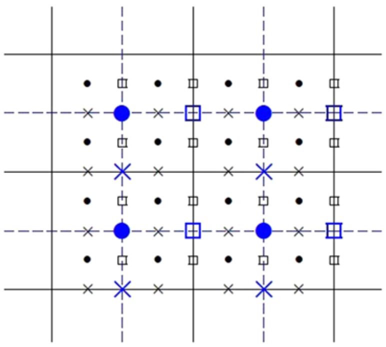


**Fig. 2** Relationship of the fine grid and coarser grid

**Algorithm 3** (FAS cycle). $\mathbf{Z}^h \leftarrow \mathrm{NMG}(\mathbf{Z}^h, g^h, \lambda, \gamma, \beta, \xi_0, \xi_1, \xi_2)$, where $Z^h$ and $g^h$ are initial values at the $h$th level, $\lambda$ and $\gamma$ are regularisation parameters, $\beta$ is the stabilizing parameter, $\xi_0$ is the number of steps of the coarsest grid, $\xi_1$ is the number of steps of presmoothing on the coarser grid and $\xi_2$ is the number of steps of the postsmoothing on the coarser grid

```
1: If $\Omega^h =$ is the coarsest grid, solve (2.10) accurately by using Algorithm 1 or
   Algorithm 2 do $\xi_0$ steps and return. Else continue the following step.
2: Presmoothing: Do $\xi_1$ steps of $Z^h \leftarrow$ LGS or GGS $(Z^h, g^h, \lambda, \gamma, \beta)$.
3: Restriction to the coarse grid, $Z^{2h} \leftarrow R_h^{2h} Z^h$.
4: Set the initial solution for the next level $\bar{Z}^{2h} \leftarrow Z^h$.
5: Compute the new right-hand side
   $g^{2h} \leftarrow R_h^{2h}(g^h - N_h(Z^h)) + N^{2h}(Z^{2h})$,
6: Implement $(Z^{2h}) \leftarrow \mathrm{NMG}(Z^{2h}, g^{2h}, \lambda, \gamma, \beta, \xi_0, \xi_1, \xi_2)$.
7: Add residual correction, $Z^h \leftarrow Z^h + I_{2h}^h(Z^{2h} - \bar{Z}^{2h})$.
8: Postsmoothing: do $\xi_2$ steps of $(Z^h) \leftarrow$ LGS or GGS $(Z^h, g^h, \lambda, \gamma, \beta)$.
```

where

$$\begin{cases} u_{i,j}^{2h} = \big[u_{2i-1,2j-1}^h + u_{2i-1,2j}^h + u_{2i,2j-1}^h + u_{2i,2j}^h\big]/4, \\ \big(\omega_1^{2h}\big)_{i,j} = \big[\big(\omega_1^h\big)_{2i-1,2j-1} + \big(\omega_1^h\big)_{2i-1,2j} + 2\big(\omega_1^h\big)_{2i,2j-1} \\ \qquad\qquad + 2\big(\omega_1^h\big)_{2i,2j} + \big(\omega_1^h\big)_{2i+1,2j} + \big(\omega_1^h\big)_{2i+1,2j-1}\big]/8, \\ \big(\omega_2^{2h}\big)_{i,j} = \big[\big(\omega_2^h\big)_{2i-1,2j-1} + \big(\omega_2^h\big)_{2i,2j-1} + 2\big(\omega_2^h\big)_{2i-1,2j} \\ \qquad\qquad + 2\big(\omega_2^h\big)_{2i,2j} + \big(\omega_2^h\big)_{2i,2j+1} + \big(\omega_2^h\big)_{2i-1,2j+1}\big]/8. \end{cases}$$

Similarly the interpolation operator for the staggered grid discretisation is expressed as

$$\begin{pmatrix} u^h \\ \omega_1^h \\ \omega_2^h \end{pmatrix} = I_{2h}^h \begin{pmatrix} u^{2h} \\ \omega_1^{2h} \\ \omega_2^{2h} \end{pmatrix},$$

with

$$\begin{cases} u_{2i,2j}^h = \big[9u_{i,j}^{2h} + 3\big(u_{i+1,j}^{2h} + u_{i,j+1}^{2h}\big) + u_{i+1,j+1}^{2h}\big]/16, \\ u_{2i+1,2j}^h = \big[9u_{i+1,j}^{2h} + 3\big(u_{i,j}^{2h} + u_{i+1,j+1}^{2h}\big) + u_{i,j+1}^{2h}\big]/16, \\ u_{2i,2j+1}^h = \big[9u_{i,j+1}^{2h} + 3\big(u_{i,j}^{2h} + u_{i+1,j+1}^{2h}\big) + u_{i+1,j}^{2h}\big]/16, \\ u_{2i+1,2j+1}^h = \big[9u_{i+1,j+1}^{2h} + 3\big(u_{i+1,j}^{2h} + u_{i,j+1}^{2h}\big) + u_{i,j}^{2h}\big]/16; \end{cases}$$

$$\begin{cases} (\omega_1^h)_{2i,2j} = [3(\omega_1^h)_{i,j} + (\omega_1^h)_{i,j+1}]/4, \\ (\omega_1^h)_{2i,2j+1} = [3(\omega_1^h)_{i,j+1} + (\omega_1^h)_{i,j}]/4, \\ (\omega_1^h)_{2i+1,2j} = [3(\omega_1^h)_{i,j} + 3(\omega_1^h)_{i+1,j} + (\omega_1^h)_{i,j+1} + (\omega_1^h)_{i+1,j+1}]/8, \\ (\omega_1^h)_{2i+1,2j+1} = [(\omega_1^h)_{i,j} + (\omega_1^h)_{i+1,j} + 3(\omega_1^h)_{i,j+1} + 3(\omega_1^h)_{i+1,j+1}]/8; \end{cases}$$

and

$$\begin{cases} (\omega_2^h)_{2i,2j} = [3(\omega_2^h)_{i,j} + (\omega_2^h)_{i+1,j}]/4, \\ (\omega_2^h)_{2i+1,2j} = [(\omega_2^h)_{i,j} + 3(\omega_2^h)_{i+1,j}]/4, \\ (\omega_2^h)_{2i,2j+1} = [3(\omega_2^h)_{i,j} + 3(\omega_2^h)_{i,j+1} + (\omega_2^h)_{i+1,j} + (\omega_2^h)_{i+1,j+1}]/8, \\ (\omega_2^h)_{2i,2j+1} = [(\omega_2^h)_{i,j} + (\omega_2^h)_{i,j+1} + 3(\omega_2^h)_{i+1,j} + 3(\omega_2^h)_{i+1,j+1}]/8. \end{cases}$$

## 3.2 Local Fourier analysis

Although we have decided to use GGS and LGS as smoothers, their smoothing properties which crucially determine the convergence of the underlying MG algorithm are as yet unknown. The most useful tool for analyzing the smoothing property of a smoother is the Local Fourier Analysis (LFA). As remarked in [21, 23, 24], one needs to freeze the nonlinear coefficients. Define the error functions $e_1^k = \bar{u} - u^k, e_2^k = \bar{\omega}_1 - \omega_1^k$, and $e_3^k = \bar{\omega}_2 - \omega_2^k$ where $(\bar{u}, \bar{\omega}_1, \bar{\omega}_2)$ are the true solutions of (2.10), and $(u^k, \omega_1^k, \omega_2^k)$ are the results at the $k$th step of the GS iteration. Then the local error functions can be expanded in a Fourier representation as follows:

$$\begin{aligned} [e_1^k]_{i,j} &= \sum_{\phi_1,\phi_2=-\frac{n}{2}}^{\frac{n}{2}} [\varphi_1^k]_{\phi_1,\phi_2} e^{\mathbf{i}\frac{\theta_1 x_i + \theta_2 y_j}{h}}, \\ [e_2^k]_{i+1/2,j} &= \sum_{\phi_1,\phi_2=-\frac{n}{2}}^{\frac{n}{2}} [\varphi_2^k]_{\phi_1,\phi_2} e^{\mathbf{i}\frac{\theta_1 x_{i+1/2} + \theta_2 y_j}{h}}, \\ [e_3^k]_{i,j+1/2} &= \sum_{\phi_1,\phi_2=-\frac{n}{2}}^{\frac{n}{2}} [\varphi_3^k]_{\phi_1,\phi_2} e^{\mathbf{i}\frac{\theta_1 x_i + \theta_2 y_{j+1/2}}{h}}, \end{aligned} \tag{3.2}$$

where $\theta = (\theta_1, \theta_2) \in \Theta = [-\pi, \pi)^2$, $\theta_1 = 2\pi\phi_1/n$, $\theta_2 = 2\pi\phi_2/n$, and $\mathbf{i} = \sqrt{-1}$. Then the local amplification matrix $\mathbf{M}(\phi_1, \phi_2)$ is given by:

$$\begin{pmatrix} [\varphi_1^{(k,p+1)}]_{\phi_1,\phi_2} \\ [\varphi_2^{(k,p+1)}]_{\phi_1,\phi_2} \\ [\varphi_3^{(k,p+1)}]_{\phi_1,\phi_2} \end{pmatrix} = \mathbf{M}(\phi_1, \phi_2) \begin{pmatrix} [\varphi_1^{(k,p)}]_{\phi_1,\phi_2} \\ [\varphi_2^{(k,p)}]_{\phi_1,\phi_2} \\ [\varphi_3^{(k,p)}]_{\phi_1,\phi_2} \end{pmatrix}, \tag{3.3}$$

with the smoothing rate for this nonlinear framework defined as the spectral radius at $(i,j)$, $W_{i,j} = \max_{\phi_1,\phi_2} \rho(\mathbf{M}(\phi_1,\phi_2))$ in the high frequency range $(\phi_1,\phi_2) \in [-n/2, n/2]\backslash[-n/4, n/4]$. The overall rate for one iteration is

$$\rho = \max_{i,j} W_{i,j}. \tag{3.4}$$

*LFA for the GGS method* It is useful to rewrite the GGS iteration with respect to a difference stencil representation. Specifically,

$$\begin{bmatrix} 0 & -C^{(k)}_{i-\frac{1}{2},j} & 0 \\ -C^{(k)}_{i,j-\frac{1}{2}} & S^{(k)}_{i,j} & 0 \\ 0 & 0 & 0 \end{bmatrix} u^{(k,p+1)} = \left(f_1^k\right)_{i,j} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & C^{(k)}_{i,j+\frac{1}{2}} \\ 0 & C^{(k)}_{i+\frac{1}{2},j} & 0 \end{bmatrix} u^{(k,p)}$$

for the first equation of (2.11);

$$\begin{aligned} &\begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{\gamma}{h}|\nabla u^{(k)}|_{i+\frac{1}{2},j} & 0 \\ 0 & 0 & 0 \end{bmatrix} u^{(k,p+1)} + \begin{bmatrix} 0 & -\frac{2\lambda}{h^2} & 0 \\ 0 & \frac{2\lambda}{h^2} + \gamma|\nabla u^{(k)}|^2_{i+\frac{1}{2},j} & 0 \\ 0 & 0 & 0 \end{bmatrix} (\omega_1)^{(k,p+1)} \\ &\quad + \begin{bmatrix} 0 & 0 & 0 \\ -\frac{2\lambda}{h^2} & \frac{2\lambda}{h^2} & 0 \\ \frac{2\lambda}{h^2} & 0 & 0 \end{bmatrix} (\omega_2)^{(k,p+1)} \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{\gamma}{h}|\nabla u^{(k)}|_{i+\frac{1}{2},j} & 0 \end{bmatrix} u^{(k,p)} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{2\lambda}{h^2} & 0 \end{bmatrix} (\omega_1)^{(k,p)} \\ &\quad + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{2\lambda}{h^2} & 0 \end{bmatrix} (\omega_2)^{(k,p)} \end{aligned}$$

for the second equation of (2.11); and

$$\begin{aligned} &\begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{\gamma}{h}|\nabla u^{(k)}|_{i,j+\frac{1}{2}} & 0 \\ 0 & 0 & 0 \end{bmatrix} u^{(k,p+1)} + \begin{bmatrix} 0 & -\frac{2\lambda}{h^2} & 0 \\ 0 & \frac{2\lambda}{h^2} & 0 \\ 0 & 0 & 0 \end{bmatrix} (\omega_1)^{(k,p+1)} \\ &\quad + \begin{bmatrix} 0 & 0 & 0 \\ -\frac{2\lambda}{h^2} & \frac{2\lambda}{h^2} + \gamma|\nabla u^{(k)}|^2_{i,j+\frac{1}{2}} & 0 \\ 0 & 0 & 0 \end{bmatrix} (\omega_2)^{(k,p+1)} \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{\gamma}{h}|\nabla u^{(k)}|_{i,j+\frac{1}{2}} \\ 0 & 0 & 0 \end{bmatrix} u^{(k,p)} + \begin{bmatrix} 0 & 0 & -\frac{2\lambda}{h^2} \\ 0 & 0 & \frac{2\lambda}{h^2} \\ 0 & 0 & 0 \end{bmatrix} (\omega_1)^{(k,p)} \end{aligned}$$

$$+\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & \frac{2\lambda}{h^2}\\ 0 & 0 & 0\end{bmatrix}(\omega_2)^{(k,p)}$$

for the third equation of (2.12). To find $\mathbf{M}_{i,j}(\phi)$, we substitute these relations into the local error functions (3.2), yielding

$$\mathbf{M}_{i,j}(\phi_1,\phi_2)$$

$$= \begin{pmatrix} S^{(k)}_{i,j} - C^{(k)}_{i-\frac12,j}e^{-\mathbf{i}\theta_1} - C^{(k)}_{i,j-\frac12}e^{-\mathbf{i}\theta_2} & 0 & 0\\ \frac{\gamma}{h}|\nabla u^{(k)}|_{i+\frac12,j} & s_2 & \frac{\lambda}{h^2}(1-e^{-\mathbf{i}\theta_2}+e^{\mathbf{i}(\theta_1-\theta_2)})\\ \frac{\gamma}{h}|\nabla u^{(k)}|_{i,j+\frac12} & \frac{\lambda}{h^2}(1-e^{-\mathbf{i}\theta_1}) & s_3 \end{pmatrix}^{-1}$$

$$\cdot\begin{pmatrix} C^{(k)}_{i+\frac12,j}e^{\mathbf{i}\theta_1} + C^{(k)}_{i,j+\frac12}e^{\mathbf{i}\theta_2} & 0 & 0\\ \frac{\gamma}{h}|\nabla u^{(k)}|_{i+\frac12,j}e^{\mathbf{i}\theta_1} & \frac{\lambda}{h^2}e^{\mathbf{i}\theta_1} & \frac{\lambda}{h^2}e^{\mathbf{i}\theta_1}\\ \frac{\gamma}{h}|\nabla u^{(k)}|_{i,j+\frac12}e^{\mathbf{i}\theta_2} & \frac{\lambda}{h^2}(e^{\mathbf{i}\theta_2}-e^{\mathbf{i}(\theta_2-\theta_1)}) & \frac{\lambda}{h^2}e^{\mathbf{i}\theta_2} \end{pmatrix},$$

where $s_2 = \frac{2\lambda}{h^2} + \gamma|\nabla u^{(k)}|^2_{i+\frac12,j} - \frac{\lambda}{h^2}e^{-\mathbf{i}\theta_1}$, $s_3 = \frac{2\lambda}{h^2} + \gamma|\nabla u^{(k)}|^2_{i,j+\frac12} - \frac{\lambda}{h^2}e^{-\mathbf{i}\theta_2}$.

*LFA for the LGS method* From (2.14)–(2.17), we know that the difference between LGS and GGS is the updating of the coefficients and the right hand side terms. Thus $\mathbf{M}_{i,j}(\phi)$ for LGS is given by

$$\mathbf{M}_{i,j}(\phi_1,\phi_2)$$

$$= \begin{pmatrix} s_1 & 0 & 0\\ \frac{\gamma}{h}|\nabla u^{(p)}|_{i+\frac12,j} & \frac{2\lambda}{h^2}+\gamma|\nabla u^{(p)}|^2_{i+\frac12,j} - \frac{\lambda}{h^2}e^{-\mathbf{i}\theta_1} & \frac{\lambda}{h^2}(1-e^{-\mathbf{i}\theta_2}+e^{\mathbf{i}(\theta_1-\theta_2)})\\ \frac{\gamma}{h}|\nabla u^{(p)}|_{i,j+\frac12} & \frac{\lambda}{h^2}(1-e^{-\mathbf{i}\theta_1}) & \frac{2\lambda}{h^2}+\gamma|\nabla u^{(p)}|^2_{i,j+\frac12} - \frac{\lambda}{h^2}e^{-\mathbf{i}\theta_2} \end{pmatrix}^{-1}$$

$$\cdot\begin{pmatrix} C^{(p)}_{i+\frac12,j}e^{\mathbf{i}\theta_1} + C^{(k)}_{i,j+\frac12}e^{\mathbf{i}\theta_2} & 0 & 0\\ \frac{\gamma}{h}|\nabla u^{(p)}|_{i+\frac12,j}e^{\mathbf{i}\theta_1} & \frac{\lambda}{h^2}e^{\mathbf{i}\theta_1} & \frac{\lambda}{h^2}e^{\mathbf{i}\theta_1}\\ \frac{\gamma}{h}|\nabla u^{(p)}|_{i,j+\frac12}e^{\mathbf{i}\theta_2} & \frac{\lambda}{h^2}(e^{\mathbf{i}\theta_2}-e^{\mathbf{i}(\theta_2-\theta_1)}) & \frac{\lambda}{h^2}e^{\mathbf{i}\theta_2} \end{pmatrix},$$

where $s_1 = S^{(p)}_{i,j} - C^{(p)}_{i-1/2,j}e^{-\mathbf{i}\theta_1} - C^{(p)}_{i,j-1/2}e^{-\mathbf{i}\theta_2}$.

To give an indication of the effectiveness of the above GGS and LGS smoothers, we take the left image in Fig. 3 as an example and compute the smoothing rate $\rho$ for $\beta = 10^{-4}$, where scattered points denote $W_{i,j} > 0.95$. From Fig. 3, we note that there are fewer scattered points for LGS than for GGS, implying that a LGS smoother is better than the GGS smoother. We also note that all dots appear in plain areas without edges. In Table 1, we compute rates with different values of $\beta$ and observe that for $\beta \geq 10^{-3}$ GGS and LGS are reasonably effective smoothers. However for $\beta \leq 10^{-4}$,

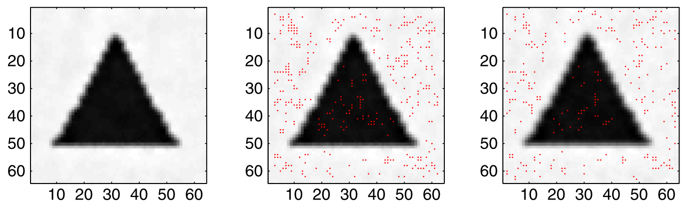


**Fig. 3** Example of the behavior of the smoothing rate $\rho$ when $\beta = 1e-4$. *On the left* the denoised image after one MG step, in the middle the smoothing rate using GGS and *on the right* the smoothing rate using LGS. The *red points* indicate the points where $W_{i,j} > 0.96$ (Color figure online)

**Table 1** The value of $\rho$ from (3.4) for different $\beta$ and an image of size $64 \times 64$

| noise level | value of $\beta$ | $\rho$ for GGS | $\rho$ for LGS |
|---|---|---|---|
| 5 % noise | $\beta = 1e-2$ | 0.4761 | 0.4772 |
| | $\beta = 1e-3$ | 0.7164 | 0.7163 |
| | $\beta = 1e-4$ | 0.9629 | 0.9628 |

they become less effective, e.g. when $\beta = 10^{-4}$ with noise level of 5 %, $\rho = 0.9629$ which implies that one needs 20 iterations to reduce a high frequency error by 0.4695 since $0.9629^{20} \approx 0.4695$. Although the evidence from the LFA suggests that LGS is better than GGS, one may see that the smoothing rate indicated by $\rho$ does not completely describe the overall efficiency of the smoother in this context.

To provide further evidence, we define an accumulated smoothing rate that takes care of the nonlinearity of the underlying equations. Let the usual rate matrix at iteration $\ell$ be $W^{(\ell)}$ and the usual smoothing rate at iteration $\ell$ be $\rho_\ell = \max_{i,j} W^{(\ell)}_{i,j}$. Then the accumulated smoothing rate after $k$ iterations will be

$$\overline{\rho}^{[k]} = \max_{i,j} \left(M^{(k)}_{accu}\right)_{i,j} \tag{3.5}$$

where $M^{(k)}_{accu} = W^{(1)}_{i,j} W^{(2)}_{i,j} \ldots W^{(k)}_{i,j}$. Here our assumption is that the location of the maximum of each $\rho_\ell$ problem is not necessarily fixed at the same place. Then $\overline{\rho}^{[k]} < \rho_1 \rho_2 \ldots \rho_k$ serves as a better indicator of smoothing efficiency for a nonlinear problem with coefficients depending on $(i, j)$.

From Tables 2, 3, one observes that the accumulated rate $\overline{\rho}$ by (3.5) is relatively smaller for all $\beta$ than the traditional rate $\rho$ by (3.4). This suggests that our new smoothers will be efficient for solving our model by a NMG.

## 4 Numerical results

In this section, the primary aim is to illustrate the effectiveness of our new Algorithm 2 and show that it is the best among existing implementations for the mean curvature based denoising model (1.4). A secondary aim is to compare with other models that were also proposed to reduce the staircasing effect of the total variation model [18] and thus to assess the competitiveness of Algorithm 2 in a wider context.

**Table 2** Comparison of $\rho, \overline{\rho}$ for GGS with different $\beta$ for an image of 64 × 64 and noise level of 5 %

| $\beta$ | $\rho_1$ | $\overline{\rho}^{[1]}$ | $\rho_2$ | $\overline{\rho}^{[2]}$ | $\rho_3$ | $\overline{\rho}^{[3]}$ | $\rho_4$ | $\overline{\rho}^{[4]}$ | $\rho_5$ | $\overline{\rho}^{[5]}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $10^{-2}$ | 0.4683 | 0.4683 | 0.5287 | 0.2476 | 0.5226 | 0.1279 | 0.5210 | 0.0665 | 0.5218 | 0.0347 |
| $10^{-3}$ | 0.6883 | 0.6883 | 0.6883 | 0.4736 | 0.6883 | 0.3260 | 0.6883 | 0.2244 | 0.6883 | 0.1544 |
| $10^{-4}$ | 0.9578 | 0.9578 | 0.9580 | 0.9173 | 0.9580 | 0.8787 | 0.9580 | 0.8418 | 0.9580 | 0.8065 |

**Table 3** Comparison of $\rho, \overline{\rho}$ for LGS with different $\beta$ for an image of 64 × 64 and noise level of 5 %

| $\beta$ | $\rho_1$ | $\overline{\rho}^{[1]}$ | $\rho_2$ | $\overline{\rho}^{[2]}$ | $\rho_3$ | $\overline{\rho}^{[3]}$ | $\rho_4$ | $\overline{\rho}^{[4]}$ | $\rho_5$ | $\overline{\rho}^{[5]}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $10^{-2}$ | 0.4920 | 0.4920 | 0.5315 | 0.2564 | 0.5492 | 0.1395 | 0.5604 | 0.0782 | 0.5641 | 0.0441 |
| $10^{-3}$ | 0.6921 | 0.6921 | 0.6922 | 0.4791 | 0.6922 | 0.3316 | 0.6922 | 0.2295 | 0.6922 | 0.1588 |
| $10^{-4}$ | 0.9586 | 0.9586 | 0.9586 | 0.9189 | 0.9586 | 0.8808 | 0.9586 | 0.8443 | 0.9586 | 0.8093 |

However it must be said that comparing different models with varying parameters cannot be done without using some heuristics for choosing feasible and non-optimal parameters; such conclusions are for reference only and taken with caution.

To measure the quality of the restored images, the relative residual, Signal-to-Noise Ratio (SNR) [18] and the peak Signal-to-Noise Ratio (PSNR) [5] are used. They are respectively defined by:

$$\text{ReRes} = \frac{\|\mathbf{N}\mathbf{z}^{(k)} - \mathbf{g}\|_2}{\|\mathbf{N}\mathbf{z}^0 - \mathbf{g}\|_2},$$

$$\text{SNR} = 10\log_{10}\left(\frac{\|u - u_{mean}\|_2^2}{\|u - \tilde{u}\|_2^2}\right) \quad \text{and} \quad \text{PSNR} = 10\log_{10}\left(\frac{255^2}{\frac{1}{mn}\|\tilde{u} - u\|_2^2}\right),$$

where $u$ and $\tilde{u}$ are the true image and the restored image respectively, and $u_{mean}$ is the mean value of the original image. Higher SNR/PSNR values indicate a better restoration. Of course, in a real life situation, the latter measures are not possible when $u$ is not known. The initial values are set to be the observed image, that is $u^0 = f, \omega = \nabla u^0/|\nabla u^0|_\beta$. The methods to be tested and compared are listed in Table 4. The AL method introduced in [20, 25, 29] can be adopted here to solve (1.4)

$$\begin{aligned}\min \mathscr{L}(u, \mathbf{m}, \mathbf{n}, \mathbf{p}; \lambda_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3) = {} & \frac{1}{2}\int_\Omega (u - f)^2 + \lambda \int_\Omega \Phi(\nabla \cdot \mathbf{n}) \\ & + \gamma_1 \int_\Omega \big(|\mathbf{p}| - \mathbf{m}\cdot\mathbf{p}\big) + \int_\Omega \lambda_1\big(|\mathbf{p}| - \mathbf{m}\cdot\mathbf{p}\big) \\ & + \frac{\gamma_2}{2}\int_\Omega |\mathbf{p} - \nabla u|^2 + \int_\Omega \boldsymbol{\lambda}_2 \cdot (\mathbf{p} - \nabla u) \\ & + \frac{\gamma_3}{2}\int_\Omega |\mathbf{m} - \mathbf{n}|^2 + \int_\Omega \boldsymbol{\lambda}_3 \cdot (\mathbf{m} - \mathbf{n}) + \delta_{\mathscr{R}}(\mathbf{m}),\end{aligned} \tag{4.1}$$

**Table 4** Test methods for numerical experiments

| Method | full name | parameters |
|---|---|---|
| AL | The AL Method [20] | $\lambda, \gamma_1, \gamma_2, \gamma_3$ |
| TV | Rudin-Osher-Fatemi [18] | $\lambda, \beta$ |
| TGV | Total Generalized Variation [2] | $\alpha_1, \alpha_0$ |
| SB | Split Bregman [12] | $\mu$ |
| MG | Stabilized smoother based MG method of Brito-Chen [4] | $\lambda,\gamma, \beta$ |
| NMG1 | NMG of Sect. 4 with GGS smoother | $\lambda, \gamma, \beta$ |
| NMG2 | NMG of Sect. 4 with LGS smoother | $\lambda, \gamma, \beta$ |

where $\delta_{\mathscr{R}}(\mathbf{m})$ denotes a characteristic function

$$\delta_{\mathscr{R}}(\mathbf{m}) = \begin{cases} 0 & \mathbf{m} \in \mathscr{R}, \\ +\infty & \text{otherwise}, \end{cases}$$

for other details; see [14].

*Parameter settings* Table 4 has indicated the parameters involved in each model. Although it is a challenging or impossible task to obtain the optimal choices (except for the TV case [27]), numerical tests can be used to establish the operational ranges for a practical but non-optimal choice. Below are the specific ranges and choices for our tests (except when stated otherwise):

1. AL: $\lambda_{max} = 1$, $\lambda_{min} = 0.001$, $(\gamma_1)_{max} = 100$, $(\gamma_1)_{min} = 0.1$, $\gamma_2 = 0.05$, $\gamma_3 = 0.05$.
2. TV: $\lambda_{max} = 1$, $\lambda_{min} = 0.001$.
3. SB: $\mu_{max} = 1$, $\mu_{min} = 0.001$.
4. MG: $\lambda_{max} = 1$, $\lambda_{min} = 0.001$, $\gamma_{max} = 1000$, $\gamma_{max} = 100$; $\xi_0 = 300$ and $\xi_1 = \xi_2 = 10$.
5. TGV: $(\alpha_0)_{max} = 1$, $(\alpha_0)_{min} = 0.001$, $(\alpha_1) = 1$, $\lambda_{max} = 1$, $\lambda_{min} = 0.001$.
6. NMG1 and NMG2: $\lambda_{max} = 1$, $\lambda_{min} = 0.001$, $\gamma_{max} = 100$, $\gamma_{max} = 0.1$; $\xi_0 = 300$ and $\xi_1 = \xi_2 = 10$.

These heuristically obtained choices give relatively good results for each model. For other examples or improvement, fine tuning might be needed. However, since the primary purpose of this paper is to present a new method for solving model (1.4) for a small $\beta$, further fine tuning of other parameters is a secondary concern. All the tests are stopped when ReRes $< 10^{-3}$.

## 4.1 Comparisons with two previous methods for model (1.4)

No fast solvers existed for (1.4) before the work of [4] i.e. MG. This MG and the method AL are the two viable methods to be compared with our new algorithms NMG1 and NGM2. Two specific comparisons are for sensitivity of $\lambda$ and $\beta$. Other parameters (shown below) are taken from their feasible ranges as given above. As the same model is solved, it is natural that we use the regularisation parameter $\lambda$ in all tests.

*$\lambda$- dependence test* We first compare the sensitivity of all four methods with respect to varying the regularisation parameter $\lambda$. The test image used is the "triangle" image of size $256 \times 256$. In Table 5, we compare the restoration quality (via SNR and PSNR) and efficiency (va 'Iteration steps' and 'CPU') of all methods. There, $\gamma$ is taken as 100 and $\beta$ is taken as $10^{-2}$ in all cases. For the AL algorithm, $\gamma_1 = 100$, $\gamma_2 = \gamma_3 = 0.005$ are taken. Usually, the regularisation parameter $\lambda$ needs to be increased as the level of noise gets higher. From this table, one can see that, at the same level of noise, both our new algorithms NMG1/2 and MG perform similarly, and AL is the worst in $\lambda$ sensitivity. Evidently NMG1 performed better than NMG2.

**Table 5** $\lambda$-sensitivity comparison using the "triangle" image of size $256 \times 256$ with varying $\lambda$ and other fixed parameters, the standard deviation of noise $\sigma = 10$

| parameter $\lambda$ | method | SNR | PSNR | iteration | CPU time |
|---|---|---|---|---|---|
| $\lambda = 0.1$ | AL | 30.31 | 38.05 | 133 | 4 |
| | MG | 30.39 | 38.11 | 2 | 51 |
| | NMG1 | 32.03 | 39.78 | 2 | 16 |
| | NMG2 | 31.68 | 39.42 | 2 | 49 |
| $\lambda = 0.01$ | AL | 17.48 | 28.22 | 236 | 25 |
| | MG | 32.68 | 40.43 | 2 | 51 |
| | NMG1 | 32.11 | 39.85 | 2 | 16 |
| | NMG2 | 31.98 | 39.72 | 2 | 49 |
| $\lambda = 0.001$ | AL | 12.47 | 20.28 | 130 | 16 |
| | MG | 27.16 | 34.90 | 2 | 51 |
| | NMG1 | 29.16 | 36.90 | 3 | 32 |
| | NMG2 | 30.29 | 38.03 | 2 | 49 |

**Table 6** $\beta$-sensitivity comparison using the "peppers" test image with size $256 \times 256$, varying $\beta$, other parameters fixed, and the standard deviation $\sigma = 10$

| parameter $\beta$ | method | SNR | PSNR | iteration | CPU time |
|---|---|---|---|---|---|
| $\beta = 1$ | MG | 22.05 | 30.32 | 2 | 49 |
| | NMG1 | 21.17 | 29.44 | 2 | 16 |
| | NMG2 | 21.16 | 29.43 | 1 | 28 |
| $\beta = 10^{-2}$ | MG | 26.76 | 35.03 | 2 | 49 |
| | NMG1 | 27.24 | 35.51 | 2 | 16 |
| | NMG2 | 27.23 | 35.50 | 1 | 28 |
| $\beta = 10^{-4}$ | MG | $*$ | $*$ | $*$ | $*$ |
| | NMG1 | 27.55 | 35.81 | 3 | 23 |
| | NMG2 | 27.54 | 35.81 | 4 | 137 |

**Table 7** Performance of NMG1/2 for a "triangle" image, $\lambda = 0.01$ and $\gamma = 10$. Here we list results of SNR and PSNR with different $\beta$, the standard deviation $\sigma = 10$

| method | $\beta$ | $n \times n$ | SNR | PSNR | MG cycles | CPU |
|---|---|---|---|---|---|---|
| NMG1 | $10^{-2}$ | $128 \times 128$ | 26.14 | 37.68 | 3 | 8 |
| | | $256 \times 256$ | 30.97 | 38.72 | 3 | 23 |
| | | $512 \times 512$ | 31.52 | 39.27 | 1 | 33 |
| | | $1024 \times 1024$ | 31.62 | 39.36 | 1 | 131 |
| NMG1 | $10^{-4}$ | $128 \times 128$ | 27.85 | 38.23 | 6 | 17 |
| | | $256 \times 256$ | 33.75 | 41.50 | 5 | 38 |
| | | $512 \times 512$ | 36.62 | 44.36 | 5 | 138 |
| | | $1024 \times 1024$ | 38.03 | 45.78 | 3 | 355 |
| NMG2 | $10^{-2}$ | $128 \times 128$ | 25.21 | 32.95 | 2 | 13 |
| | | $256 \times 256$ | 31.23 | 38.98 | 1 | 27 |
| | | $512 \times 512$ | 31.74 | 39.48 | 1 | 112 |
| | | $1024 \times 1024$ | 31.84 | 39.59 | 1 | 446 |
| NMG2 | $10^{-4}$ | $128 \times 128$ | 25.33 | 33.07 | 3 | 18 |
| | | $256 \times 256$ | 33.65 | 41.46 | 2 | 49 |
| | | $512 \times 512$ | 35.55 | 43.29 | 2 | 199 |
| | | $1024 \times 1024$ | 38.01 | 45.73 | 1 | 446 |

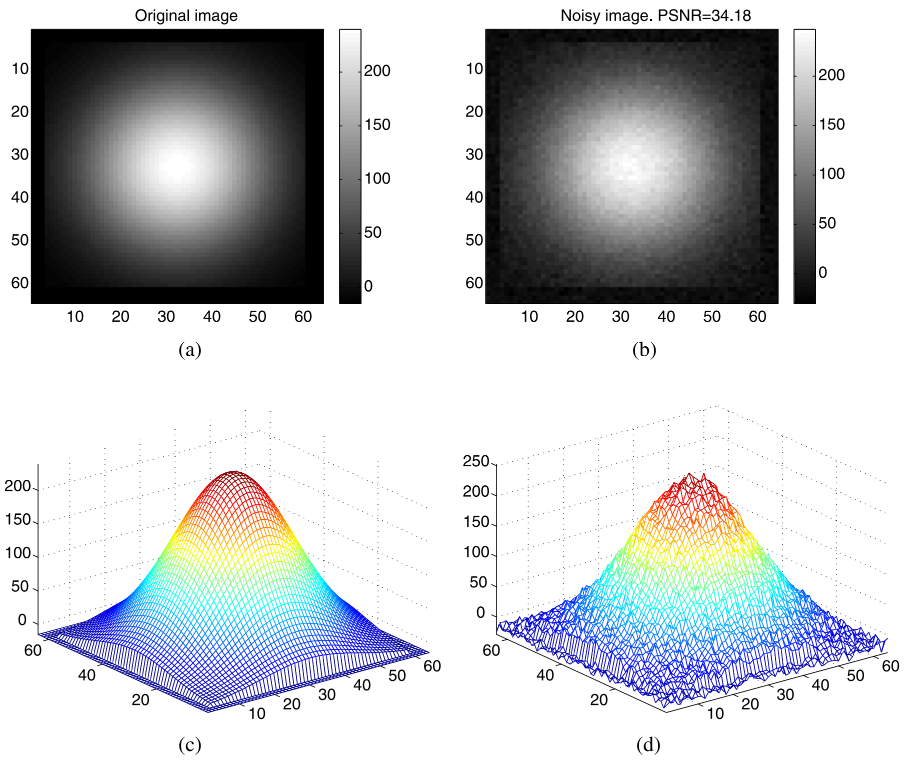


**Fig. 4** The "hemi-sphere" test image: (**a**) the original image; (**b**) the noisy image with the standard deviation $\sigma = 5$; (**c**) surface plot of (**a**); (**d**) surface plot of (**b**)

*$\beta$-dependence test* We next compare the sensitivity of three methods with respect to varying the stabilizing parameter $\beta$. We do not include the AL algorithm since it does not involve the parameter $\beta$ and it is not as good as MG as known from Table 5. As mentioned in the introduction, the method MG does not work for $\beta < 10^{-2}$. Hence this set of tests has 2 purposes: (i) comparison of performances of NMG1/2 against MG for $\beta \geq 10^{-2}$; (ii) comparison of convergence of NMG1/2 for $\beta < 10^{-2}$. Therefore, this second test is our top priority and the first $\beta$-sensitivity test is a lesser concern as the main competitor MG dos not even work for $\beta < 10^{-2}$.

The test image used is the "peppers" image of size $256 \times 256$ (as in [4]). The stabilizing parameter $\beta$ varies from 1 to $10^{-4}$. Here $\lambda = 0.01$ is fixed in all algorithms, $\gamma = 100$ in the MG and $\gamma = 2$ in the NMG1 and NMG2. The numerical results are shown in Table 6, where '*' means no convergence. From Table 6, evidently, our new algorithms NMG1/2 perform well for $\beta = 10^{-4}$ (and also for even smaller $\beta$). From this table, when the other parameters are fixed, we gets more better quality of recovery as $\beta$ gets small, but the cost increases due to increased nonlinearity. Thus we can conclude that our new algorithms still converge for small $\beta$.

Table 7 shows some test results with varying the sizes of the "triangle" image, in order to further compare the efficiency of NMG1 and NMG2 where $\lambda = 0.01$ and $\gamma = 10$ are fixed for all tests. From the Table 7, one can see although NMG2

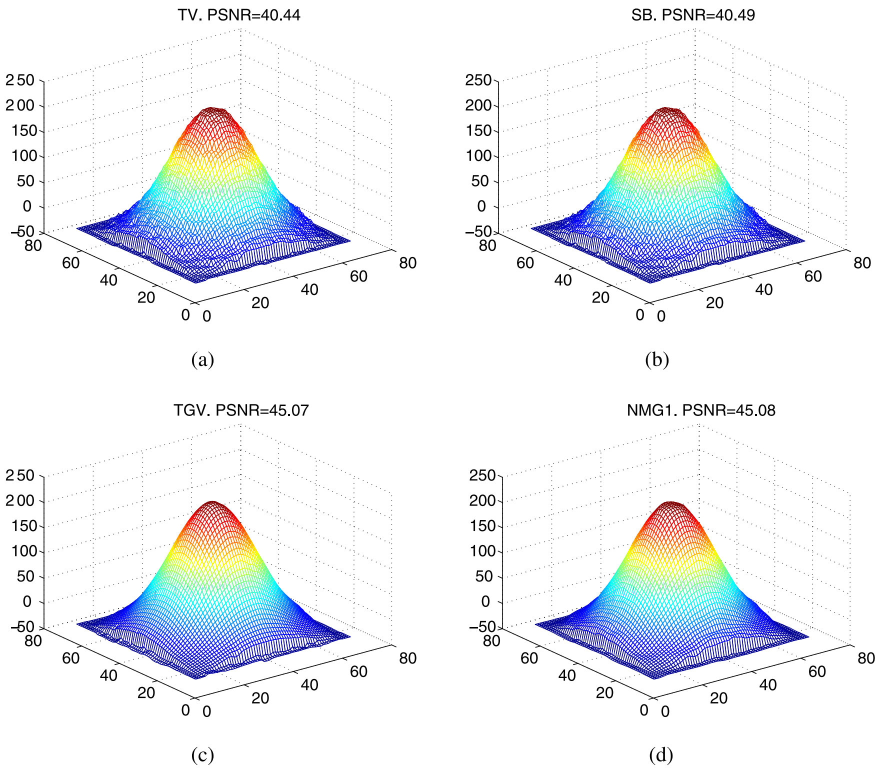


**Fig. 5** Mesh plots of magnified regions. (**a**) restoration by TV; (**b**) by SB; (**c**) by TGV; (**d**) by NMG1

**Table 8** Comparison of performance of staircasing reduction methods for three test images

| image | noise | method | PSNR | parameters |
|---|---|---|---|---|
| "hemi-sphere" | $\sigma = 5$ | TV | 40.44 | $\lambda = 0.13$, $\beta = 10^{-5}$ |
| | | SB | 40.49 | $\mu = 0.13$ |
| | | TGV | 45.07 | $\lambda = 0.1$, $\alpha_0 = 3.5$, $\alpha_1 = 1$ |
| | | NMG1 | 45.08 | $\lambda = 0.008$, $\gamma = 19$, $\beta = 10^{-5}$ |
| "triangle" | $\sigma = 10$ | TV | 37.37 | $\lambda = 0.1$, $\beta = 10^{-5}$ |
| | | SB | 37.40 | $\mu = 0.1$ |
| | | TGV | 38.74 | $\lambda = 0.06$, $\alpha_0 = 0.8$, $\alpha_1 = 1$ |
| | | NMG1 | 38.37 | $\lambda = 0.002$, $\gamma = 5$, $\beta = 10^{-4}$ |
| "pepper" | $\sigma = 15$ | TV | 34.99 | $\lambda = 0.06$, $\beta = 10^{-5}$ |
| | | SB | 35.08 | $\mu = 0.07$ |
| | | TGV | 35.32 | $\lambda = 0.04$, $\alpha_0 = 0.8$, $\alpha_1 = 1$ |
| | | NMG1 | 35.55 | $\lambda = 0.007$, $\gamma = 8$, $\beta = 10^{-4}$ |

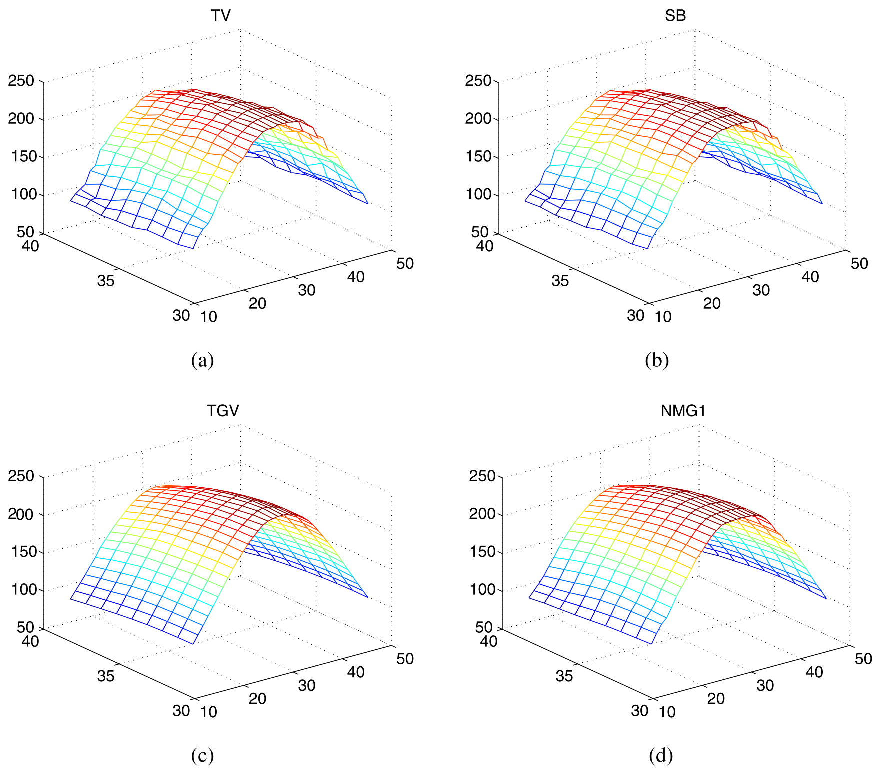
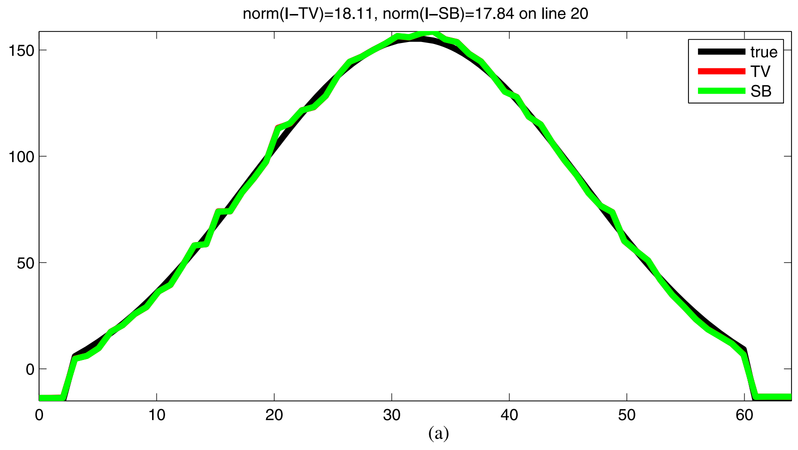


**Fig. 6** Mesh plots of magnified regions. (**a**) restoration by TV; (**b**) by SB; (**c**) by TGV; (**d**) by NMG1



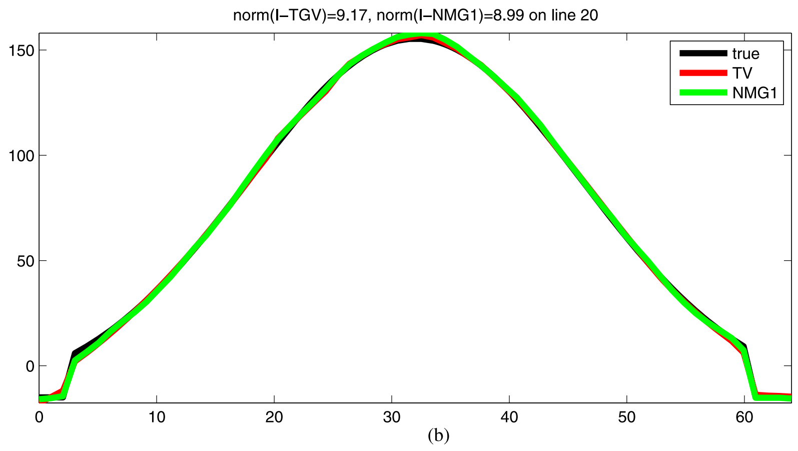


**Fig. 7** Comparison results plotted along a line: (**a**) the TV model; (**b**) higher order models

sometimes achieves better quality than NMG1, the convergence speed is slower as the size of image gets larger. So here NMG1 is our recommended solver for the mean curvature- based model (1.4).

## 4.2 Comparisons of model (1.4) with three other models

We finally compare our recommended NMG1 algorithm with the TV, the SB and the TGV; the latter two have staircasing reduction capability and they are known to perform better than the TV. The test images taken are a synthetic semi-hemisphere image (Fig. 4), a non-smooth image (Fig. 8) and a natural image (Fig. 11) contaminated with Gaussian noise of different standard deviations $\sigma$.

In Table 8, we show results for the four algorithms for all test images.

– In Fig. 5, we present surface plots of the restored results. For clear visual inspection, we show in Fig. 6 zoomed plots of one small surface region from Fig. 5. In Fig. 7, we compare four restored results along a line.

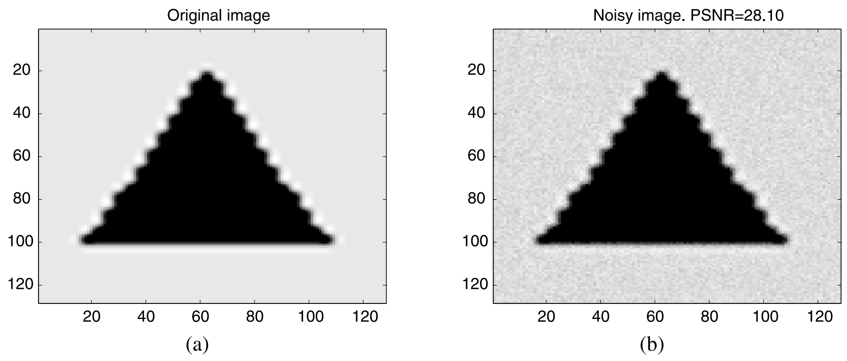


**Fig. 8** The “triangle” test image: (**a**) the original image; (**b**) the noisy image with the standard deviation $\sigma = 10$

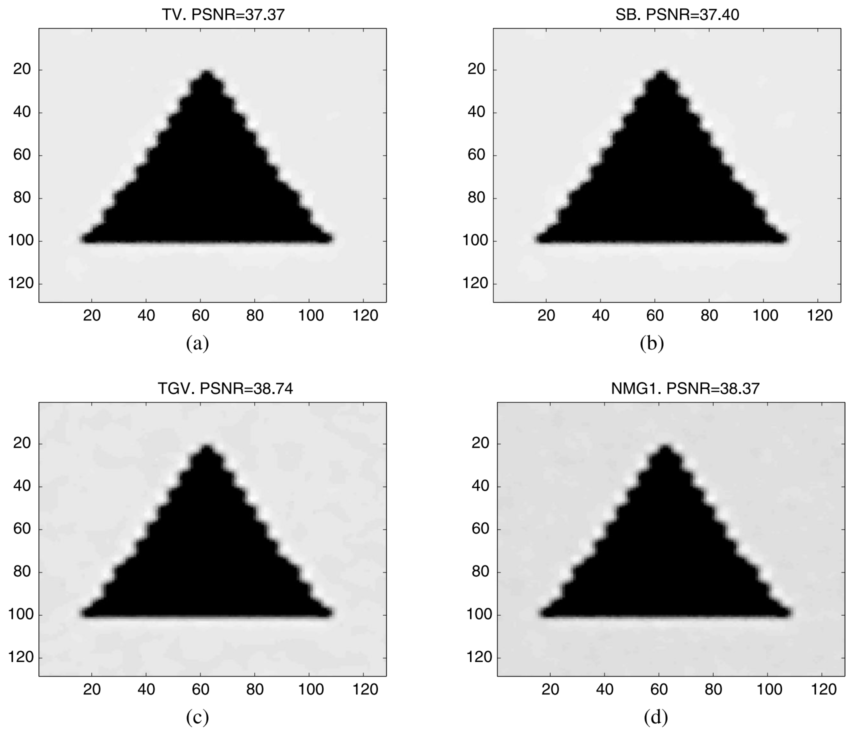
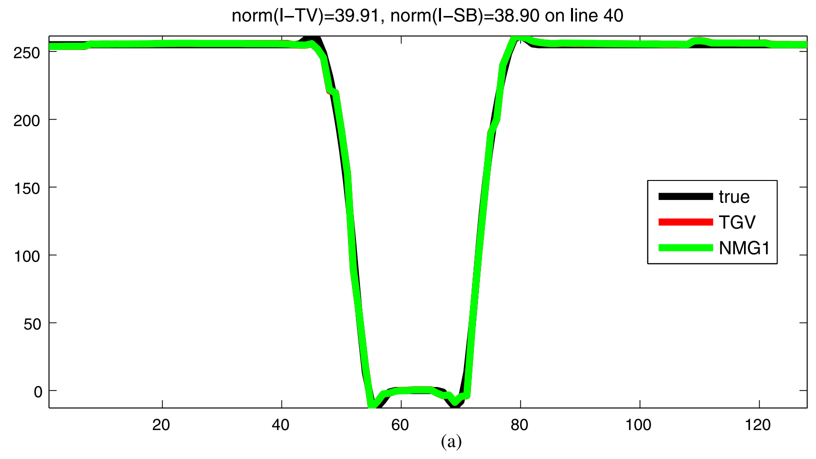


**Fig. 9** Comparison of restoration: (**a**) by TV; (**b**) by SB; (**c**) by TGV; (**d**) by NMG1

– From Fig. 8 to Fig. 13, we show the performance of the “triangle” image and “pepper” image. We can see TGV and NMG1 efficiently reduce the staircasing effects both on smooth and non-smooth images, as illustrated in Figs. 10 and 13.



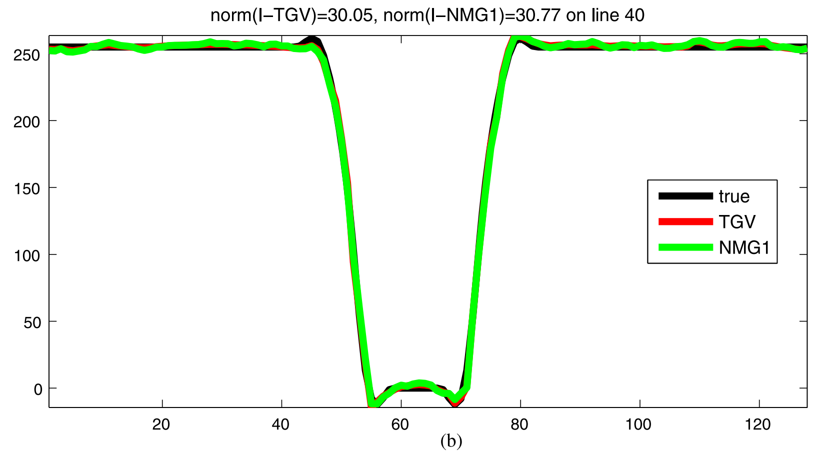


**Fig. 10** Comparison results along a line: (**a**) the TV model; (**b**) the higher order models

From Table 8 and Fig. 5–13, one can see that

- TGV/NMG1/SB are better than the TV as expected and NMG1/TGV give much better quality restorations than SB;
- both NMG1 and TGV can avoid undesirable staircasing effects and remove noise effectively;
- Our NMG1 produces a restoration quality comparable to that of the TGV.

## 4.3 Conclusions

In this paper we proposed a new solution method for solving the mean curvature-based model for image denoising using an AL formulation. Two stabilized fixed points algorithms (global GS and local GS) are developed and analyzed as suitable smoothers for a nonlinear multigrid method. Instead of using the usual Neumann boundary conditions, we derived and implemented the precise boundary conditions

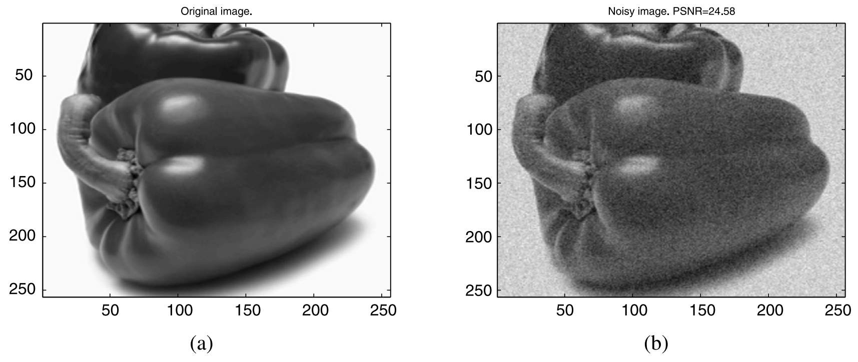


**Fig. 11** The “pepper” test images: (**a**) the original image; (**b**) the noisy image with the standard deviation $\sigma = 15$

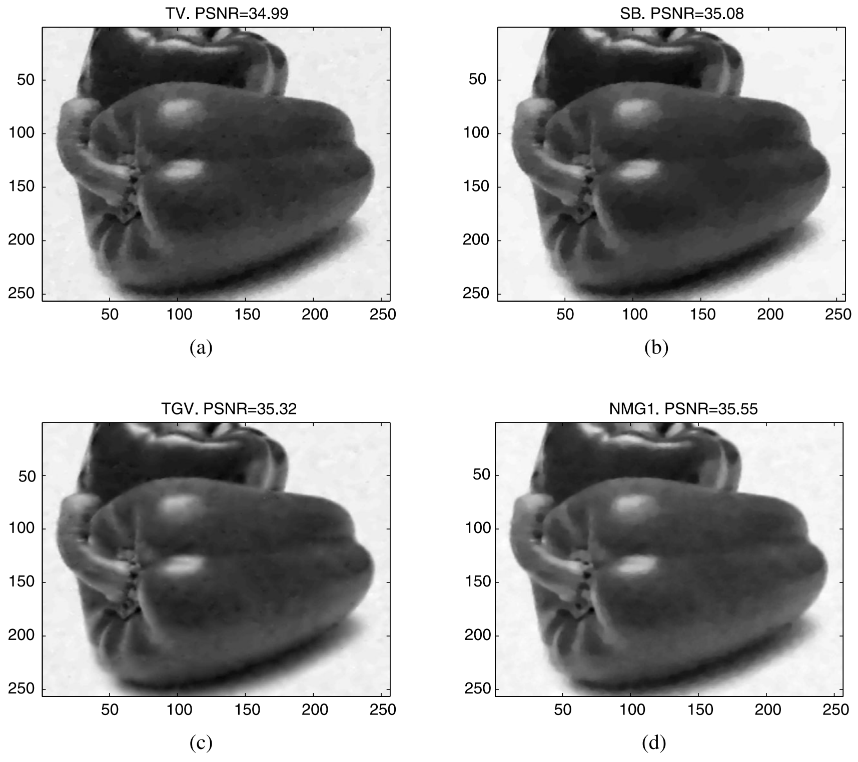
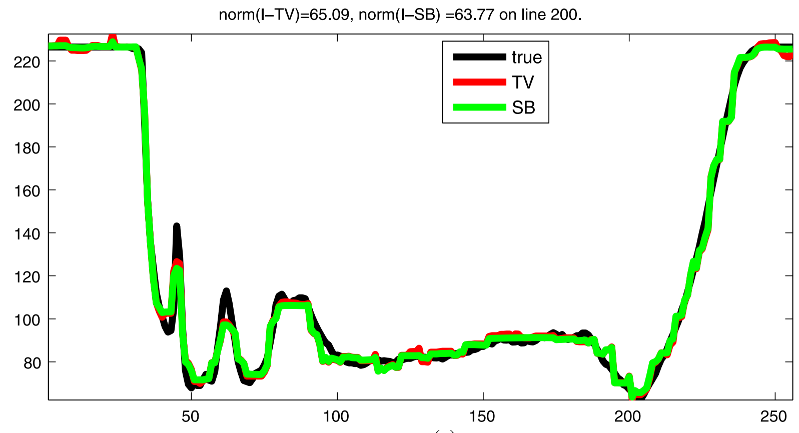
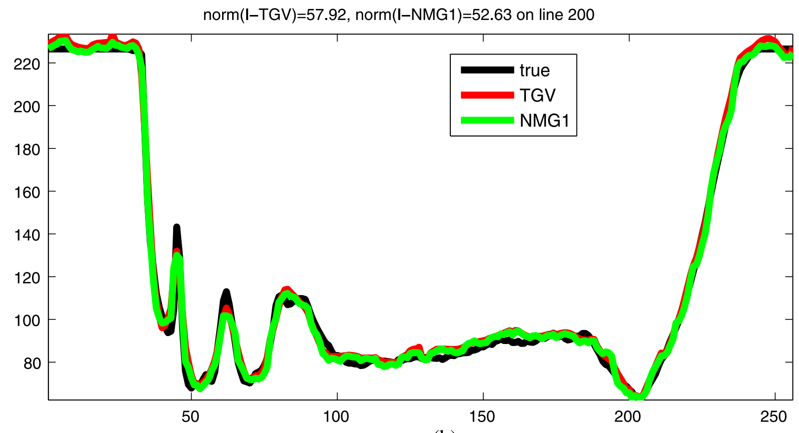


**Fig. 12** Comparison of restoration: (**a**) by TV; (**b**) by SB; (**c**) by TGV; (**d**) by NMG1

using staggered grids and locally coupled iterative solvers. Numerical results have shown that the new algorithm can deliver better quality of restoration than the previously fast method [4], use less regularisation of the nonlinearity and obtain a comparable restoration quality to the TGV model [2].



(a)



(b)

**Fig. 13** Comparison results along a line: (**a**) the TV model; (**b**) higher order models

## References

1. Blomgren, P., Mulet, P., Chan, T.F., Wong, C.K.: Total variation image restoration: numerical methods and extensions. In: Proceedings of the 1997 IEEE International Conference on Image Processing, Santa Barbara, CA, vol. 3, pp. 384–387 (1997)
2. Bredies, K., Kunisch, K., Pock, T.: Total generalized variation. SIAM J. Imaging Sci. **3**(3), 492–526 (2010)
3. Briggs, W.: A Multigrid Tutorial. SIAM, Philadelphia (1987)
4. Brito-Loeza, C., Chen, K.: Multigrid algorithm for high order denoising. SIAM J. Imaging Sci. **3**(3), 363–389 (2010)
5. Brito-Loeza, C., Chen, K.: On high-order denoising models and fast algorithms for vector-valued images. IEEE Trans. Image Process. **19**(6), 1518–1527 (2010)
6. Chen, K.: Matrix Preconditioning Techniques and Applications. Cambridge University Press, Cambridge (2005)
7. Chumchobn, N., Chen, K., Brito-Loeza, C.: Fourth order variational image registration on model and its fast multigrid algorithm. SIAM Multiscale Model. Simul. **9**(1), 89–128 (2011)
8. Chumchobn, N., Chen, K., Brito-Loeza, C.: Variational model for removal of combined additive and multiplicative noise. Int. J. Comput. Math. **90**(1), 140–161 (2013)

9. Eyre, D.J.: Unconditionally gradient stable time marching the Cahn-Hilliard equation. In: Bullard, J.W., et al. (eds.) Computational and Mathematical Models of Microstructural Evolution, pp. 39–46. Materials Research Society, Warrendale (1998)
10. Eyre, D.J.: An unconditionally stable one-step scheme for gradient systems. See www.math.utah.edu/~eyre/research/methods/stable.ps. Unpublished article (1998)
11. Gilboa, G., Osher, S.: Nonlocal operators with applications to image processing. SIAM Multiscale Model. Simul. **7**(3), 1005–1028 (2008)
12. Goldstein, T., Osher, S.: The split Bregman method for L1 regularized problems. SIAM J. Imaging Sci. **2**(2), 323–343 (2009)
13. Henn, S.: A multigrid method for a fourth-order diffusion equation with application to image processing. SIAM J. Sci. Comput. **27**(3), 831–849 (2005)
14. Lions, P.L., Mercier, B.: Splitting algorithms for the sum of two nonlinear operators. SIAM J. Numer. Anal. **16**(6), 964–979 (1979)
15. Lou, Y., Zhang, X., Osher, S., Bertozzi, A.: Image recovery via nonlocal operators. J. Sci. Comput. **42**(2), 185–197 (2010)
16. Lysaker, M., Osher, S., Tai, X.-C.: Noise removal using smoothed normals and surface fitting. IEEE Trans. Image Process. **13**(10), 1345–1357 (2004)
17. Osher, S., Burger, M., Goldfarb, D., Xu, J., Yin, W.: An iterative regularization method for total variation-based image restoration. Multiscale Model. Simul. **4**(2), 460–489 (2005)
18. Rudin, L., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D **60**, 259–268 (1992)
19. Shyy, W., Sue, C.-S.: Development of a pressure-correction/staggered-grid based multigrid solver for incompressible recirculating flows. Comput. Fluids **22**(1), 51–76 (1993)
20. Tai, X.-C., Hahn, J., Chung, G.J.: A fast algorithm for Euler's elastic method using augmented Lagrangian method. SIAM J. Imaging Sci. **4**(1), 313–344 (2011)
21. Trottenberg, U., Oosterlee, C., Schuller, A.: Multigrid. Academic Press, London (2001)
22. Wesseling, P.: An Introduction to Multigrid Methods. Wiley, Chichester (1992)
23. Wienands, R., Joppich, W.: Practical Fourier Analysis for Multigrid Methods. Chapman and Hall/CRC Press, Boca Raton (2005)
24. Wise, S., Kim, J., Lowengrub, J.: Solving the regularized, strongly anisotropic Cahn-Hilliard equation by an adaptive nonlinear multigrid method. J. Comput. Phys. **226**(1), 414–446 (2007)
25. Wu, C., Tai, X.-C.: Augmented Lagrangian method, dual methods, and split Bregman iteration for ROF, vectorial TV, and high order models. SIAM J. Imaging Sci. **3**(3), 300–339 (2010)
26. Yuille, A.L., Rangarajan, A.: The concave-convex procedure. Neural Comput. **15**, 915–936 (2003)
27. Zhang, J.P., Chen, K., Yu, B.: An iterative Lagrange multiplier method for constrained total-variation-based image denoising. SIAM J. Numer. Anal. **50**(3), 983–1003 (2012)
28. Zhu, W., Chan, T.F.: Image denoising using mean curvature. SIAM J. Imaging Sci. **5**(1), 1–32 (2012)
29. Zhu, W., Tai, X.-C., Chan, T.F.: Augmented Lagrangian method for a mean curvature based image denoising model. UCLA CAM report 12-02 (2012)