Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 7696798, 21 pages
https://doi.org/10.1155/2018/7696798

Research Article

Performance of the Restarted Homotopy Perturbation Method and Split Bregman Method for Multiplicative Noise Removal

Yu Du Han[1] **and Jae Heon Yun**[2]

[1]*Department of Mathematics, Chungbuk National University, Cheongju, 28644, Republic of Korea*
[2]*Department of Mathematics, College of Natural Sciences, Chungbuk National University, Cheongju, 28644, Republic of Korea*

Correspondence should be addressed to Jae Heon Yun; gmjae@chungbuk.ac.kr

Received 31 August 2018; Accepted 19 November 2018; Published 6 December 2018

Academic Editor: Thomas Schuster

Copyright © 2018 Yu Du Han and Jae Heon Yun. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In this paper, we first propose restarted homotopy perturbation methods (RHPM) for multiplicative noise removal of the RLO and AA2 models. The main difficulty in applying the RHPM to the nonlinear denoising problem is settled by using binomial series techniques. We next propose the split Bregman methods for multiplicative noise removal of the RLO and AA2 models. The difficulty in applying the split Bregman method to the nonlinear denoising problem can be handled by transforming ill-conditioned linear systems into well-conditioned linear systems using splitting techniques of singular matrices. Lastly, numerical experiments for several test problems are provided to demonstrate the efficiency and reliability of the RHPM and split Bregman methods.

1. Introduction

Image denoising which is one of the fundamental problems in image processing is to recover an original image from a given noisy image. Let Ω be an open rectangular domain in $\mathbb{R}^2$, let $z : \Omega \longrightarrow R$ be an observed noisy image to be denoised, and let u be the original unknown image to restore. The mathematical models of noisy images can be classified into additive or multiplicative ones according to the relations of noises and clean images, i.e.,

$$\begin{aligned} z &= u + \tilde{\eta} \\ \text{or } z &= u \cdot \eta, \end{aligned} \tag{1}$$

where $\tilde{\eta}$ and η are unknown additive and multiplicative noise functions, respectively. In this paper, we focus on multiplicative noise removal for the observed images corrupted by multiplicative Gaussian white noise or Gamma noise with mean equal to one.

There are a lot of investigations for additive model (1) using the following general nonlinear variational model [1, 2]:

$$\min_u \left\{ E(u) = \int_\Omega \varphi(|\nabla u|)\, dx + \frac{\alpha}{2} \int_\Omega (u - z)^2\, dx \right\}, \tag{2}$$

where the first term of the right-hand side is a regularizer in general form to ensure the smoothness of the desired image with edge preserving, the second term is a data fitting term which reflects the fidelity between the original image and observed noisy image, and $\alpha > 0$ is a regularization parameter.

Many approaches have been proposed to remove the additive noise, like variational approaches [3, 4], the stochastic approach [5, 6], wavelet approaches [7, 8], framelet approaches [9, 10], and RHPM approaches [11], to name a few. It is well known that the model introduced by Rudin, Osher, and Fatemi [12] (called the ROF model) is a popular model for image denoising due to its property of preserving edges. If $\varphi(|\nabla u|) = |\nabla u|$ in (2), then (2) reduces to the ROF model using Total Variation (TV):

$$\min_u \left\{ E(u) = \int_\Omega |\nabla u|\, dx + \frac{\alpha}{2} \int_\Omega (u - z)^2\, dx \right\}. \tag{3}$$

Since the images restored by the ROF model are in the space of bounded variation functions, the ROF model preserves sharp edges or object boundaries, and it is especially effective on restoring the piecewise constant images. However, the ROF model does not preserve the details and textures since it yields staircase effects.

Multiplicative noise models occur in the study of several coherent imaging systems, such as synthetic aperture radar (SAR) and sonar (SAS), ultrasound or laser imaging, and magnetic resonance imaging (MRI). The researches on variational models of additive noise removal have been successful, but the investigations on variational models of multiplicative noise removal have not been done much. The difference between a variational model of additive and that of multiplicative noise removal is that the data fitting term depends on noise distributions, such as Gamma, Poisson, Rayleigh, and Gauss distributions due to different means of image acquisition. SAR images are Gamma distributions [13], medical ultrasound or resonance images fit Rayleigh distributions [14], and astronomical microscopy images and medical SPECT/PET images fit Poisson distributions [15, 16]. If the multiplicative noise is not too strong, the abovementioned noise can be considered approximately to the Gauss distributions [17].

For the multiplicative Gaussian white noise with mean equal to one, Rudin, Lions, and Osher [17] proposed the following variational diffusion model (called the RLO model):

$$\min_u \left[E(u) = \int_\Omega |\nabla u| \, dx + \int_\Omega \left\{ \alpha \frac{z}{u} + \frac{\beta}{2} \left(\frac{z}{u} - 1 \right)^2 \right\} dx \right], \tag{4}$$

where α and β are weighted parameters.

For the multiplicative Gamma noise with mean equal to one, Jin and Yang [18] applied the exponential transformation $u \longrightarrow e^u$ introduced by Huang et al. [19] with the fitting term of the AA model [13] and proposed the following variational model (called the AA2 model in this paper):

$$\min_u \left\{ E(u) = \int_\Omega |\nabla u| \, dx + \alpha \int_\Omega \left(u + z \cdot e^{-u} \right) dx \right\}, \tag{5}$$

where α is a weighted parameter.

The investigations on the algorithm design of the classic TV model have attracted a lot of interest since it was invented to improve its computation efficiency and the quality of restored images, such as the artificial time marching method [12], fixed-point iterative method [20], primal-dual method [21], dual method [22], split method [23], Bregman iterative method [24], split Bregman method [25], and unbiased Box-Cox transformation method [26]. Among them, the split Bregman method combines the advantages of the split method in easy implementation and the Bregman iterative method in good quality of restored images. The variational models of multiplicative noise removal are more complex than the corresponding ones of additive noise removal, but its previous studies focused on models according to different noise distributions. Some researchers pay their attention to algorithm design, such as [12, 20, 27, 28]. The starting point of our investigation is the RLO model of (4) and AA2 model of (5) with the TV regularization term. The purpose of this paper is to propose restarted homotopy perturbation methods (RHPM) and split Bregman methods for multiplicative noise removal models (4) and (5).

The paper is organized as follows. In Section 2, we review the TM (time marching) method for multiplicative noise removal models. In Section 3, we briefly review a restarted homotopy perturbation method (RHPM). In Section 4, we propose RHPM algorithms for the multiplicative noise removal models. In Section 5, we propose split Bregman algorithms for the multiplicative noise removal models. In Section 6, we provide numerical experiments for several test problems in order to evaluate the performance of the RHPM and split Bregman methods. Lastly we provide concluding remarks.

2. Review of the TM Method for Multiplicative Noise Removal

The Euler-Lagrange equations corresponding to RLO model (4) and AA2 model (5) lead to the following nonlinear elliptic PDEs with the homogeneous Neumann boundary conditions, respectively:

$$\begin{aligned} &\nabla \cdot \left(\frac{\nabla u(x,y)}{|\nabla u(x,y)|} \right) + (\alpha - \beta) \frac{z(x,y)}{u(x,y)^2} + \beta \frac{z(x,y)^2}{u(x,y)^3} \\ &\quad = 0, \quad (x,y) \in \Omega, \\ &\nabla u(x,y) \cdot \vec{n} = 0, \quad (x,y) \in \partial\Omega, \end{aligned} \tag{6}$$

$$\begin{aligned} &\nabla \cdot \left(\frac{\nabla u(x,y)}{|\nabla u(x,y)|} \right) - \alpha \left\{ 1 - z(x,y) e^{-u(x,y)} \right\} = 0, \\ &\qquad (x,y) \in \Omega, \\ &\nabla u(x,y) \cdot \vec{n} = 0, \quad (x,y) \in \partial\Omega, \end{aligned} \tag{7}$$

where $\vec{n}$ is the unit normal vector exterior to the boundary $\partial\Omega$.

We briefly describe the TM (time marching) method [12] only for AA2 model (7). To avoid division by zero in numerical implementation, we replace the nondifferentiable term $|\nabla u(x,y)|$ in (7) with a smooth approximation term

$$|\nabla u(x,y)|_\varepsilon = \sqrt{|\nabla u(x,y)|^2 + \varepsilon} \quad \text{for a small } \varepsilon > 0. \tag{8}$$

Then (7) is transformed into

$$\begin{aligned} &\nabla \cdot \left(\frac{\nabla u(x,y)}{|\nabla u(x,y)|_\varepsilon} \right) - \alpha \left\{ 1 - z(x,y) e^{-u(x,y)} \right\} = 0, \\ &\qquad (x,y) \in \Omega, \\ &\nabla u(x,y) \cdot \vec{n} = 0, \\ &\qquad (x,y) \in \partial\Omega. \end{aligned} \tag{9}$$

In order to apply the TM method to (9), we consider the following time-dependent nonlinear parabolic PDE corresponding to (9):

1: MAE(0) = 255, $u_{i,j}^0 = z_{i,j}$ for $1 \le i \le m, 1 \le j \le n$
2: **for** $k = 0$ to *maxit* **do**
3: Compute $\kappa(u_{i,j}^k)$ using Equation (12) for $1 \le i \le m, 1 \le j \le n$
4: Compute $u_{i,j}^{k+1}$ using Equation (11) for $1 \le i \le m, 1 \le j \le n$
5: **if** MAE(k) < MAE($k+1$) **then**
6: Stop
7: **end if**
8: **end for**

Algorithm 1: TM algorithm for the AA2 model.

$$\frac{\partial u(x,y;t)}{\partial t} = \nabla \cdot \left(\frac{\nabla u(x,y;t)}{|\nabla u(x,y;t)|_\varepsilon} \right) - \alpha \left\{ 1 - z(x,y) e^{-u(x,y;t)} \right\}, \quad (x,y) \in \Omega \tag{10}$$

$$\nabla u \cdot \vec{n} = 0, \quad (x,y) \in \partial\Omega,$$

where $u(x,y,0)$ is given.

For numerical implementation, let us assume that the domain Ω has been split into $m \times n$ cells where the grid points are located at $(x_i = ih_x, y_j = jh_y)$, $1 \le i \le m$, $1 \le j \le n$, $t_k = k\triangle t$, where $\triangle t$ and $k = 1, 2, \ldots$ denote the time step and iteration number, respectively. We denote the values of $u(x,y;t)$ at the grid points $(x_i, y_j; t_k)$ by $u_{i,j}^k$ and $u_{i,j}^0 = z(x_i, y_j)$.

Without loss of generality, we can assume that $h_x = h_y = 1$. Then, nonlinear PDE (10) can be approximated by the following finite difference formula:

$$u_{i,j}^{k+1} = u_{i,j}^k + \triangle t \left\{ \kappa\left(u_{i,j}^k\right) - \alpha\left(1 - z_{i,j} e^{-u_{i,j}^k}\right) \right\}, \tag{11}$$

where for $1 \le i \le m$, $1 \le j \le n$

$$\begin{aligned} \kappa\left(u_{i,j}^k\right) &= \left[\nabla \cdot \left(\frac{\nabla u}{|\nabla u|_\varepsilon} \right) \right]_{i,j}^k = \left[\frac{\partial}{\partial x} \left(\frac{u_x}{|\nabla u|_\varepsilon} \right) + \frac{\partial}{\partial y} \left(\frac{u_y}{|\nabla u|_\varepsilon} \right) \right]_{i,j}^k \\ &= \left[\triangle_-^x \left(\frac{\triangle_+^x u_{i,j}^k}{\sqrt{\left(\triangle_+^x u_{i,j}^k\right)^2 + \left(\triangle_+^y u_{i,j}^k\right)^2 + \varepsilon}} \right) + \triangle_-^y \left(\frac{\triangle_+^y u_{i,j}^k}{\sqrt{\left(\triangle_+^x u_{ij}^k\right)^2 + \left(\triangle_+^y u_{i,j}^k\right)^2 + \varepsilon}} \right) \right]_{i,j} \end{aligned} \tag{12}$$

with

$$\begin{aligned} \triangle_+^x u_{i,j}^k &= u_{i+1,j}^k - u_{i,j}^k, \\ \triangle_-^x u_{i,j}^k &= -u_{i-1,j}^k + u_{i,j}^k \\ \triangle_+^y u_{i,j}^k &= u_{i,j+1}^k - u_{i,j}^k, \\ \triangle_-^y u_{i,j}^k &= -u_{i,j-1}^k + u_{i,j}^k, \\ u_{0,j}^k &= u_{1,j}^k, \\ u_{m+1,j}^k &= u_{m,j}^k, \\ u_{i,0}^k &= u_{i,1}^k, \\ u_{i,n+1}^k &= u_{i,n}^k. \end{aligned} \tag{13}$$

In a similar way as was done for the AA2 model, we can obtain the following formula for the RLO model:

$$u_{i,j}^{k+1} = u_{i,j}^k + \triangle t \left[\kappa\left(u_{i,j}^k\right) + \frac{z_{i,j}\left\{(\alpha - \beta) u_{i,j}^k + \beta z_{i,j}\right\}}{\left(u_{i,j}^k\right)^3 + \varepsilon} \right], \tag{14}$$

where $\kappa(u_{i,j}^k)$ and the boundary conditions are defined the same as those in the AA2 model.

Hence, we obtain TM Algorithms 1 and 2 for the AA2 and RLO models. *maxit* denotes the maximum number of iterations, z denotes the noisy image, u^k denotes the restored image at the kth iteration, and MAE(k) denotes the *mean absolute error* at the kth iteration, i.e.,

$$\mathrm{MAE}(k) = \frac{\left\| u^k - u^{k-1} \right\|_1}{m \cdot n}, \tag{15}$$

where $\| \cdot \|_1$ stands for the l_1-norm.

3. Review of the Restarted Homotopy Perturbation Method (RHPM)

We briefly describe the homotopy perturbation method (HPM) for solving nonlinear partial differential equation

$$A(u) - f(r) = 0, \quad r \in \Omega \tag{16}$$

with the boundary conditions

$$B\left(u, \frac{\partial u}{\partial n}\right) = 0, \quad r \in \Gamma, \tag{17}$$

1: MAE(0) = 255, $u_{i,j}^0 = z_{i,j}$ for $1 \le i \le m, 1 \le j \le n$
2: **for** $k = 0$ to *maxit* **do**
3: Compute $\kappa(u_{i,j}^k)$ using Equation (12) for $1 \le i \le m, 1 \le j \le n$
4: Compute $u_{i,j}^{k+1}$ using Equation (14) for $1 \le i \le m, 1 \le j \le n$
5: **if** MAE(k) < MAE($k+1$) **then**
6: Stop
7: **end if**
8: **end for**

Algorithm 2: TM algorithm for the RLO model.

where A is a general differential operator, B is a boundary operator, $f(r)$ is a known analytic function, and Γ is the boundary of the domain Ω. The operator A can be divided into two parts L and N, where L is linear and N is nonlinear. Therefore, (16) can be written as

$$L(u) + N(u) - f(r) = 0. \tag{18}$$

By using the homotopy technique, one can construct a homotopy $v(r, p) : \Omega \times [0, 1] \longrightarrow \mathbb{R}$ which satisfies

$$\begin{aligned} H(v, p) &= (1 - p)[L(v) - L(u_0)] + p[A(v) - f(r)] \\ &= 0, \end{aligned} \tag{19}$$

or, equivalently,

$$\begin{aligned} H(v, p) &= L(v) - L(u_0) + pL(u_0) \\ &\quad + p[N(v) - f(r)] = 0, \end{aligned} \tag{20}$$

where $p \in [0, 1]$ is an embedding parameter and u_0 is the initial approximation of (16) which satisfies the boundary conditions. Clearly, we have

$$\begin{aligned} H(v, 0) &= L(v) - L(u_0) = 0, \\ H(v, 1) &= A(v) - f(r) = 0. \end{aligned} \tag{21}$$

The changing process of p from zero to unity is just that of $v(r, p)$ changing from $u_0(r)$ to $u(r)$. This is called deformation and also $L(v) - L(u_0)$ and $A(v) - f(r)$ are called homotopic in topology. If the embedding parameter $0 \le p \le 1$ is considered as a "small parameter" applying the classical perturbation technique, then the solution of (20) can be expressed as a power series in p, that is,

$$v = v(r, p) = v_0 + pv_1 + p^2 v_2 + \cdots. \tag{22}$$

Setting $p \longrightarrow 1$, the solution u of (16) can be expressed as the sum of an infinite series

$$u = \lim_{p \longrightarrow 1} v(r, p) = v_0 + v_1 + v_2 + \cdots. \tag{23}$$

Generally speaking, the computation of v_k in the HPM becomes more complicated as k increases. To circumvent this problem, the restarted homotopy perturbation method (RHPM) proposed by Han and Yun [11] repeats the HPM process by computing only the first few terms instead of computing infinite terms of v_k. This is a big advantage of the RHPM as compared with the HPM. The general procedure of the RHPM for solving a nonlinear equation is described below (see [11] for more details).

Let u_0^0 be an initial approximation of a nonlinear equation. First, we compute the first k-th term approximation u_0^1 in the following way:

$$\begin{aligned} v_0^0 &= u_0^0 \\ &\text{Compute } v_1^0, v_2^0, \ldots, v_k^0 \text{ using (20) and (22)} \\ u_0^1 &= v_0^0 + v_1^0 + v_2^0 + \cdots + v_k^0. \end{aligned} \tag{24}$$

Following the above computational process with u_0^1 as an initial approximation, a new approximation u_0^2 is computed as follows:

$$\begin{aligned} v_0^1 &= u_0^1 \\ &\text{Compute } v_1^1, v_2^1, \ldots, v_k^1 \text{ using (20) and (22)} \\ u_0^2 &= v_0^1 + v_1^1 + v_2^1 + \cdots + v_k^1. \end{aligned} \tag{25}$$

By repeating the above process, a new approximation $u_0^{\ell+1}$ can be computed from the initial approximation u_0^ℓ, which was obtained at the ℓth step, in the following way:

$$\begin{aligned} v_0^\ell &= u_0^\ell \\ &\text{Compute } v_1^\ell, v_2^\ell, \ldots, v_k^\ell \text{ using (20) and (22)} \\ u_0^{\ell+1} &= v_0^\ell + v_1^\ell + v_2^\ell + \cdots + v_k^\ell. \end{aligned} \tag{26}$$

Notice that $v_0^\ell, v_1^\ell, \ldots, v_k^\ell$ are coefficients of $p^0, p^1, \ldots, p^k$, respectively. Then the exact solution u can be approximated by $u_0^{\ell+1}$, that is, under some appropriate conditions $u = \lim_{\ell \longrightarrow \infty} u_0^\ell$. The RHPM which computes only k vectors $v_1^\ell, v_2^\ell, \ldots, v_k^\ell$ every step is called the RHPM(k) method.

4. RHPM Algorithms for Multiplicative Noise Removal

In this section, we describe an application of the RHPM to multiplicative denoising problems of the RLO and AA2

models. Before doing this, we describe the binomial series which is used for an application of the RHPM. The difficulty in applying the RHPM to the TV-based image denoising problem comes from (36) and (62) (i.e., $(v_x^2 + v_y^2)^{-1/2}$, $(v_x^2 + v_y^2)^{-3/2}$, u^{-2}, and u^{-3}). In order to handle this difficulty, the following binomial series is used: for $|x| < 1$,

$$\begin{aligned}(1+x)^n &= \sum_{k=0}^{\infty} \binom{n}{k} x^k \\ &= 1 + nx + \frac{n(n-1)}{2!}x^2 + \frac{n(n-1)(n-2)}{3!}x^3 \\ &\quad + \cdots.\end{aligned} \tag{27}$$

If $n = -1/2$ in (27), then

$$(1+x)^{-1/2} = \frac{1}{\sqrt{1+x}} = 1 - \frac{1}{2}x + \frac{3}{8}x^2 - \frac{5}{16}x^3 + \cdots. \tag{28}$$

Letting $x = \tau - 1$ in (28), for $0 < \tau < 2\tau^{-1/2}$ can be approximated by

$$\begin{aligned}\tau^{-1/2} = \frac{1}{\sqrt{\tau}} &\approx 1 - \frac{1}{2}(\tau - 1) + \frac{3}{8}(\tau - 1)^2 \\ &= \frac{15}{8} - \frac{5}{4}\tau + \frac{3}{8}\tau^2.\end{aligned} \tag{29}$$

If $\tau > 0$ is large, then, by introducing a small parameter $\varepsilon > 0$ and using (29),

$$\tau^{-1/2} = \frac{1}{\sqrt{\tau}} = \frac{\sqrt{\varepsilon}}{\sqrt{\varepsilon\tau}} \approx \sqrt{\varepsilon}\left(\frac{15}{8} - \frac{5\varepsilon}{4}\tau + \frac{3\varepsilon^2}{8}\tau^2\right). \tag{30}$$

Similarly, if $n = -3/2$ and $x = \tau - 1$ in (27), then, by introducing a small parameter $\varepsilon > 0$,

$$\begin{aligned}\tau^{-3/2} = \frac{1}{\sqrt{\tau^3}} &= \frac{\sqrt{\varepsilon^3}}{\sqrt{(\varepsilon\tau)^3}} \\ &\approx \sqrt{\varepsilon^3}\left(\frac{35}{8} - \frac{21\varepsilon}{4}\tau + \frac{15\varepsilon^2}{8}\tau^2\right).\end{aligned} \tag{31}$$

Note that a small parameter $\varepsilon > 0$ in (30) and (31) is used to guarantee convergence of the binomial series when $\tau > 0$ is large.

We first describe how to apply the RHPM to the following time-dependent nonlinear parabolic PDE corresponding to AA2 model (7):

$$\begin{aligned}\frac{\partial u(x,y;t)}{\partial t} &= \nabla \cdot \left(\frac{\nabla u(x,y;t)}{|\nabla u(x,y;t)|}\right) \\ &\quad - \alpha\left\{1 - z(x,y)\, e^{-u(x,y;t)}\right\}, \\ &\qquad (x,y) \in \Omega \\ \nabla u \cdot \vec{n} &= 0, \quad (x,y) \in \partial\Omega.\end{aligned} \tag{32}$$

For simplicity of exposition, we describe the computational process for RHPM(2) (i.e., $k = 2$). We use three terms of binomial series (27) for numerical computation of (36) and (62), so division by zero does not occur. Let the operators L, N, and f be defined as

$$\begin{aligned}L(v) &= \frac{\partial v(x,y;t)}{\partial t}, \quad f(x,y) = 0, \\ N(v) &= -\nabla \cdot \left(\frac{\nabla v(x,y;t)}{|\nabla v(x,y;t)|}\right) \\ &\quad + \alpha\left\{1 - z(x,y)\, e^{-v(x,y;t)}\right\},\end{aligned} \tag{33}$$

where

$$\begin{aligned}\nabla v(x,y;t) &= \left(\frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}\right)^T, \\ |\nabla v(x,y;t)| &= \sqrt{(v_x)^2 + (v_y)^2}.\end{aligned} \tag{34}$$

From (20), one obtains

$$\begin{aligned}\frac{\partial v}{\partial t} &= \frac{\partial u_0}{\partial t} - p\frac{\partial u_0}{\partial t} + p\nabla \cdot \left(\frac{\nabla v(x,y;t)}{|\nabla v(x,y;t)|}\right) \\ &\quad - p\alpha\left\{1 - z(x,y)\, e^{-v(x,y;t)}\right\},\end{aligned} \tag{35}$$

where

$$\begin{aligned}\nabla \cdot \left(\frac{\nabla v}{|\nabla v|}\right) &= \frac{v_{xx} + v_{yy}}{\sqrt{(v_x)^2 + (v_y)^2}} \\ &\quad - \frac{(v_x)^2 v_{xx} + (v_y)^2 v_{yy}}{\sqrt{\left\{(v_x)^2 + (v_y)^2\right\}^3}}.\end{aligned} \tag{36}$$

Using the Taylor series expansion of e^{-v} with the second order and power series expansion of v of form (22), e^{-v} can be approximated as

$$\begin{aligned}e^{-v} &\approx 1 - v + \frac{1}{2}v^2 \\ &= 1 - \left(v_0 + pv_1 + p^2 v_2 + \cdots\right) \\ &\quad + \frac{1}{2}\left(v_0 + pv_1 + p^2 v_2 + \cdots\right)^2 \\ &= \left\{1 - v_0 + \frac{1}{2}(v_0)^2\right\} - p(1 - v_0)v_1 + \cdots.\end{aligned} \tag{37}$$

Since the solution v of (35) is of form (22), by simple computation

$$\begin{aligned}v_x &= (v_0)_x + p(v_1)_x + p^2(v_2)_x + \cdots, \\ v_{xx} &= (v_0)_{xx} + p(v_1)_{xx} + p^2(v_2)_{xx} + \cdots, \\ v_y &= (v_0)_y + p(v_1)_y + p^2(v_2)_y + \cdots, \\ v_{yy} &= (v_0)_{yy} + p(v_1)_{yy} + p^2(v_2)_{yy} + \cdots,\end{aligned}$$

$$\begin{aligned}(v_x)^2 &= (v_0)_x^2 + 2\{(v_0)_x (v_1)_x\}\, p + \cdots,\\ (v_y)^2 &= (v_0)_y^2 + 2\{(v_0)_y (v_1)_y\}\, p + \cdots.\end{aligned} \tag{38}$$

From these equations, one easily obtains

$$\begin{aligned}(v_x)^2 + (v_y)^2 &= (D_1) + 2(D_2)\, p + \cdots,\\ \left\{(v_x)^2 + (v_y)^2\right\}^2 &= (D_1)^2 + 4(D_1)(D_2)\, p + \cdots,\\ v_{xx} + v_{yy} &= (D_3) + (D_4)\, p + \cdots,\\ (v_x)^2 v_{xx} + (v_y)^2 v_{yy} &= (D_5) + (D_6)\, p + \cdots,\end{aligned} \tag{39}$$

where

$$\begin{aligned}(D_1) &\overleftarrow{\equiv} (v_0)_x^2 + (v_0)_y^2,\\ (D_2) &\overleftarrow{\equiv} (v_0)_x (v_1)_x + (v_0)_y (v_1)_y,\\ (D_3) &\overleftarrow{\equiv} (v_0)_{xx} + (v_0)_{yy},\\ (D_4) &\overleftarrow{\equiv} (v_1)_{xx} + (v_1)_{yy},\\ (D_5) &\overleftarrow{\equiv} (v_0)_x^2 (v_0)_{xx} + (v_0)_y^2 (v_0)_{yy},\\ (D_6) &\overleftarrow{\equiv} (v_0)_x^2 (v_1)_{xx} + (v_0)_y^2 (v_1)_{yy}\\ &\quad + 2(v_0)_x (v_0)_{xx} (v_1)_x\\ &\quad + 2(v_0)_y (v_0)_{yy} (v_1)_y.\end{aligned} \tag{40}$$

Substituting $\tau = (v_x)^2 + (v_y)^2$ into (30) and using (39), one obtains

$$\begin{aligned}\frac{1}{|\nabla v|} = \frac{\sqrt{\varepsilon}}{\sqrt{\varepsilon\tau}} &= \frac{\sqrt{\varepsilon}}{\sqrt{\varepsilon\left\{(v_x)^2 + (v_y)^2\right\}}} \approx \sqrt{\varepsilon}\left[\frac{15}{8}\right.\\ &\quad \left. - \frac{5\varepsilon}{4}\left\{(v_x)^2 + (v_y)^2\right\} + \frac{3\varepsilon^2}{8}\left\{(v_x)^2 + (v_y)^2\right\}^2\right]\\ &= \sqrt{\varepsilon}\left\{\frac{15}{8} - \frac{5\varepsilon}{4}(D_1) + \frac{3\varepsilon^2}{8}(D_1)^2\right\} + \sqrt{\varepsilon^3}\left\{-\frac{5}{2}\right.\\ &\quad \left. + \frac{3\varepsilon}{2}(D_1)\right\}(D_2)\, p + \cdots = (A_1) + (B_1)(D_2)\, p\\ &\quad + \cdots,\end{aligned} \tag{41}$$

where $(A_1) \overleftarrow{\equiv} \sqrt{\varepsilon}\{15/8 - (5\varepsilon/4)(D_1) + (3\varepsilon^2/8)(D_1)^2\}$ and $(B_1) \overleftarrow{\equiv} \sqrt{\varepsilon^3}\{-5/2 + (3\varepsilon/2)(D_1)\}$.

Hence, the first term of the right-hand side in (36) can be written as

$$\begin{aligned}\frac{v_{xx} + v_{yy}}{|\nabla v|} &\approx (A_1)(D_3) + (A_1)(D_4)\, p\\ &\quad + (B_1)(D_2)(D_3)\, p + \cdots.\end{aligned} \tag{42}$$

Substituting $\tau = (v_x)^2 + (v_y)^2$ into (31) and using (39), one obtains

$$\begin{aligned}\frac{1}{|\nabla v|^3} = \frac{\sqrt{\varepsilon^3}}{\sqrt{(\varepsilon\tau)^3}} &= \frac{\sqrt{\varepsilon^3}}{\sqrt{\varepsilon^3\left\{(v_x)^2 + (v_y)^2\right\}^3}} \approx \sqrt{\varepsilon^3}\left[\frac{35}{8}\right.\\ &\quad \left. - \frac{21\varepsilon}{4}\left\{(v_x)^2 + (v_y)^2\right\} + \frac{15\varepsilon^2}{8}\left\{(v_x)^2 + (v_y)^2\right\}^2\right]\\ &= \sqrt{\varepsilon^3}\left\{\frac{35}{8} - \frac{21\varepsilon}{4}(D_1) + \frac{15\varepsilon^2}{8}(D_1)^2\right\}\\ &\quad + \sqrt{\varepsilon^5}\left\{-\frac{21}{2} + \frac{15\varepsilon}{2}(D_1)\right\}(D_2)\, p + \cdots = (A_2)\\ &\quad + (B_2)(D_2)\, p + \cdots,\end{aligned} \tag{43}$$

where $(A_2) \overleftarrow{\equiv} \sqrt{\varepsilon^3}\{35/8 - (21\varepsilon/4)(D_1) + (15\varepsilon^2/8)(D_1)^2\}$ and $(B_2) \overleftarrow{\equiv} \sqrt{\varepsilon^5}\{-21/2 + (15\varepsilon/2)(D_1)\}$.

Hence, the second term of the right-hand side in (36) can be written as

$$\begin{aligned}&\frac{(v_x)^2 v_{xx} + (v_y)^2 v_{yy}}{|\nabla v|^3}\\ &\quad \approx (A_2)(D_5) + (B_2)(D_2)(D_5)\, p + (A_2)(D_6)\, p\\ &\qquad + \cdots.\end{aligned} \tag{44}$$

From (36), (37), (42), and (44), (35) can be expressed as

$$\begin{aligned}&\frac{\partial v_0}{\partial t} + p\frac{\partial v_1}{\partial t} + p^2\frac{\partial v_2}{\partial t} + \cdots = \frac{\partial u_0}{\partial t} - p\frac{\partial u_0}{\partial t}\\ &\quad + p\left(\frac{v_{xx} + v_{yy}}{|\nabla n v|} - \frac{(v_x)^2 v_{xx} + (v_x)^2 v_{yy}}{|\nabla v|^3}\right) - p\left[1\right.\\ &\quad \left. - z\left\{1 - v_0 + \frac{1}{2}(v_0)^2\right\}\right]\alpha - p^2 z(1 - v_0) v_1 \alpha\\ &\quad \approx \frac{\partial u_0}{\partial t} - p\left[\frac{\partial u_0}{\partial t} - \{(A_1)(D_3) - (A_2)(D_5)\}\right]\\ &\quad + p^2\left[(A_1)(D_4) - (A_2)(D_6)\right.\\ &\quad \left. + \{(B_1)(D_3) - (B_2)(D_5)\}(D_2)\right] - p\alpha\left[1\right.\\ &\quad \left. - z\left\{1 - v_0 + \frac{1}{2}(v_0)^2\right\}\right] - p^2\alpha z(1 - v_0) v_1 + \cdots.\end{aligned} \tag{45}$$

Comparing coefficients of p^0, p^1, and p^2 in (44),

$$p^0 : \frac{\partial v_0}{\partial t} = \frac{\partial u_0}{\partial t} \Longrightarrow v_0 = u_0 = z : \tag{46}$$

given an initial approximation

$$p^1 : \frac{\partial v_1}{\partial t} = -\frac{\partial v_0}{\partial t} + \{(A_1)(D_3) - (A_2)(D_5)\} - \alpha\left[1\right. \left. - z\left\{1 - v_0 + \frac{1}{2}(v_0)^2\right\}\right] \tag{47}$$

$$p^2 : \frac{\partial v_2}{\partial t} = [(A_1)(D_4) - (A_2)(D_6) + \{(B_1)(D_3) - (B_2)(D_5)\}(D_2)] - \alpha z(1 - v_0)v_1. \tag{48}$$

Notice that the initial approximation u_0 is set to z which is a noisy image. Applying the inverse operator of L to (47), one obtains

$$v_1 = (z - v_0) + \left[\{(A_1)(D_3) - (A_2)(D_5)\} - \alpha\left[1 - z\left\{1 - v_0 + \frac{1}{2}(v_0)^2\right\}\right]\right]t = (v_{10}) + (v_{11})t, \tag{49}$$

where

$$(v_{10}) \overleftarrow{=} z - v_0,$$
$$(v_{11}) \overleftarrow{=} \{(A_1)(D_3) - (A_2)(D_5)\} - \alpha\left[1 - z\left\{1 - v_0 + \frac{1}{2}(v_0)^2\right\}\right]. \tag{50}$$

From (49), the partial derivatives of v_1 are

$$(v_1)_x = (v_{10})_x + t(v_{11})_x,$$
$$(v_1)_y = (v_{10})_y + t(v_{11})_y,$$
$$(v_1)_{xx} = (v_{10})_{xx} + t(v_{11})_{xx},$$
$$(v_1)_{yy} = (v_{10})_{yy} + t(v_{11})_{yy}. \tag{51}$$

Using (51), (D_2), (D_4), and (D_6) can be expressed as

$$(D_2) = \{(v_0)_x(v_{10})_x + (v_0)_y(v_{10})_y\} + \{(v_0)_x(v_{11})_x + (v_0)_y(v_{11})_y\}t = (D_{20}) + (D_{21})t,$$
$$(D_4) = \{(v_{10})_{xx} + (v_{10})_{yy}\} + \{(v_{11})_{xx} + (v_{11})_{yy}\}t = (D_{40}) + (D_{41})t,$$
$$(D_6) = [(v_0)_x^2(v_{10})_{xx} + (v_0)_y^2(v_{10})_{yy} + 2\{(v_0)_x(v_0)_{xx}(v_{10})_x + (v_0)_y(v_0)_{yy}(v_{10})_y\}] + [(v_0)_x^2(v_{11})_{xx} + (v_0)_y^2(v_{11})_{yy} + 2\{(v_0)_x(v_0)_{xx}(v_{11})_x + (v_0)_y(v_0)_{yy}(v_{11})_y\}]t = (D_{60}) + (D_{61})t, \tag{52}$$

where

$$(D_{20}) \overleftarrow{=} (v_0)_x(v_{10})_x + (v_0)_y(v_{10})_y,$$
$$(D_{21}) \overleftarrow{=} (v_0)_x(v_{11})_x + (v_0)_y(v_{11})_y,$$
$$(D_{40}) \overleftarrow{=} (v_{10})_{xx} + (v_{10})_{yy},$$
$$(D_{41}) \overleftarrow{=} (v_{11})_{xx} + (v_{11})_{yy},$$
$$(D_{60}) \overleftarrow{=} (v_0)_x^2(v_{10})_{xx} + (v_0)_y^2(v_{10})_{yy} + 2\{(v_0)_x(v_0)_{xx}(v_{10})_x + (v_0)_y(v_0)_{yy}(v_{10})_y\},$$
$$(D_{61}) \overleftarrow{=} (v_0)_x^2(v_{11})_{xx} + (v_0)_y^2(v_{11})_{yy} + 2\{(v_0)_x(v_0)_{xx}(v_{11})_x + (v_0)_y(v_0)_{yy}(v_{11})_y\}. \tag{53}$$

Substituting (49) and (52) into (48) and applying the inverse operator of L to (48), one obtains

$$v_2 = [(A_1)(D_{40}) - (A_2)(D_{60}) + \{(B_1)(D_3) - (B_2)(D_5)\}(D_{20})]t + \frac{1}{2}[(A_1)(D_{41}) - (A_2)(D_{61}) + \{(B_1)(D_3) - (B_2)(D_5)\}(D_{21})]t^2 - \alpha z\{1 - v_0\} \cdot (v_{10})t - \frac{\alpha}{2}z\{1 - v_0\}(v_{11})t^2 = (v_{21})t + (v_{22})t^2, \tag{54}$$

where

$$(v_{21}) \overleftarrow{=} [(A_1)(D_{40}) - (A_2)(D_{60}) + \{(B_1)(D_3) - (B_2)(D_5)\}(D_{20})] - \alpha z\{1 - v_0\} \cdot (v_{10}),$$
$$(v_{22}) \overleftarrow{=} \frac{1}{2}[(A_1)(D_{41}) - (A_2)(D_{61}) + \{(B_1)(D_3) - (B_2)(D_5)\}(D_{21})] - \frac{\alpha}{2}z\{1 - v_0\} \cdot (v_{11}). \tag{55}$$

Now update u_0 whose new value is set to $v_0 + v_1 + v_2$. By repeating the aforementioned process with the new updated u_0, we obtain the RHPM(2) method for the AA2 model.

For numerical implementation of the RHPM(2) method, let us assume that the domain Ω has been split into $m \times n$ cells where the grid points are located at $(x_i = ih_x, y_j = jh_y)$, $1 \le i \le m$, $1 \le j \le n$. Let $t_k = k\triangle t$, where $\triangle t$ and $k = 1, 2, \ldots$ refer to the time step and iteration number, respectively. We denote the values of $u(x, y; t)$ at the grid points $(x_i, y_j; t_k)$ by $(u^k)_{i,j}$. Without loss of generality, we can assume that $h = h_x = h_y = 1$. For simplicity of exposition, we define the following notation:

$$v_0^k = \left\{\left(v_0^k\right)_{i,j} = v_0\left(x_i, y_j; t_k\right) \mid 1 \le i \le m,\ 1 \le j \le n,\ k = 1, 2, \ldots\right\},$$

$$v_1^k = \left\{\left(v_1^k\right)_{i,j} = v_1\left(x_i, y_j; t_k\right) \mid 1 \le i \le m,\ 1 \le j \le n,\ k = 1, 2, \ldots\right\},$$
$$v_2^k = \left\{\left(v_2^k\right)_{i,j} = v_2\left(x_i, y_j; t_k\right) \mid 1 \le i \le m,\ 1 \le j \le n,\ k = 1, 2, \ldots\right\}. \tag{56}$$

The boundary conditions are defined the same as those in the TM method. Assume that $(u_0^0)_{i,j} = z(x_i, y_j) = z_{i,j}$ and $(u_0^k)_{i,j}$ has been computed for all i and j. Then $(v_0^k)_{i,j} = (u_0^k)_{i,j}$ and discretization of (49) is given by

$$\left(v_1^k\right)_{i,j} = (v_{10})_{i,j} + (v_{11})_{i,j} \cdot t_k, \tag{57}$$

where

$$(v_{10})_{i,j} = z_{i,j} - \left(v_0^k\right)_{i,j},$$
$$(v_{11})_{i,j} = \left\{(A_1)_{i,j}(D_3)_{i,j} - (A_2)_{i,j}(D_5)_{i,j}\right\} - \alpha\left[1 - z_{i,j}\left\{1 - \left(v_0^k\right)_{i,j} + \frac{1}{2}\left(v_0^k\right)_{i,j}^2\right\}\right]. \tag{58}$$

Also, discretization of (54) is given by

$$\left(v_2^k\right)_{i,j} = (v_{21})_{i,j} \cdot t_k + (v_{22})_{i,j} \cdot (t_k)^2, \tag{59}$$

where

$$(v_{21})_{i,j} = \left[(A_1)_{i,j}(D_{40})_{i,j} - (A_2)_{i,j}(D_{60})_{i,j} + \left\{(B_1)_{i,j}(D_3)_{i,j} - (B_2)_{i,j}(D_5)_{i,j}\right\}(D_{20})_{i,j}\right] - \alpha z_{i,j}\left\{1 - \left(v_0^k\right)_{i,j}\right\}(v_{10})_{i,j},$$
$$(v_{22})_{i,j} = \frac{1}{2}\left[(A_1)_{i,j}(D_{41})_{i,j} - (A_2)_{i,j}(D_{61})_{i,j} + \left\{(B_1)_{i,j}(D_3)_{i,j} - (B_2)_{i,j}(D_5)_{i,j}\right\}(D_{21})_{i,j}\right] - \frac{\alpha}{2} \cdot z_{i,j}\left\{1 - \left(v_0^k\right)_{i,j}\right\}(v_{11})_{i,j}. \tag{60}$$

We now compute $u_0^{k+1} = v_0^k + v_1^k + v_2^k$. Repeating the above process with u_0^{k+1} as an initial approximation, we can obtain the RHPM(2) method, called Algorithm 3, for multiplicative noise removal of the AA2 model.

We next describe how to apply RHPM(2) to the following time-dependent nonlinear parabolic PDE corresponding to RLO model (6):

$$\frac{\partial u(x, y; t)}{\partial t} = \nabla \cdot \left(\frac{\nabla u(x, y; t)}{\left|\nabla u(x, y; t)\right|}\right) + (\alpha - \beta)\frac{z(x, y)}{u(x, y; t)^2} + \beta\frac{z(x, y)^2}{u(x, y; t)^3}, \quad (x, y) \in \Omega$$
$$\nabla u \cdot \vec{n} = 0, \quad (x, y) \in \partial\Omega, \tag{61}$$

where $u(x, y, 0)$ is given. Let the operators L, N, and f be defined as

$$L(v) = \frac{\partial v(x, y; t)}{\partial t}, \quad f(x, y) = 0,$$
$$N(v) = -\nabla \cdot \left(\frac{\nabla v(x, y; t)}{\left|\nabla v(x, y; t)\right|}\right) - (\alpha - \beta)\frac{z(x, y)}{v(x, y; t)^2} - \beta\frac{z(x, y)^2}{v(x, y; t)^3}. \tag{62}$$

In order to handle the difficulty in applying the RHPM to the RLO model, we use three terms of binomial series (27) for v^{-2} and v^{-3} in (62). That is, for a small parameter $\varepsilon > 0$, v^{-2} and v^{-3} can be approximated by

$$\frac{1}{v^2} = \frac{\varepsilon^2}{(\varepsilon v)^2} \approx \varepsilon^2\left(6 - 8\varepsilon v + 3\varepsilon^2 v^2\right),$$
$$\frac{1}{v^3} = \frac{\varepsilon^3}{(\varepsilon v)^3} \approx \varepsilon^3\left(10 - 15\varepsilon v + 6\varepsilon^2 v^2\right). \tag{63}$$

Using (62) and (63), the RHPM(2) method for the RLO model can be derived in a similar way as was done for the AA2 model as follows:

$$\frac{\partial v_0}{\partial t} + p\frac{\partial v_1}{\partial t} + p^2\frac{\partial v_2}{\partial t} + \cdots = \frac{\partial u_0}{\partial t} - p\frac{\partial u_0}{\partial t} + p\left(\frac{v_{xx} + v_{yy}}{|\nabla v|} - \frac{(v_x)^2 v_{xx} + (v_x)^2 v_{yy}}{|\nabla v|^3}\right) + p(\alpha - \beta)\frac{z}{v^2} + p\beta\frac{z^2}{v^3} \approx \frac{\partial u_0}{\partial t} - p\frac{\partial u_0}{\partial t} + p\left[\{(A_1)(D_3) - (A_2)(D_5)\} + z\{(\alpha - \beta)(A_3) + \beta z(A_4)\}\right]$$

1: MAE(0) = 1, $(u_0^0)_{i,j} = z_{i,j}$ for all i and j
2: **for** $k = 0$ to *maxit* **do**
3: Set $v_0^k = u_0^k$
4: Compute v_1^k using (57) for all i and j
5: Compute v_2^k using (59) for all i and j
6: $u_0^{k+1} = v_0^k + v_1^k + v_2^k$
7: **if** MAE(k) < MAE($k + 1$) **then**
8: Stop
9: **end if**
10: **end for**

Algorithm 3: RHPM(2) algorithm for the AA2 model.

$$+ p^2 \left[\{(A_1)(D_4) - (A_2)(D_6)\} + \{(B_1)(D_3) - (B_2)(D_5)\}(D_2)\right] + p^2 z\{(\alpha-\beta)(B_3) + \beta z(B_4)\} + \cdots, \tag{64}$$

where $(A_1), (A_2), (B_1), (B_2), (D_2), (D_3), (D_4)$, and (D_5) are defined the same as the AA2 model and $(A_3), (A_4), (B_3)$, and (B_4) are defined as

$$\begin{aligned}(A_3) &= \varepsilon^2\left\{6 - 8\varepsilon(v_0) + 3\varepsilon^2(v_0)^2\right\},\\ (B_3) &= -\varepsilon^2\{8 - 6\varepsilon(v_0)\},\\ (A_4) &= \varepsilon^3\left\{10 - 15\varepsilon(v_0) + 6\varepsilon^2(v_0)^2\right\},\\ (B_4) &= -\varepsilon^3\{15 - 12\varepsilon(v_0)\}.\end{aligned} \tag{65}$$

Comparing coefficients of p^0, p^1, and p^2 in (64),

$$p^0 : \frac{\partial v_0}{\partial t} = \frac{\partial u_0}{\partial t} \Longrightarrow \quad v_0 = u_0 = z : \text{given an initial approximation} \tag{66}$$

$$p^1 : \frac{\partial v_1}{\partial t} = -\frac{\partial v_0}{\partial t} + \{(A_1)(D_3) - (A_2)(D_5)\} + z\{(\alpha-\beta)(A_3) + \beta z(A_4)\} \tag{67}$$

$$p^2 : \frac{\partial v_2}{\partial t} = [(A_1)(D_4) - (A_2)(D_6) + \{(B_1)(D_3) - (B_2)(D_5)\}(D_2)] + z\{(\alpha-\beta)(B_3) + \beta z(B_4)\}. \tag{68}$$

Applying the inverse operator of L to (67), one obtains

$$v_1 = (z - v_0) + [\{(A_1)(D_3) - (A_2)(D_5)\} + z\{(\alpha-\beta)(A_3) + \beta z(A_4)\}]t = (v_{10}) + (v_{11})t, \tag{69}$$

where (v_{10}) is the constant term and (v_{11}) is the coefficient of t.

Substituting (69) and (52) into (68) and applying the inverse operator of L to (68), one obtains

$$\begin{aligned}v_2 = {} & [\{(A_1)(D_{40}) - (A_2)(D_{60})\} + \{(B_1)(D_3) - (B_2)(D_5)\}(D_{20})]t\\ & + \frac{1}{2}[(A_1)(D_{41}) - (A_2)(D_{61}) + \{(B_1)(D_3) - (B_2)(D_5)\}(D_{21})]t^2\\ & + z\{(\alpha-\beta)(B_3) + \beta z(B_4)\}t + \frac{z}{2}\{(\alpha-\beta)(B_3) + \beta z(B_4)\}t^2 = (v_{21})t + (v_{22})t^2,\end{aligned} \tag{70}$$

where (v_{21}) and (v_{22}) are coefficients of t and t^2, respectively.

1: MAE(0) = 1, $(u_0^0)_{i,j} = z_{i,j}$ for all i and j
2: **for** $k = 0$ to *maxit* **do**
3: Set $v_0^k = u_0^k$
4: Compute v_1^k using (71) for all i and j
5: Compute v_2^k using (72) for all i and j
6: $u_0^{k+1} = v_0^k + v_1^k + v_2^k$
7: **if** MAE(k) < MAE($k+1$) **then**
8: Stop
9: **end if**
10: **end for**

Algorithm 4: RHPM(2) algorithm for the RLO model.

For numerical implementation of the RHPM(2) method, assume that $(u_0^0)_{i,j} = z(x_i, y_j) = z_{i,j}$ and $(u_0^k)_{i,j}$ has been computed for all i and j. Then $(v_0^k)_{i,j} = (u_0^k)_{i,j}$, and discretizations of (69) and (70) are given by

$$\left(v_1^k\right)_{i,j} = (v_{10})_{i,j} + (v_{11})_{i,j} \cdot t_k, \tag{71}$$

$$\left(v_2^k\right)_{i,j} = (v_{21})_{i,j} \cdot t_k + (v_{22})_{i,j} \cdot (t_k)^2, \tag{72}$$

where

$$\begin{aligned}(v_{10})_{i,j} &= z_{i,j} - \left(v_0^k\right)_{i,j}\\ (v_{11})_{i,j} &= \left\{(A_1)_{i,j}(D_3)_{i,j} - (A_2)_{i,j}(D_5)_{i,j}\right\} + z_{i,j}\left\{(\alpha-\beta)(A_3)_{i,j} + \beta z_{i,j}(A_4)_{i,j}\right\}\\ (v_{21})_{i,j} &= \left[\left\{(A_1)_{i,j}(D_{40})_{i,j} - (A_2)_{i,j}(D_{60})_{i,j}\right\} + \left\{(B_1)_{i,j}(D_3)_{i,j} - (B_2)_{i,j}(D_5)_{i,j}\right\}(D_{20})_{i,j}\right]\\ &\quad + z_{i,j}\left\{(\alpha-\beta)(B_3)_{i,j} + \beta z(B_4)_{i,j}\right\}\\ (v_{22})_{i,j} &= \frac{1}{2}\left[(A_1)_{i,j}(D_{41})_{i,j} - (A_2)_{i,j}(D_{61})_{i,j} + \left\{(B_1)_{i,j}(D_3)_{i,j} - (B_2)_{i,j}(D_5)_{ij}\right\}(D_{21})_{i,j}\right] + \frac{1}{2}\\ &\quad \cdot z_{i,j}\left\{(\alpha-\beta)(B_3)_{i,j} + \beta z(B_4)_{i,j}\right\}.\end{aligned} \tag{73}$$

We now compute $u_0^{k+1} = v_0^k + v_1^k + v_2^k$. Repeating the above process with u_0^{k+1} as an initial approximation, we can obtain the RHPM(2) method, called Algorithm 4, for multiplicative noise removal of the RLO model.

From RHPM(2) Algorithms 3 and 4, we can easily obtain RHPM(1) Algorithms 5 and 6 whose computational costs per iteration are much cheaper than those of Algorithms 3 and 4. In Algorithms 3–6, *maxit*, z, and u_0^k denote the maximum number of iterations, the noisy image, and the restored image at the kth iteration, respectively. Notice that all images are assumed to have an intensity range of $[0, 1]$ to guarantee numerical stability of the RHPM methods.

1: MAE(0) = 1, $(u_0^0)_{i,j} = z_{i,j}$ for all i and j
2: **for** $k = 0$ to $maxit$ **do**
3: Set $v_0^k = u_0^k$
4: Compute v_1^k using (57) for all i and j
5: $u_0^{k+1} = v_0^k + v_1^k$
6: **if** MAE(k) < MAE($k+1$) **then**
7: Stop
8: **end if**
9: **end for**

Algorithm 5: RHPM(1) algorithm for the AA2 model.

1: MAE(0) = 1, $(u_0^0)_{i,j} = z_{i,j}$ for all i and j
2: **for** $k = 0$ to $maxit$ **do**
3: Set $v_0^k = u_0^k$
4: Compute v_1^k using (71) for all i and j
5: $u_0^{k+1} = v_0^k + v_1^k$
6: **if** MAE(k) < MAE($k+1$) **then**
7: Stop
8: **end if**
9: **end for**

Algorithm 6: RHPM(1) algorithm for the RLO model.

5. Split Bregman Algorithms for Multiplicative Noise Removal

From (4) and (5), the general variational models based on TV regularization for multiplicative noise removal can be written as

$$\min_u \left\{ E(u) = \int_\Omega |\nabla u|\, dx + \int_\Omega H(u)\, dx \right\}. \tag{74}$$

To derive the alternating split Bregman algorithm for general multiplicative noise removal problem (74), we first introduce an auxiliary vector $\mathbf{w} = (w_x, w_y)^T$ so that u and $\mathbf{w}$ minimize the following unconstrained problem:

$$\min_{u,\mathbf{w}} \left\{ \int_\Omega |\mathbf{w}|\, dx + \frac{\theta}{2} \int_\Omega |\mathbf{w} - \nabla u|^2\, dx + \int_\Omega H(u)\, dx \right\}, \tag{75}$$

where $|\mathbf{w}| = \sqrt{(w_x)^2 + (w_y)^2}$ and $\theta > 0$ is a penalty parameter. Then unconstrained minimization problem (75) can be solved by the following split Bregman algorithm using an auxiliary vector $\mathbf{b} = (b_x, b_y)^T$:

$$\begin{aligned} &\left(u^{k+1}, \mathbf{w}^{k+1}\right) \\ &= \arg\min_{u,\mathbf{w}} \int_\Omega \left\{ |\mathbf{w}| + \frac{\theta}{2} \left|\mathbf{w} - \nabla u - \mathbf{b}^k\right|^2 + H(u) \right\} dx, \end{aligned} \tag{76}$$

$$\mathbf{b}^{k+1} = \mathbf{b}^k + \nabla u^{k+1} - \mathbf{w}^{k+1}, \tag{77}$$

where $\mathbf{b}^0 = \mathbf{w}^0 = \mathbf{0}$ and u^0 is set to z (noisy image).

Minimizing (76) alternatingly with respect to u and $\mathbf{w}$, one obtains the *alternating split Bregman algorithm* introduced in [25]:

$$u^{k+1} = \arg\min_u \int_\Omega \left\{ H(u) + \frac{\theta}{2} \left|\mathbf{w}^k - \nabla u - \mathbf{b}^k\right|^2 \right\} dx, \tag{78}$$

$$\mathbf{w}^{k+1} = \arg\min_{\mathbf{w}} \int_\Omega \left\{ |\mathbf{w}| + \frac{\theta}{2} \left|\mathbf{w} - \nabla u^{k+1} - \mathbf{b}^k\right|^2 \right\} dx. \tag{79}$$

The Euler-Lagrange equations corresponding to (78) and (79) are as follows:

$$\frac{\partial}{\partial u} H\left(u^{k+1}\right) - \theta \nabla \cdot \nabla u^{k+1} + \theta \nabla \cdot \left(\mathbf{w}^k - \mathbf{b}^k\right) = 0 \tag{80}$$

$$\frac{1}{\theta} \cdot \frac{\mathbf{w}^{k+1}}{\left|\mathbf{w}^{k+1}\right|} + \left(\mathbf{w}^{k+1} - \nabla u^{k+1} - \mathbf{b}^k\right) = 0, \tag{81}$$

where $\mathbf{w}^k = (w_x^k, w_y^k)^T$, $\mathbf{b}^k = (b_x^k, b_y^k)^T$, and $|\mathbf{w}^{k+1}| = \sqrt{(w_x^{k+1})^2 + (w_y^{k+1})^2}$.

First we consider how to solve (80) for u^{k+1}. For the case of RLO model (4), $H(u) = \alpha(z/u) + (\beta/2)(z/u - 1)^2$ and thus

$$\frac{\partial}{\partial u} H(u) = (\beta - \alpha) \frac{z}{u^2} - \beta \frac{z^2}{u^3}. \tag{82}$$

To avoid division by zero or near-zero for z/u^2 and z^2/u^3, u^{-2} and u^{-3} are approximated using four terms of binomial series (27). That is,

$$\begin{aligned} \frac{1}{u^2} &\approx 10 - 20u + 15u^2 - 4u^3, \\ \frac{1}{u^3} &\approx 20 - 45u + 36u^2 - 10u^3, \end{aligned} \tag{83}$$

where u need to be scaled so that the value of u lies between 0 and 2 to guarantee convergence of the binomial series. Thus $(\partial/\partial u)H(u)$ can be approximated as

$$\begin{aligned} \frac{\partial}{\partial u} H(u) \approx{}& (\beta - \alpha)\left(10 - 20u + 15u^2 - 4u^3\right) z \\ &- \beta\left(20 - 45u + 36u^2 - 10u^3\right) z^2. \end{aligned} \tag{84}$$

From this approximate equation, $(\partial/\partial u)H(u^{k+1})$ in (80) can be approximated as

$$\begin{aligned} \frac{\partial}{\partial u} H\left(u^{k+1}\right) \approx{}& -10(\alpha - \beta) z - 20\beta z^2 \\ &+ \left\{20(\alpha - \beta) z + 45\beta z^2\right\} u^{k+1} \\ &- \left\{15(\alpha - \beta) z + 36\beta z^2\right\} \left(u^k\right)^2 \\ &+ \left\{4(\alpha - \beta) z + 10\beta z^2\right\} \left(u^k\right)^3. \end{aligned} \tag{85}$$

Substituting this approximate equation into (80), one obtains the following approximate PDE for the RLO model:

$$\begin{aligned}&\left\{20\left(\alpha-\beta\right)z+45\beta z^2\right\}u^{k+1}-\theta\nabla\cdot\nabla u^{k+1}\\&\quad=\left\{10\left(\alpha-\beta\right)z+20\beta z^2\right\}\\&\qquad+\left\{15\left(\alpha-\beta\right)z+36\beta z^2\right\}\left(u^k\right)^2\\&\qquad-\left\{4\left(\alpha-\beta\right)z+10\beta z^2\right\}\left(u^k\right)^3-\theta\nabla\\&\qquad\cdot\left(\mathbf{w}^k-\mathbf{b}^k\right).\end{aligned}\tag{86}$$

For the case of AA2 model (5), $H(u)=\alpha(u+ze^{-u})$. Using four terms of the Taylor series for e^{-u}, $(\partial/\partial u)H(u)$ can be approximated as

$$\begin{aligned}\frac{\partial}{\partial u}H\left(u\right)&=\alpha\left(1-ze^{-u}\right)\\&\approx\alpha\left\{1-z\left(1-u+\frac{1}{2}u^2-\frac{1}{6}u^3\right)\right\}.\end{aligned}\tag{87}$$

From this approximate equation, $(\partial/\partial u)H(u^{k+1})$ in (80) can be approximated as

$$\begin{aligned}&\frac{\partial}{\partial u}H\left(u^{k+1}\right)\\&\quad\approx\alpha\left\{1-z+zu^{k+1}-\frac{1}{2}z\left(u^k\right)^2+\frac{1}{6}z\left(u^k\right)^3\right\}.\end{aligned}\tag{88}$$

Substituting this approximate equation into (80), we can obtain the following approximate PDE for the AA2 model:

$$\begin{aligned}&\alpha zu^{k+1}-\theta\nabla\cdot\nabla u^{k+1}\\&\quad=-\alpha\left\{1-z-\frac{1}{2}z\left(u^k\right)^2+\frac{1}{6}z\left(u^k\right)^3\right\}-\theta\nabla\\&\qquad\cdot\left(\mathbf{w}^k-\mathbf{b}^k\right).\end{aligned}\tag{89}$$

Now we consider the numerical approach for solving PDE problems of RLO model (86) and AA2 model (89). Let $u_{i,j}$, $z_{i,j}$, $((b_x)_{i,j},(b_y)_{i,j})^T$, and $((w_x)_{i,j},(w_y)_{i,j})^T$ be the (i,j)−components of u, z, **b,** and **w,** respectively, where $1\le i\le m$ and $1\le j\le n$. For an $m\times n$ array v, let vec(v) denote a long vector of size mn which is defined by vec$(v)=(v_{*1}^T,v_{*2}^T,\ldots,v_{*n}^T)^T$ and $v_{*i}=(v_{1,i},v_{2,i},\ldots,v_{m,i})^T$. Let $\tilde{\nabla}=(\tilde{\nabla}_x,\tilde{\nabla}_y)^T:\mathbb{R}^{m\times n}\longrightarrow\mathbb{R}^{2m\times n}$ be a discrete gradient operator defined by

$$\begin{aligned}\left(\tilde{\nabla}u\right)_{i,j}&=\left(\left(\tilde{\nabla}_x u\right)_{i,j},\left(\tilde{\nabla}_y u\right)_{i,j}\right)^T,\\\left(\tilde{\nabla}_x u\right)_{i,j}&=\begin{cases}0, & i=1\\u_{i,j}-u_{i-1,j}, & 1<i\le m,\end{cases}\\\left(\tilde{\nabla}_y u\right)_{i,j}&=\begin{cases}0, & j=1\\u_{i,j}-u_{i,j-1}, & 1<j\le n.\end{cases}\end{aligned}\tag{90}$$

Then the discrete gradient operator $\tilde{\nabla}$ can be represented by a matrix operator B of size $2mn\times mn$:

$$B=\begin{pmatrix}I_n\otimes D_m\\D_n\otimes I_m\end{pmatrix},\tag{91}$$

where $\otimes$ denotes the Kronecker product, I_m denotes the identity matrices of order m, and D_m denotes the following matrix of order m:

$$D_m=\begin{pmatrix}0&0&\cdots&\cdots&0\\-1&1&\cdots&\cdots&0\\\vdots&\ddots&\ddots&\ddots&\vdots\\0&\cdots&-1&1&0\\0&\cdots&0&-1&1\end{pmatrix}_{m\times m}.\tag{92}$$

Using the matrix operator B, it is easy to show that $\nabla\cdot\nabla u$ and $\nabla\cdot\mathbf{w}$ in (86) and (89) can be approximated as

$$\begin{aligned}\nabla\cdot\nabla u&\approx-B^TB\,\mathrm{vec}\left(u\right),\\\nabla\cdot\mathbf{w}&\approx B^T\begin{pmatrix}\mathrm{vec}\left(w_x\right)\\\mathrm{vec}\left(w_y\right)\end{pmatrix}.\end{aligned}\tag{93}$$

Hence, we obtain the finite difference approximate equations for RLO model (86) and AA2 model (89) which are given by the following linear systems, respectively:

$$\begin{aligned}&\left[\mathrm{diag}\left(20\left(\alpha-\beta\right)z+45\beta z^2\right)+\theta B^TB\right]\mathrm{vec}\left(u^{k+1}\right)\\&\quad=\mathrm{vec}\left(\left\{10\left(\alpha-\beta\right)z+20\beta^2\right\}\right.\\&\qquad+\left\{15\left(\alpha-\beta\right)z+36\beta z^2\right\}\left(u^k\right)^2\\&\qquad\left.-\left\{4\left(\alpha-\beta\right)z+10\beta z^2\right\}\left(u^k\right)^3\right)\\&\qquad-\theta B^T\begin{pmatrix}\mathrm{vec}\left(w_x^k-b_x^k\right)\\\mathrm{vec}\left(w_y^k-b_y^k\right)\end{pmatrix},\end{aligned}\tag{94}$$

$$\begin{aligned}&\left[\mathrm{diag}\left(\alpha z\right)+\theta B^TB\right]\mathrm{vec}\left(u^{k+1}\right)\\&\quad=\mathrm{vec}\left(-\alpha\left\{1-z-\frac{1}{2}z\left(u^k\right)^2+\frac{1}{6}z\left(u^k\right)^3\right\}\right)\\&\qquad-\theta B^T\begin{pmatrix}\mathrm{vec}\left(w_x^k-b_x^k\right)\\\mathrm{vec}\left(w_y^k-b_y^k\right)\end{pmatrix},\end{aligned}\tag{95}$$

where z^2, $(u^k)^2$, and $(u^k)^3$ denote elementwise exponentiations, u^k denotes the restored image at the kth iteration, all multiplications of two $m\times n$ arrays denote elementwise multiplications, and diag(z) is an $mn\times mn$ diagonal matrix whose diagonal elements are the same as vec(z).

Next, we consider how to solve (81) for $\mathbf{w}^{k+1}$. Equation (81) can be expressed as

$$\begin{aligned}\left(1+\frac{1}{\theta}\cdot\frac{1}{\left|\mathbf{w}^{k+1}\right|}\right)w_x^{k+1}&=\nabla_x u^{k+1}+b_x^k\\\left(1+\frac{1}{\theta}\cdot\frac{1}{\left|\mathbf{w}^{k+1}\right|}\right)w_y^{k+1}&=\nabla_y u^{k+1}+b_y^k,\end{aligned}\tag{96}$$

where $\nabla u^{k+1} = (\nabla_x u^{k+1}, \nabla_y u^{k+1})^T$. By squaring and adding two equations in (96), one can obtain

$$\left(1 + \frac{1}{\theta} \cdot \frac{1}{\left|\mathbf{w}^{k+1}\right|}\right)^2 \left|\mathbf{w}^{k+1}\right|^2 = \left|\nabla u^{k+1} + \mathbf{b}^k\right|^2, \quad (97)$$

which implies

$$\left|\mathbf{w}^{k+1}\right| = \left|\nabla u^{k+1} + \mathbf{b}^k\right| - \frac{1}{\theta}. \quad (98)$$

Substituting (98) into (96), we have a generalized shrinkage formula

$$\begin{aligned} w_x^{k+1} &= \left(\left|\nabla u^{k+1} + \mathbf{b}^k\right| - \frac{1}{\theta}\right) \frac{\nabla_x u^{k+1} + b_x^k}{\left|\nabla u^{k+1} + \mathbf{b}^k\right|} \\ w_y^{k+1} &= \left(\left|\nabla u^{k+1} + \mathbf{b}^k\right| - \frac{1}{\theta}\right) \frac{\nabla_y u^{k+1} + b_y^k}{\left|\nabla u^{k+1} + \mathbf{b}^k\right|}. \end{aligned} \quad (99)$$

Generalized shrinkage formula (99) can be also found in [25, 29]. To make this paper self-contained, we provided a detailed proof for (99). Numerical implementation for formula (99) can be done as follows: for each (i, j)-component of $\mathbf{w}^{k+1} = (w_x^{k+1}, w_y^{k+1})^T$,

$$\begin{aligned} \left(w_x^{k+1}\right)_{i,j} &= \max\left\{\left|\left(\tilde{\nabla} u^{k+1} + \mathbf{b}^k\right)_{i,j}\right| - \frac{1}{\theta}, 0\right\} \\ &\cdot \frac{\left(\tilde{\nabla}_x u^{k+1}\right)_{i,j} + \left(b_x^k\right)_{i,j}}{\left|\left(\tilde{\nabla} u^{k+1} + \mathbf{b}^k\right)_{i,j}\right|}, \quad 0\frac{0}{0} = 0 \\ \left(w_y^{k+1}\right)_{i,j} &= \max\left\{\left|\left(\tilde{\nabla} u^{k+1} + \mathbf{b}^k\right)_{i,j}\right| - \frac{1}{\theta}, 0\right\} \\ &\cdot \frac{\left(\tilde{\nabla}_y u^{k+1}\right) i, j + \left(b_y^k\right)_{i,j}}{\left|\left(\tilde{\nabla} u^{k+1} + \mathbf{b}^k\right)_{i,j}\right|}, \quad 0\frac{0}{0} = 0. \end{aligned} \quad (100)$$

where $(\tilde{\nabla} u^{k+1} + \mathbf{b}^k)_{i,j} = ((\tilde{\nabla}_x u^{k+1})_{i,j} + (b_x^k)_{i,j}, (\tilde{\nabla}_y u^{k+1})_{i,j} + (b_y^k)_{i,j})^T$.

Lastly, numerical implementation for (77) can be done as follows:

$$\begin{pmatrix} \operatorname{vec}\left(b_x^{k+1}\right) \\ \operatorname{vec}\left(b_y^{k+1}\right) \end{pmatrix} = \begin{pmatrix} \operatorname{vec}\left(b_x^k\right) \\ \operatorname{vec}\left(b_y^k\right) \end{pmatrix} + B \cdot \operatorname{vec}\left(u^{k+1}\right) - \begin{pmatrix} \operatorname{vec}\left(w_x^{k+1}\right) \\ \operatorname{vec}\left(w_y^{k+1}\right) \end{pmatrix}. \quad (101)$$

Linear systems (94) and (95) are singular or ill-conditioned problems, so direct or iterative methods fail to get an approximate solution for these problems. For this reason, we propose a new technique for solving linear systems (94) and (95). Let

$$\begin{aligned} A_1 &= \operatorname{diag}\left(20(\alpha - \beta) z + 45\beta z^2\right) + \theta B^T B, \\ A_2 &= \operatorname{diag}(\alpha z) + \theta B^T B \\ r_1^k &= \text{right-hand side vector of (94)}, \\ r_2^k &= \text{right-hand side vector of (95)} \\ u_{k+1} &= \operatorname{vec}\left(u^{k+1}\right), \\ u_k &= \operatorname{vec}\left(u^k\right) \end{aligned} \quad (102)$$

D = diagonal matrix whose diagonal elements are the same as those of $\theta B^T B$.

Then linear systems (94) and (95) can be rewritten as

$$\begin{aligned} A_1 u_{k+1} &= r_1^k, \\ A_2 u_{k+1} &= r_2^k. \end{aligned} \quad (103)$$

In order to solve ill-conditioned linear systems (103) for u_{k+1}, we propose the following iterative method:

Algorithm: SOLVER(i)
Choose $y^1 = u_k$
for $\ell = 1$ to $maxl$
Solve $(A_i + \delta D) y^{\ell+1} = \delta D y^\ell + r_i^k$ for $y^{\ell+1}$
end for
$u_{k+1} = y^{\ell+1}$

In it $i = 1$ or 2, u_k refers to the restored image computed at the previous kth step, and $\delta > 0$ is a parameter which should be chosen so that the coefficient matrix $A_i + \delta D$ is well conditioned. To study semiconvergence of SOLVER(i), we first introduce some important properties for an iterative method.

Theorem 1 (see [30, 31]). *Let $S \in \mathbb{R}^{N \times N}$ be a symmetric positive semidefinite matrix and $b \in \mathbb{R}^N$. Let $S = M - N$ with M nonsingular and let $G = M^{-1}N$. If $M^T + N$ is symmetric*

positive definite, then G is semiconvergent, and an iterative method $y^{\ell+1} = Gy^{\ell} + M^{-1}b$, or equivalently $My^{\ell+1} = Ny^{\ell} + b$, is semiconvergent. In other words, y^{ℓ} generated by the iterative method converges to some solution to $Sy = b$ for each y^1.

Lemma 2. *Let $\alpha \geq \beta \geq 0$ and $\delta > 0$. Then A_i $(i = 1, 2)$ are symmetric positive semidefinite, and D and $A_i + \delta D$ $(i = 1, 2)$ are symmetric positive definite.*

Proof. It can be easily shown that all diagonal elements of $B^T B$ are positive. Hence all properties of this lemma clearly hold from the assumptions. □

Theorem 3. *Let $\alpha \geq \beta \geq 0$ and $\delta > 0$. Then y^{ℓ} generated by SOLVER(i) converges to a solution u_{k+1} to $A_i\, u = r_i^k$.*

Proof. Note that $A_i = (A_i + \delta D) - \delta D$. Let $M = A_i + \delta D$ and $N = \delta D$. From Lemma 2, A_i is symmetric positive semidefinite, and M and N are symmetric positive definite. Hence, M is nonsingular and $M^T + N$ is symmetric positive definite. From Theorem 1, the iterative method $My^{\ell+1} = Ny^{\ell} + r_i^k$, which is equivalent to $(A_i + \delta D)y^{\ell+1} = \delta D y^{\ell} + r_i^k$, is semiconvergent. Hence the proof is complete. □

Since $A_i + \delta D$ is symmetric positive definite for $\alpha \geq \beta \geq 0$ and $\delta > 0$, the linear systems in SOLVER(i) can be solved using the CG (Conjugate Gradient) method [32, 33]. Notice that convergence rate of the CG method depends on the condition number of coefficient matrix $A_i + \delta D$. So we next provide some conditional properties for $A_i + \delta D$ in SOLVER(i). For a square matrix G, let $\kappa(G)$ denote the condition number of G. For a symmetric matrix $S \in \mathbb{R}^{N\times N}$, let $\lambda_k(S)$ denote the kth largest eigenvalue of the matrix S for $1 \leq k \leq N$.

Lemma 4 (see [34]). *Let $S \in \mathbb{R}^{N\times N}$ and $S + E \in \mathbb{R}^{N\times N}$ be symmetric matrices. Then for each $1 \leq k \leq N$*

$$\lambda_k(S) + \lambda_N(E) \leq \lambda_k(S + E) \leq \lambda_k(S) + \lambda_1(E). \tag{104}$$

Lemma 5. *Let $\alpha \geq \beta \geq 0$, $\delta > 0$, and $N = mn$. Then for each $i = 1, 2$*

$$\begin{aligned} \frac{\lambda_1(A_i) + \delta\lambda_N(D)}{\lambda_N(A_i) + \delta\lambda_1(D)} &\leq \kappa(A_i + \delta D) \\ &\leq \frac{\lambda_1(A_i) + \delta\lambda_1(D)}{\lambda_N(A_i) + \delta\lambda_N(D)}. \end{aligned} \tag{105}$$

Proof. Since A_i and $A_i + \delta D$ are symmetric, from Lemma 4 one obtains

$$\begin{aligned} \lambda_1(A_i) + \delta\lambda_N(D) &\leq \lambda_1(A_i + \delta D) \\ &\leq \lambda_1(A_i) + \delta\lambda_1(D) \\ \lambda_N(A_i) + \delta\lambda_N(D) &\leq \lambda_N(A_i + \delta D) \\ &\leq \lambda_N(A_i) + \delta\lambda_1(D). \end{aligned} \tag{106}$$

From Lemma 2, $A_i + \delta D$ is symmetric positive definite and thus

$$\kappa(A_i + \delta D) = \frac{\lambda_1(A_i + \delta D)}{\lambda_N(A_i + \delta D)}. \tag{107}$$

From (106) and (107), this lemma follows. □

Lemma 6. *Let $\alpha \geq \beta \geq 0$, $\delta > 0$, and $N = mn$. Then for each $i = 1, 2$*

$$\frac{\lambda_1(A_i) + 2\delta}{\lambda_N(A_i) + 4\delta} \leq \kappa(A_i + \delta D) \leq \frac{\lambda_1(A_i) + 4\delta}{\lambda_N(A_i) + 2\delta}. \tag{108}$$

Proof. Note that all diagonal elements of the diagonal matrix D belong to a set $\{2, 3, 4\}$. Hence $\lambda_1(D) = 4$ and $\lambda_N(D) = 2$. From Lemma 5, this lemma follows. □

Theorem 7. *Let $\alpha \geq \beta \geq 0$, $\delta > 0$, and $N = mn$. If $2\lambda_N(A_i) < \lambda_1(A_i)$, then, for a given $\epsilon > 0$, there exists $\delta > 0$ such that $1/2 < \kappa(A_i + \delta D) < 2 + \epsilon$.*

Proof. For $\delta > 0$, let

$$\begin{aligned} f(\delta) &= \frac{\lambda_1(A_i) + 4\delta}{\lambda_N(A_i) + 2\delta}, \\ g(\delta) &= \frac{\lambda_1(A_i) + 2\delta}{\lambda_N(A_i) + 4\delta}. \end{aligned} \tag{109}$$

From Lemma 6, $g(\delta) < \kappa(A_i + \delta D) < f(\delta)$. By simple calculation, one has

$$\begin{aligned} f'(\delta) &= \frac{4\lambda_N(A_i) - 2\lambda_1(A_i)}{(\lambda_N(A_i) + 2\delta)^2}, \\ g'(\delta) &= \frac{2\lambda_N(A_i) - 4\lambda_1(A_i)}{(\lambda_N(A_i) + 4\delta)^2}. \end{aligned} \tag{110}$$

Since $2\lambda_N(A_i) < \lambda_1(A_i)$, from (110) $f'(\delta) < 0$ and $g'(\delta) < 0$. Also note that $\lim_{\delta\to\infty} f(\delta) = 2$ and $\lim_{\delta\to\infty} g(\delta) = 1/2$. Thus f is a decreasing function which converges to 2, and g is a decreasing function which converges to $1/2$. It follows that $g(\delta) > 1/2$ for any $\delta > 0$, and for a given $\epsilon > 0$ there exists a large $\delta > 0$ such that $2 < f(\delta) < 2 + \epsilon$. Since $g(\delta) < \kappa(A_i + \delta D) < f(\delta)$, for a given $\epsilon > 0$ there exists a large $\delta > 0$ such that $1/2 < \kappa(A_i + \delta D) < 2 + \epsilon$. Hence the proof is complete. □

Notice that the condition $2\lambda_N(A_i) < \lambda_1(A_i)$ in Theorem 7 usually holds when A_i is a singular or ill-conditioned matrix. Theorem 7 implies that the condition number of $A_i + \delta D$ is small if a large number of $\delta > 0$ is chosen. In other words, for a suitably chosen large $\delta > 0$ the coefficient matrix $A_i + \delta D$ in SOLVER(i) is well conditioned, so that the CG method for solving the linear systems in SOLVER(i) converges fast to the exact solution. The split Bregman method is an iterative method which monotonically decreases the residual norm between the original image and noisy image, so we do not have to get an accurate solution every iteration. For this reason, we have used $maxl = 1$

1: $\mathbf{b}^0 = \mathbf{w}^0 = 0$ and $u^0 = z$
2: **for** $k = 0$ to $maxit$ **do**
3: Compute u^{k+1} using SOLVER(1)
4: Compute $\mathbf{w}^{k+1}$ using Equation (100)
5: Compute $\mathbf{b}^{k+1}$ using Equation (101)
6: **if** $\dfrac{\|u^k - u^{k+1}\|_2}{\|u^{k+1}\|_2} < tol$ **then**
7: Stop
8: **end if**
9: **end for**

Algorithm 7: Split Bregman algorithm for the RLO model.

1: $\mathbf{b}^0 = \mathbf{w}^0 = 0$ and $u^0 = z$
2: **for** $k = 0$ to $maxit$ **do**
3: Compute u^{k+1} using SOLVER(2)
4: Compute $\mathbf{w}^{k+1}$ using Equation (100)
5: Compute $\mathbf{b}^{k+1}$ using Equation (101)
6: **if** $\dfrac{\|u^k - u^{k+1}\|_2}{\|u^{k+1}\|_2} < tol$ **then**
7: Stop
8: **end if**
9: **end for**

Algorithm 8: Split Bregman algorithm for the AA2 model.

and the CG method with tolerance set to 1×10^{-3} to get an approximate solution of the linear system in SOLVER(i). Notice that the CG method always achieves the desired tolerance within 15 iterations.

Finally, one can obtain the split Bregman algorithms for the RLO and AA2 models. In Algorithms 7 and 8, *maxit* denotes the maximum number of outer iterations, $\|\cdot\|_2$ stands for the l_2-norm, and *tol* denotes the tolerance of the stopping criterion which is set to 3×10^{-3} for the test problems used in this paper. Notice that all images are assumed to have an intensity range of $[0, 1]$ to guarantee convergence of the binomial series used in the split Bregman algorithm.

6. Numerical Experiments

In this section, we provide numerical performance results of the TM, split Bregman, RHPM(2), and RHPM(1) methods for multiplicative noise removal problems. Numerical tests are performed using MATLAB R2016a on a personal computer with 3.6GHz CPU and 8GB RAM. The multiplicative noisy images are generated by multiplying Gaussian white noise or Gamma noise to the clean image using the built-in MATLAB function *randn* or *randg*. More specifically, the multiplicative noisy images z are generated by

$$\begin{aligned} \eta &= 1 + \sqrt{0.01} \times \text{randn}\,(m,n)\,; \quad z = u \cdot \eta, \\ \eta &= 0.01 \times \text{randg}\left(\frac{1}{0.01}, (m,n)\right); \quad z = u \cdot \eta, \end{aligned} \tag{111}$$

where u is the clean image, $\eta = 1 + \sqrt{0.01} \times randn(m,n)$ generates the Gaussian white noise with mean 1 and variance 0.01, $\eta = 0.01 \times randg(1/0.01, (m,n))$ generates the Gamma noise with mean 1 and variance 0.01, and $u \cdot \eta$ is componentwise multiplication of u and η.

In order to illustrate the efficiency and reliability of the proposed denoising algorithms (i.e., Algorithms 1–8), we provide numerical results for 4 test images such as the Caribou, Boat, Jet Plane, and Lake. The pixel size of the Caribou image is 256×256, and the pixel size of the other 3 test images is 512×512. To evaluate the quality of the restored images, we use the peak signal-to-noise ratio (PSNR) between the restored image and original image which is defined by

$$\text{PSNR} = 10 \log_{10}\left(\frac{m \cdot n \cdot \max_{i,j} \left|u_{i,j}\right|^2}{\|u - \widetilde{u}\|_F^2}\right), \tag{112}$$

where $\|\cdot\|_F$ refers to the Frobenius norm and u and $\widetilde{u}$ are the original and restored images, respectively. Also $u_{i,j}$ stands for the value of the original image u at the pixel point (i, j) and $m \cdot n$ is the total number of pixels. It is generally true that the larger PSNR stands for the better quality of the denoised image.

For numerical experiments, the TM method uses the test images with an intensity range of $[0, 255]$, the split Bregman and RHPM methods use the test images with an intensity range of $[0, 1]$, and the noise η is Gaussian white noise or Gamma noise with mean 1 and variance 0.01 or 0.03. For all test problems, an initial image was set to the observed noisy image z, and we have used $maxit = 200$, $\alpha = 0.2$, and $\beta = 0.1$ for the RLO model and $\alpha = 10^{-4}$ for the AA2 model. For the TM and RHPM methods (i.e., Algorithms 1–6), we have used $\triangle t = 0.35$, and the iteration was terminated when MAE(k), defined in Section 2, started to increase. For the split Bregman method (i.e., Algorithms 7 and 8), we have used $\theta = 255$, and the iteration was terminated when the following stopping criterion was satisfied:

$$\frac{\left\|u^k - u^{k+1}\right\|_2}{\left\|u^{k+1}\right\|_2} < tol \tag{113}$$

where $tol = 3 \times 10^{-3}$ for all test problems.

Tables 1 and 2 provide numerical results for the TM, split Bregman, RHPM(1), and RHPM(2) methods corresponding to the AA2 model with multiplicative Gamma noise and the RLO model with multiplicative Gaussian white noise, respectively. In Tables 1 and 2, P_0 represents the PSNR values for the noisy images, *Psnr* represents the PSNR values for the restored images, parameters ϵ and δ are chosen as the best one by numerical tries, *Iter* denotes the number of iterations, and *Time* denotes the elapsed CPU time in seconds.

Figures 1–4 show the denoised images by the TM, split Bregman, RHPM(1), and RHPM(2) methods for the AA2 model with multiplicative Gamma noise of variance 0.03. Figures 5–8 show the denoised images by the TM, split Bregman, RHPM(1), and RHPM(2) methods for the RLO model with multiplicative Gaussian white noise of variance 0.03. The first-row images are the original image, noisy image,

Table 1: Numerical results of multiplicative noise removal for the AA2 model.

Method	Image	Variance= 0.01				Variance= 0.03			
		Caribou	Boat	Jet Plane	Lake	Caribou	Boat	Jet Plane	Lake
	P_0	24.89	25.34	22.12	24.65	20.12	20.58	17.34	19.90
TM	ε	0.012	0.011	0.011	0.033	0.05	0.049	0.072	0.2
	Iter	41	39	53	48	63	67	99	91
	Psnr	28.25	29.33	27.77	27.25	25.57	26.12	23.95	23.60
	Time	1.27	4.81	6.55	6.08	1.75	7.32	10.71	11.75
Split Bregman	δ	24	30	16	28	17	21	12	22
	Iter	16	16	14	16	20	21	18	22
	Psnr	30.18	29.82	28.94	29.38	27.82	27.48	26.50	27.03
	Time	0.45	1.91	1.65	1.82	0.55	2.52	2.05	2.51
RHPM(1)	ε	0.00062	0.0008	0.0008	0.00082	0.00014	0.00014	0.00014	0.00014
	Iter	23	21	21	21	53	53	53	53
	Psnr	29.67	29.58	27.82	29.06	26.18	26.34	23.76	25.76
	Time	0.03	1.08	0.93	1.01	1.47	2.81	2.77	2.73
RHPM(2)	ε	0.052	0.039	0.067	0.044	0.071	0.07	0.078	0.073
	Iter	3	3	3	3	3	3	3	3
	Psnr	30.38	29.92	29.05	29.49	27.53	27.36	25.58	26.94
	Time	0.12	0.49	0.49	0.49	0.14	0.5	0.5	0.5

Table 2: Numerical results of multiplicative noise removal for the RLO model.

Method	Image	Variance= 0.01				Variance= 0.03			
		Caribou	Boat	Jet Plane	Lake	Caribou	Boat	Jet Plane	Lake
	P_0	24.83	25.35	22.10	24.67	20.05	20.58	17.33	19.90
TM	ε	0.0004	0.00021	0.00019	0.00019	0.0004	0.00042	0.0004	0.038
	Iter	40	39	53	47	63	66	97	84
	Psnr	28.36	29.38	27.76	27.33	25.60	26.24	24.10	23.97
	Time	1.57	6.50	8.69	8.00	2.70	11.24	16.53	14.33
Split Bregman	δ	24	30	16	28	18	22	12	22
	Iter	16	16	14	16	21	22	18	22
	Psnr	30.20	29.84	28.92	29.39	27.84	27.53	26.52	27.03
	Time	0.58	2.43	2.41	2.39	0.59	2.40	2.06	2.47
RHPM(1)	ε	0.0005	0.00045	0.00045	0.00045	0.00014	0.00014	0.00014	0.00014
	Iter	27	29	28	29	53	53	53	53
	Psnr	29.71	29.60	27.75	29.06	26.14	26.38	23.78	25.76
	Time	0.30	1.38	1.38	1.39	0.64	2.48	2.24	2.45
RHPM(2)	ε	0.053	0.039	0.067	0.044	0.071	0.07	0.077	0.073
	Iter	3	3	3	3	3	3	3	3
	Psnr	30.40	29.95	29.03	29.50	27.51	27.40	25.59	26.93
	Time	0.14	0.41	0.42	0.40	0.59	0.36	0.46	0.42

and the denoised image by the TM method. The second-row images are the images denoised by the split Bregman, RHPM(1), and RHPM(2) methods.

As can be seen in Tables 1 and 2, the split Bregman method denoises best for variance 0.03, while RHPM(2) denoises best for variance 0.01. That is, the split Bregman method has the highest PSNR values for variance 0.03, while RHPM(2) has the highest PSNR values for variance 0.01. Observe that RHPM(1) computes 2 vectors v_0, v_1 every iteration, while RHPM(2) computes 3 vectors v_0, v_1, v_2 every iteration. It means that the computational cost of RHPM(2) per iteration is more expensive than RHPM(1). However, the convergence rate of RHPM(2) is much faster than RHPM(1), so that total CPU time of RHPM(2) is much less than RHPM(1). In addition, RHPM(2) denoises much better than RHPM(1), and RHPM(2) has the smallest CPU time of all

(a) True image (b) Noisy image (c) TM denoising

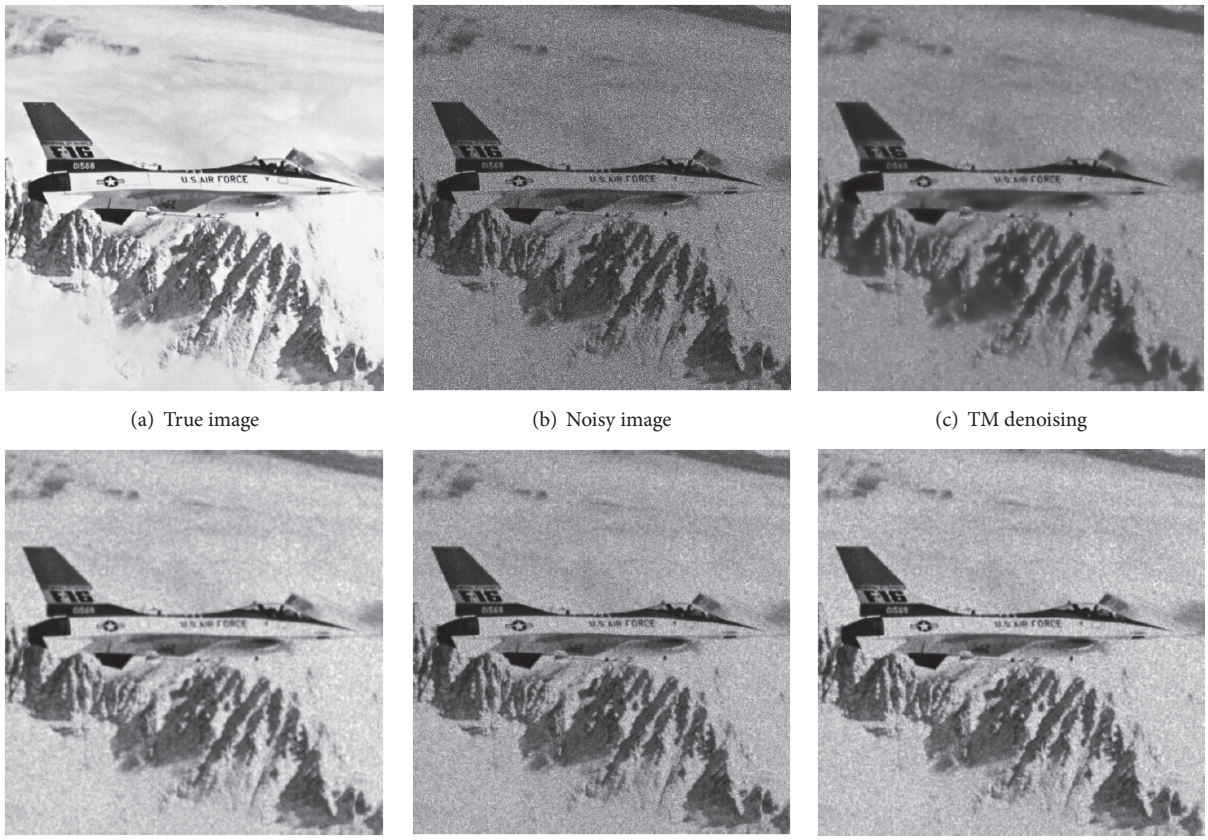
(d) Split Bregman denoising (e) RHPM(1) denoising (f) RHPM(2) denoising

Figure 1: Denoising of multiplicative Gamma noise for the Caribou image with a 256×256 size.

(a) True image (b) Noisy image (c) TM denoising

(d) Split Bregman denoising (e) RHPM(1) denoising (f) RHPM(2) denoising

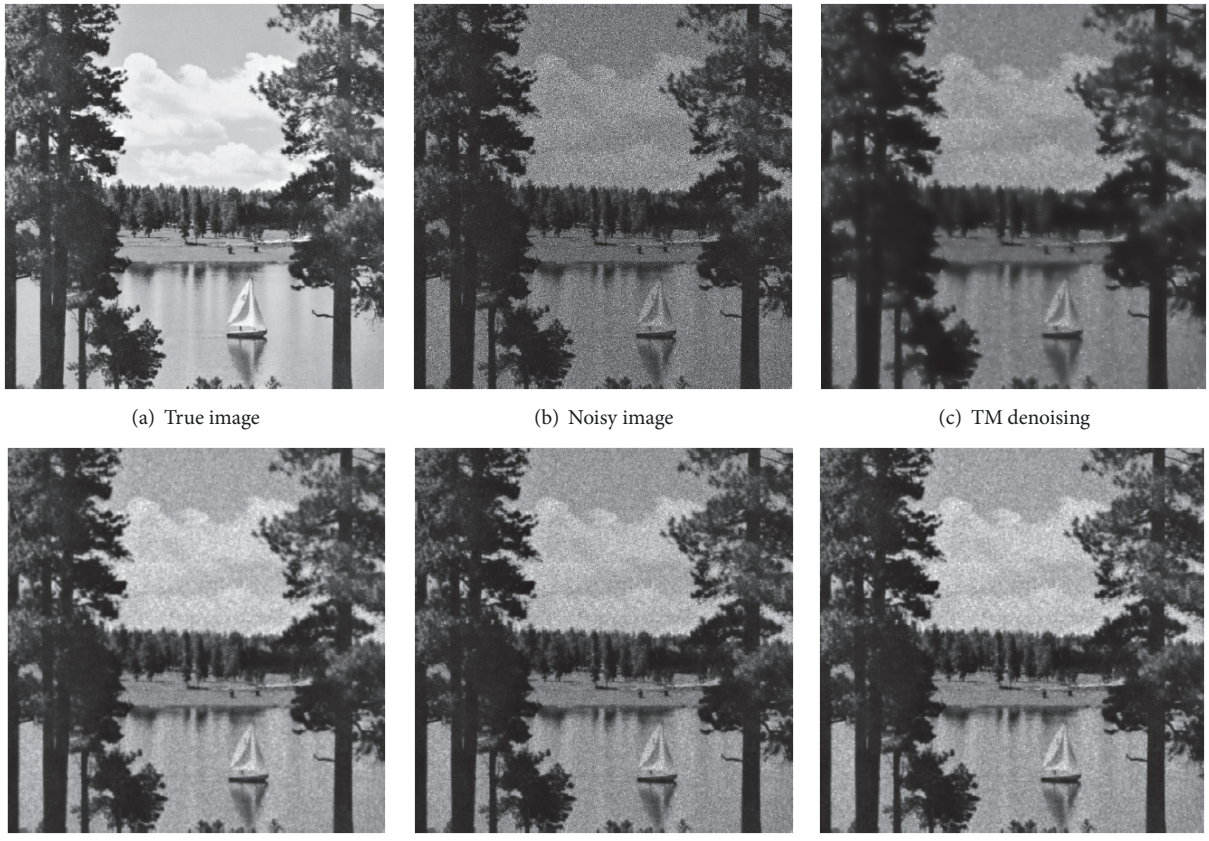
Figure 2: Denoising of multiplicative Gamma noise for the Boat image with a 512×512 size.



(a) True image (b) Noisy image (c) TM denoising

(d) Split Bregman denoising (e) RHPM(1) denoising (f) RHPM(2) denoising

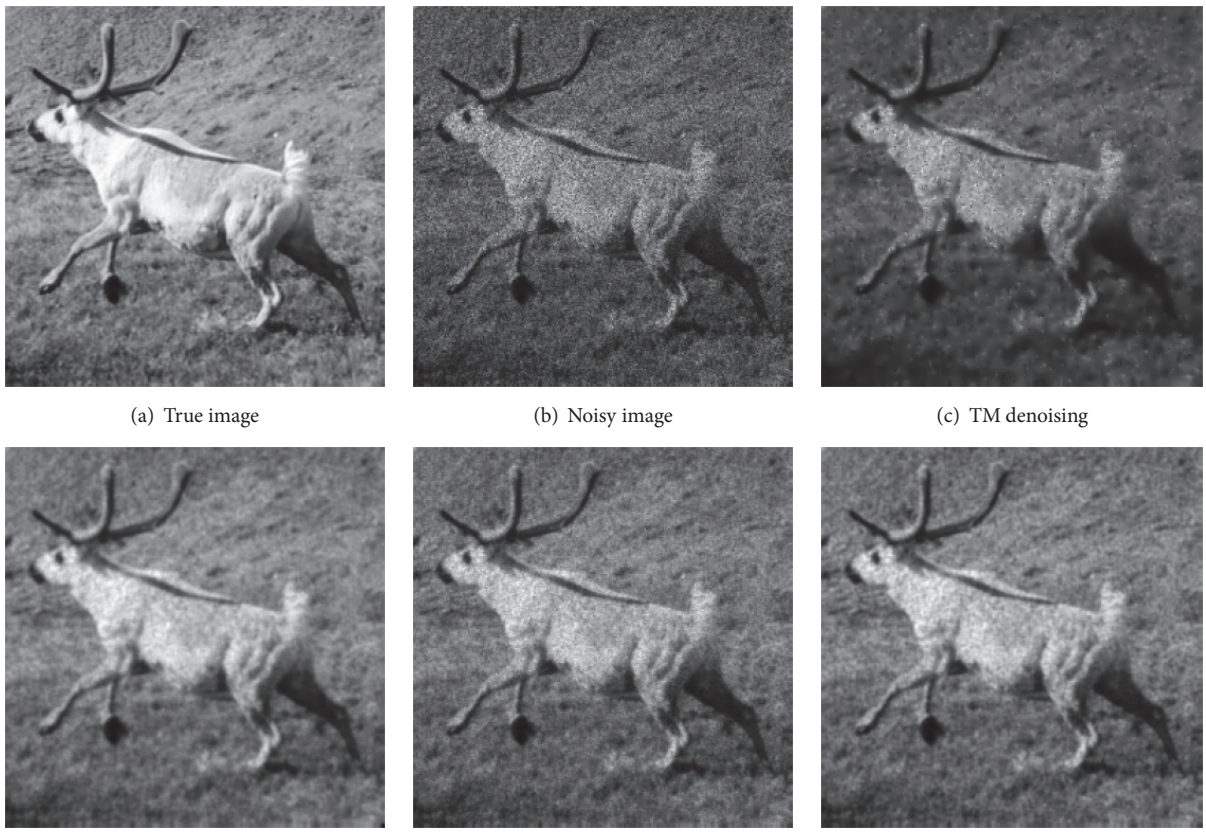
Figure 3: Denoising of multiplicative Gamma noise for the Jet Plane image with a 512 × 512 size.



(a) True image (b) Noisy image (c) TM denoising

(d) Split Bregman denoising (e) RHPM(1) denoising (f) RHPM(2) denoising

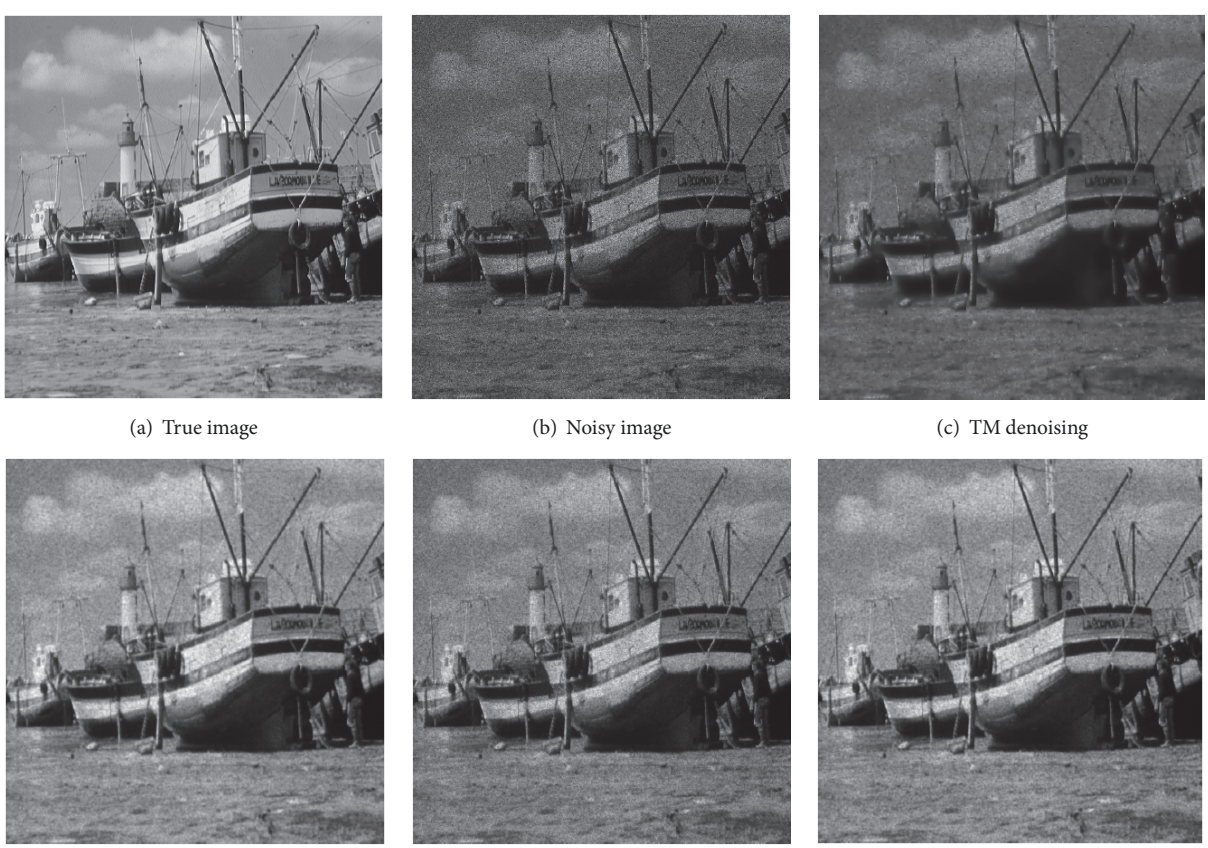
Figure 4: Denoising of multiplicative Gamma noise for the Lake image with a 512 × 512 size.



(a) True image (b) Noisy image (c) TM denoising

(d) Split Bregman denoising (e) RHPM(1) denoising (f) RHPM(2) denoising

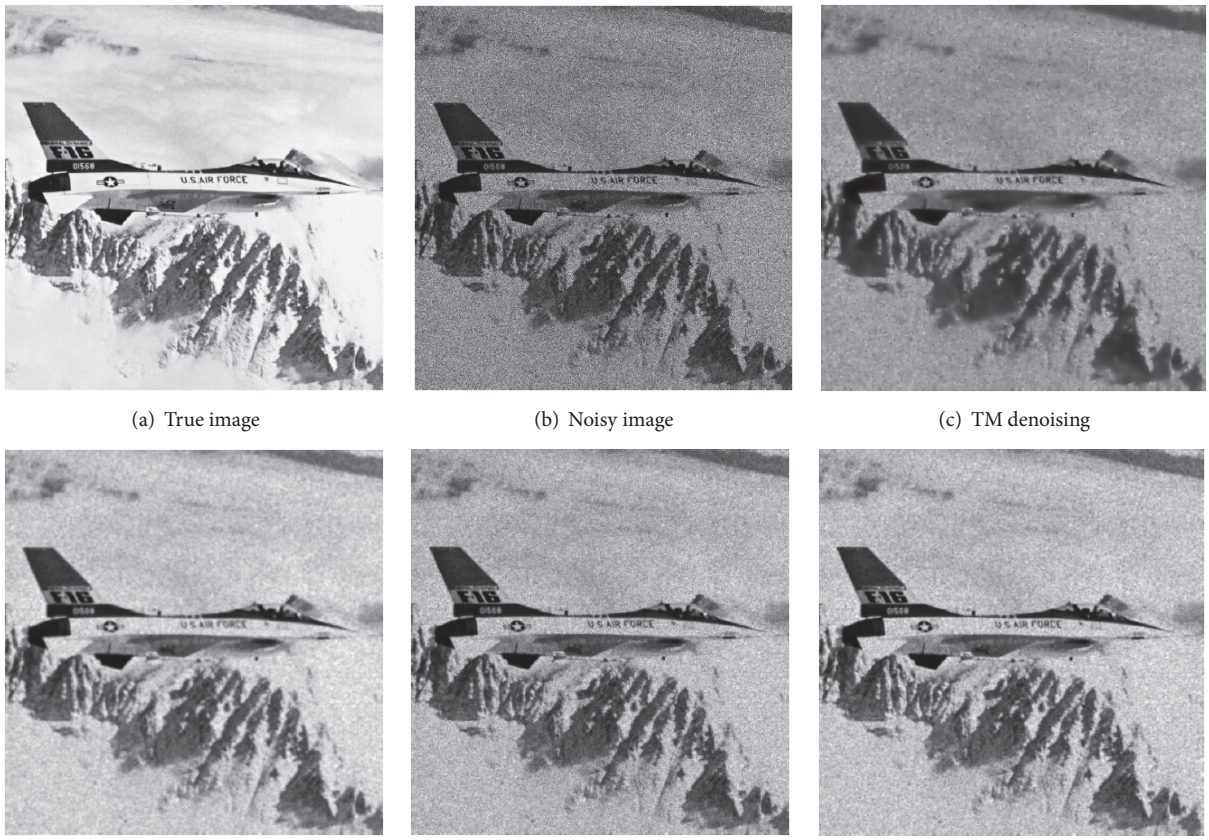
Figure 5: Denoising of multiplicative Gaussian white noise for the Caribou image with a 256×256 size.



(a) True image (b) Noisy image (c) TM denoising

(d) Split Bregman denoising (e) RHPM(1) denoising (f) RHPM(2) denoising

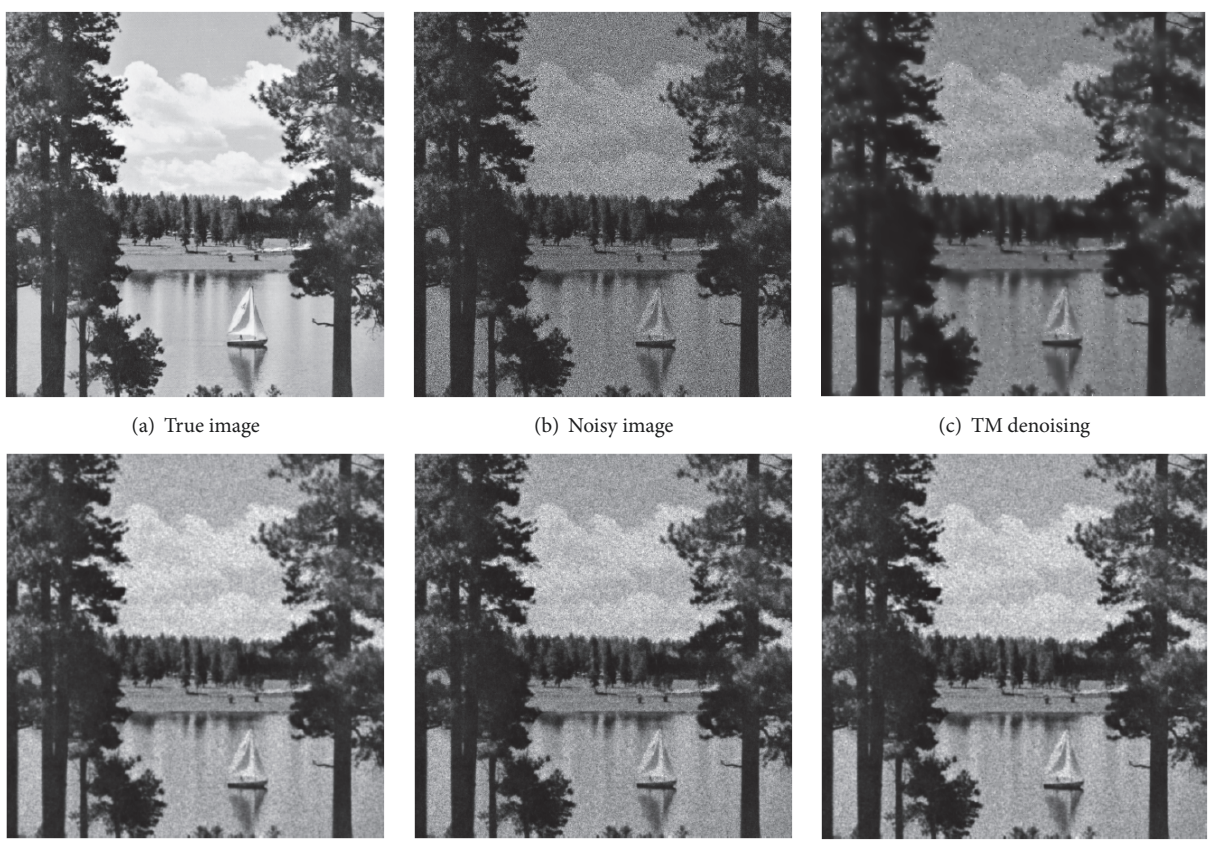
Figure 6: Denoising of multiplicative Gaussian white noise for the Boat image with a 512×512 size.



(a) True image (b) Noisy image (c) TM denoising

(d) Split Bregman denoising (e) RHPM(1) denoising (f) RHPM(2) denoising

Figure 7: Denoising of multiplicative Gaussian white noise for the Jet Plane image with a 512 × 512 size.



(a) True image (b) Noisy image (c) TM denoising

(d) Split Bregman denoising (e) RHPM(1) denoising (f) RHPM(2) denoising

Figure 8: Denoising of multiplicative Gaussian white noise for the Lake image with a 512 × 512 size.

the methods considered in this paper. Overall, the split Bregman and RHPM(2) methods perform better than the RHPM(1) and TM methods. The split Bregman method denoises almost as well as RHPM(2) for most cases, while it denoises significantly better than RHPM(2) for the Jet Plane image with variance 0.03. In this regard, the split Bregman method performs more stably than RHPM(2).

7. Conclusion

In this paper, we have proposed restarted homotopy perturbation methods (RHPM) and split Bregman methods for multiplicative noise removal of the RLO and AA2 models. For the RHPM method, we have used binomial series techniques to settle the main difficulty in handling nonlinear terms. For the split Bregman method, we have used splitting techniques of singular matrices to handle the difficulty in solving ill-conditioned linear systems. Numerical experiments have shown that these techniques work well for all test problems. More specifically, the split Bregman and RHPM methods perform better than the TM method.

Numerical experiments also showed that the split Bregman and RHPM(2) methods perform better than RHPM(1). In addition, the split Bregman method denoises almost as well as RHPM(2) for most cases, but it performs more stably than RHPM(2) for noisy images with large variance. Notice that RHPM(2) has the smallest execution time because of the fastest convergence rate. Based on these facts, the RHPM(2) and split Bregman methods are preferred over RHPM(1), and the split Bregman method is preferred over RHPM(2) for noisy images with large variance. The proposed RHPM and split Bregman methods are only compared with the TM method, so future work will include comparison studies between the proposed methods and other existing methods for multiplicative noise removal models.

Data Availability

All the data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A1A09917364).

References

[1] G. Aubert and P. Kornprobst, *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations*, Springer, NY, USA, 2006.

[2] T. F. Chan and J. Shen, *Image Processing and Analysis: Variational, PDE, Wavelet, and Stochastic Methods*, SIAM, Pa, USA, 2005.

[3] K. Bredies, K. Kunisch, and T. Pock, "Total generalized variation," *SIAM Journal on Imaging Sciences*, vol. 3, no. 3, pp. 492–526, 2010.

[4] A. Chambolle and P.-L. Lions, "Image recovery via total variation minimization and related problems," *Numerische Mathematik*, vol. 76, no. 2, pp. 167–188, 1997.

[5] J. Besag, "Digital image processing: Towards Bayesian image analysis," *Journal of Applied Statistics*, vol. 16, no. 3, pp. 395–407, 1989.

[6] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 721–741, 1984.

[7] J. Lu, Y. Xu, J. B. Weaver, and D. M. Healy, *Noise Reduction by Constrained Reconstructions in the Wavelet-Transform Domain*, Dartmouth Math Department, Lake Placid, NY, USA, 1992.

[8] S. Mallat and W. L. Hwang, "Singularity detection and processing with wavelets," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 617–643, 1992.

[9] J. Cai, S. Osher, and Z. Shen, "Split bregman methods and frame based image restoration," *Multiscale Modeling and Simulation: A SIAM Interdisciplinary Journal*, vol. 8, no. 2, pp. 337–369, 2009.

[10] B. Dong and Z. Shen, "MRA Based Wavelet Frames and Applications," in *IAS Lecture Notes Series, Summer Program on The Mathematics of Image Processing*, Park City Mathematics Institute, Salt Lake City, 2010.

[11] Y. D. Han and J. H. Yun, "Performance of Restarted Homotopy Perturbation Method for TV-Based Image Denoising Problem," *Mathematical Problems in Engineering*, vol. 2015, Article ID 207541, 18 pages, 2015.

[12] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1–4, pp. 259–268, 1992.

[13] G. Aubert and J. Aujol, "A variational approach to removing multiplicative noise," *SIAM Journal on Applied Mathematics*, vol. 68, no. 4, pp. 925–946, 2008.

[14] A. Sarti, C. Corsi, E. Mazzini, and C. Lamberti, "Maximum likelihood segmentation of ultrasound images with rayleigh distribution," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 52, no. 6, pp. 947–960, 2005.

[15] E. Bratsolis and M. Sigelle, "A spatial regularization method preserving local photometry for Richardson-Lucy restoration," *Astronomy & Astrophysics* , vol. 375, no. 3, pp. 1120–1128, 2001.

[16] L. A. Shepp and Y. Vardi, "Maximum likelihood reconstruction for emission tomography," *IEEE Transactions on Medical Imaging*, vol. 1, no. 2, pp. 113–122, 1982.

[17] L. Rudin, P.-L. Lions, and S. Osher, "Multiplicative denoising and deblurring: theory and algorithms," in *Geometric Level Set Methods in Imaging, Vision, and Graphics*, pp. 103–119, Springer, NY, USA, 2003.

[18] Z. Jin and X. Yang, "Analysis of a new variational model for multiplicative noise removal," *Journal of Mathematical Analysis and Applications*, vol. 362, no. 2, pp. 415–426, 2010.

[19] Y. Huang, M. K. Ng, and Y. Wen, "A new total variation method for multiplicative noise removal," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 20–40, 2009.

[20] T. F. Chan and P. Mulet, "On the convergence of the lagged diffusivity fixed point method in total variation image restoration,"

SIAM Journal on Numerical Analysis, vol. 36, no. 2, pp. 354–367, 1999.

[21] T. F. Chan, G. H. Golub, and P. Mulet, "A nonlinear primal-dual method for total variation-based image restoration," *SIAM Journal on Scientific Computing*, vol. 20, no. 6, pp. 1964–1977, 1999.

[22] A. Chambolle, "An algorithm for total variation minimization and applications," *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1-2, pp. 89–97, 2004.

[23] Y. Wang, J. Yang, W. Yin, and Y. Zhang, "A new alternating minimization algorithm for total variation image reconstruction," *SIAM Journal on Imaging Sciences*, vol. 1, no. 3, pp. 248–272, 2008.

[24] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, "An iterative regularization method for total variation-based image restoration," *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 460–489, 2005.

[25] T. Goldstein and S. Osher, "The Split Bregman Method for L1 Regularized Problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 2, pp. 323–343, 2009.

[26] Y.-M. Huang, H.-Y. Yan, and T. Zeng, "Multiplicative noise removal based on unbiased Box-Cox transformation," *Communications in Computational Physics*, vol. 22, no. 3, pp. 803–828, 2017.

[27] S. Setzer, G. Steidl, and T. Teuber, "Deblurring Poissonian images by split Bregman techniques," *Journal of Visual Communication and Image Representation*, vol. 21, no. 3, pp. 193–199, 2010.

[28] J. Shi and S. Osher, "A nonlinear inverse scale space method for a convex multiplicative noise model," *SIAM Journal on Imaging Sciences*, vol. 1, no. 3, pp. 294–321, 2008.

[29] Y. Wang, W. Yin, and Y. Zhang, "A Fast Algorithm for Image Deblurring with total variation regularization," CAAM Technical Report TR07-10, Rice University, Houston, USA, 2007.

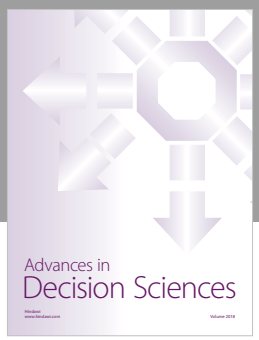
[30] A. Berman and R. J. Plemmons, *Nonnegative Matrices in Mathematical Sciences*, Academic Press, NY, USA, 1979.

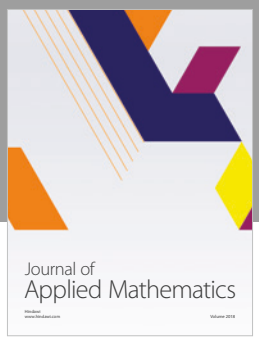
[31] H. B. Keller, "On the solution of singular and semidefinite linear systems by iteration," *SIAM Journal on Numerical Analysis*, vol. 2, pp. 281–290, 1965.

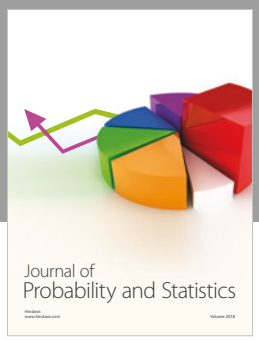
[32] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 409–436 (1953), 1952.

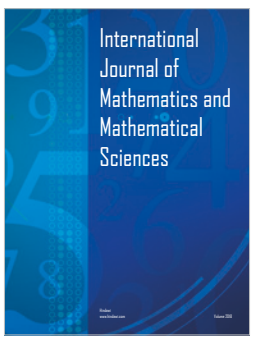
[33] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, USA, 2nd edition, 1996.

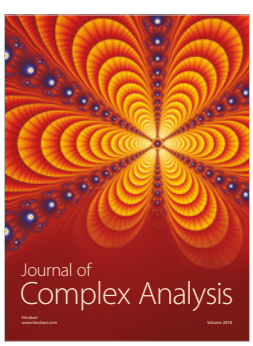
[34] J. H. Wilkinson, *The Algebraic Eigenvlaue Problem*, Clarendon Press, Oxford, UK, 1965.

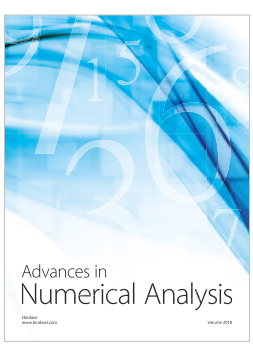


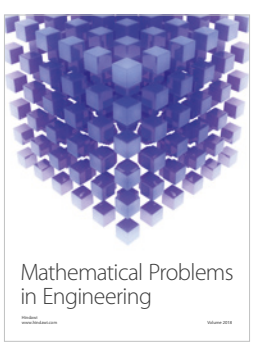


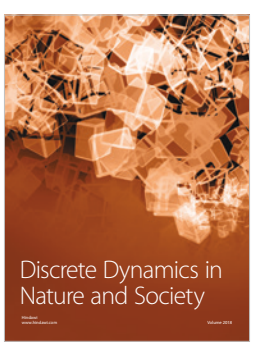


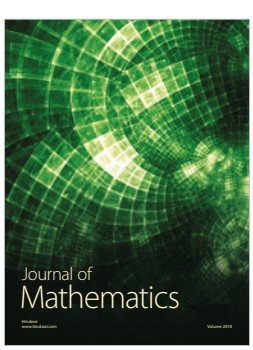


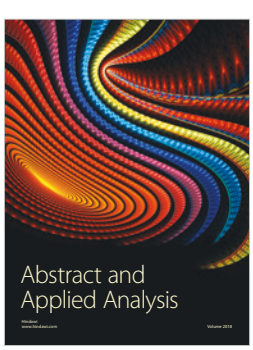


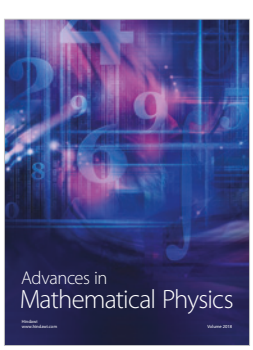












Hindawi

Submit your manuscripts at
www.hindawi.com