New Splitting Algorithms for Multiplicative Noise Removal Based on Aubert-Aujol Model

Yu Gan¹, Jie Zhang² and Huibin Chang¹,*

¹ School of Mathematical Sciences, Tianjin Normal University, Tianjin, China
² Department of Mathematics, The University of Hong Kong, Hong Kong, SAR, China

Received 29 July 2021; Accepted (in revised version) 8 December 2021

Abstract. In this paper, we propose new algorithms for multiplicative noise removal based on the Aubert-Aujol (AA) model. By introducing a constraint from the forward model with an auxiliary variable for the noise, the NEMA (short for Noise Estimate based Multiplicative noise removal by alternating direction method of multipliers (ADMM)) is firstly given. To further reduce the computational cost, an additional proximal term is considered for the subproblem with regard to the original variable, the NEMAₙ (short for a variant of NEMA with fully splitting form) is further proposed. We conduct numerous experiments to show the convergence and performance of the proposed algorithms. Namely, the restoration results by the proposed algorithms are better in terms of SNRs for image deblurring than other compared methods including two popular algorithms for AA model and three algorithms of its convex variants.

AMS subject classifications: 15A29, 65K10, 68U10, 94A08

Key words: Alternating direction method of multipliers, image denoising and deblurring, multiplicative noise, total variation.

1. Introduction

Image restoration is of great significance in image processing, as the noise and blur are ubiquitous in image acquisition and transmission. In most existing works, additive noise models [7, 24, 27] have been widely studied, aiming to recover the original image $u$ from the observed image $f$ corrupted by the additive noise $v$. As one of the typical non-Gaussian noise, the multiplicative noise [2, 10, 19, 21, 36, 40, 44] have also been widely studied. It has severe pollution on the clean images, such degradation can be formulated as the clean image $u$ blurred by a known linear operator $A \in \mathbb{R}^{N \times N}$ and
further corrupted by the multiplicative noise $\eta$, i.e.

$$f = (Au) \circ \eta,$$

(1.1)

where $f$, $u$, and $\eta \in \mathbb{R}^N$ all denote the $\sqrt{N} \times \sqrt{N}$ image rewritten as a vector in an alphabetical order, $N$ denotes the number of pixels, $\circ$ denotes the Hadamard product of two vectors and $\eta$ denotes the noise following independent and identically distributed Gamma distribution on each pixel with mean 1. A typical example of multiplicative noise is the speckle noise in synthetic aperture radar images [29]. In addition, the multiplicative noise also appears in ultrasound imaging [39], laser [35], and other coherent image systems [3, 4, 38].

Researchers have developed various denoising algorithms for such noise, including total variation (TV) regularization based methods [8], non-local low rank based methods [45], diffusion based methods [46, 49], non-local patch-based methods [9, 30], and deep learning based methods [41]. Among them, TV regularized model is one of the most widely used methods due to its edge-preserving and scale-dependent properties [8]. Rudin et al. [33] firstly applied the TV based method to multiplicative noise removal. Aubert and Aujol [1] derived a non-convex variational model (AA model) via the maximum a posteriori (MAP) estimation for Gamma noise. Huang et al. [16] established a strictly convex model (HNW model) by introducing a new variable and a new fitting term. By utilizing the statistical properties of the multiplicative Gamma noise, Dong and Zeng [10] proposed a convex model (DZ model) by adding a quadratic penalty term. Other convex approach was proposed by Zhao et al. [48] (ZWN model), where they decoupled variables by transforming the form of the multiplicative noise equation. Besides, Li and Lou [20] proposed the AA-DCA algorithm by reformulating the AA model into the difference of convex (DC) functions programming. Lu et al. [25] proposed a variational convex model by setting a new term to replace the quadratic penalty in DZ, and transforming it into a exp-model according to the fact that pointed in [16, 36] the exponent-like models usually perform better than logarithm-like counterparts in terms of the quality of denoised images.

In many applications, restoration results demonstrate that non-convex methods outperform convex one, e.g. image denoising [7, 23, 26–28] and image segmentation [31, 37]. Hence in this paper, we are interested in designing more efficient algorithms for the original nonconvex AA model, in order to achieve optimal capability of the multiplicative noise removal. By introducing a constraint from the forward model with an auxiliary variable for the noise, we propose an efficient splitting algorithm based on alternating direction method of multipliers (ADMM) [5, 11, 13, 14, 43]. In order to avoid extra computational cost of inner loop due to the existence of the TV term, we further present a fully splitting form by introducing constraint of the gradient and additional proximal terms. The main contribution of this paper is given below.

- We design an operator splitting algorithm for Noise Estimate based Multiplicative noise removal by ADMM (NEMA) for the original AA model by introducing a constraint with an auxiliary variable for the noise such that the subproblem of the noise variable has a closed-form solution.
We establish a variant of NEMA with fully splitting form (NEMA_f) by the proximal linearization technique and the constraint of the gradient, each subproblem of which can be efficiently solved with closed form solutions such that the computational cost is reduced greatly.

Experimental results demonstrate that the SNR value and the speeds of convergence for proposed algorithms are better in the case of image deblurring compared with the some popular variational methods. Although for the proposed NEMA inner loop is needed, numerically very few inner iterations can guarantee the convergence of the overall algorithm. The numerical tests of the objective functional decay and error changes are also conducted to show the convergence of the proposed algorithms.

The rest structure of this paper is as follows. In Section 2, we first review these classical multiplicative noise removal models obtained from AA and on this basis review those models which can be extended to blur removal. In Section 3, we introduce our proposed algorithms NEMA and NEMA_f for multiplicative noise and blur removal. Numerical results are presented in Section 4 to demonstrate the performance of the proposed algorithms in SNRs and convergence speeds. Finally, conclusions are given in Section 5.

2. Review of existing variational models

The AA model is strictly derived from MAP, so it gives an accurate estimate for multiplicative gamma noise. Its discrete form can be expressed below

$$\min_u E(u) := \lambda \left\langle \ln(Au) + \frac{f}{Au}, 1 \right\rangle + \text{TV}(u),$$

where $1 \in \mathbb{R}^N$ is the all-ones vector, $\langle \cdot, \cdot \rangle$ denotes the standard inner product over $\mathbb{R}^N$ and $\lambda > 0$ is the balancing parameter. Noting that when consider purely denoising problem, we can simply set $A$ as the identity operator. The second term $\text{TV}(u)$ is the isotropic TV in the discrete setting, more specifically

$$\text{TV}(u) = \sum_{1 \leq i, j \leq \sqrt{N}} |(\nabla u)[i, j]|_2$$

$$= \sum_{1 \leq i, j \leq \sqrt{N}} \sqrt{|(\nabla u)_x[i, j]|^2 + |(\nabla u)_y[i, j]|^2} \quad (2.2)$$

with $| \cdot |_2$ denoting the Euclidean norm in $\mathbb{R}^2$, where the discrete gradient operator $\nabla : \mathbb{R}^N \to \mathbb{R}^{N,2}$ is defined by

$$\nabla u[i, j] = ((\nabla u)_x[i, j], (\nabla u)_y[i, j])$$

with

$$(\nabla u)_x[i, j] = \begin{cases} u[i + 1, j] - u[i, j], & \text{if } i < \sqrt{N}, \\ u[1, j] - u[\sqrt{N}, j], & \text{if } i = \sqrt{N}, \end{cases}$$

and

$$(\nabla u)_y[i, j] = \begin{cases} u[i, j + 1] - u[i, j], & \text{if } j < \sqrt{N}, \\ u[i, 1] - u[i, \sqrt{N}], & \text{if } j = \sqrt{N}. \end{cases}$$
\[(\nabla u)_y[i,j] = \begin{cases} 
 u[i, j + 1] - u[i, j], & \text{if } j < \sqrt{N}, \\
 u[i, 1] - u[i, \sqrt{N}], & \text{if } j = \sqrt{N}
\end{cases}
\]
for \(i,j = 1, \ldots, \sqrt{N}\). Here \(u[i,j]\) refers to the \((i\sqrt{N} + j)\)-th entry of the vector \(u\) (it is the \((i,j)\)-th pixel location of the image), and both \((\nabla u)_x, (\nabla u)_y\) are rewritten as vectors in alphabet orders. More generally, the discretization of the gradient can be easily extended to arbitrary dimension according to the above setting. A gradient projection algorithm has been applied to solve the AA model, producing more accurate restoration results compared to the ROF model [34].

To overcome the non-convexity of AA model, researchers have explored several approaches, mainly consisted of the following three models:

- **HNW model**: Huang et al. [16] introduced a nonlinear transform as \(z = \ln u\) and then replaced the TV term of the original variable by that of the auxiliary variable \(w\). In order to further decouple the nonlinear and non-differential terms, an approximation model with the penalization technique was proposed as the following convex minimization model:

\[
\min_{z,w} \langle z + fe^{-z}, 1 \rangle + \alpha_1 \| z - w \|_2^2 + \alpha_2 \text{TV}(w)
\]

with \(\alpha_1, \alpha_2\) as the balanced parameters and \(\| \cdot \|_2\) denotes the standard \(\ell_2\) norm in Euclidean space. An alternating minimization algorithm was also employed to solve the above TV minimization problem efficiently.

- **DZ model**: Other than the convexification by the nonlinear transform, based on the statistical property of the multiplicative noise, Dong and Zeng [10] added a quadratic penalty term to AA model, and then obtained the following convex model:

\[
\min_u \left\langle \ln(Au) + \frac{f}{Au}, 1 \right\rangle + \alpha_1 \left\| \sqrt{\frac{Au}{f}} - 1 \right\|_2^2 + \lambda \text{TV}(u),
\]

such minimization problem was solved by the famous primal-dual algorithm [12].

- **ZWN model**: Considering the specific distribution of the multiplicative noise, by rewriting the blur and multiplicative noise equation, Zhao et al. [48] established the following convex model:

\[
\min_{u,h} \frac{1}{2} \| h - \nu 1 \|_2^2 + \alpha_1 \| Fu - Au \|_1 + \alpha_2 \text{TV}(u),
\]

where \(\| \cdot \|_1\) is the \(\ell_1\) norm, \(F\) is a diagonal matrix where the main diagonal entries of \(F\) are given by \(f[i]\), \(h\) is a vector with \(h[i] = 1/\eta[i]\), \(\nu\) can be set to be the mean value of \(h\). Here and in what follows, if \(X \in \mathbb{R}^N\), \(X[i]\) denotes the \(i\)-th entry of the vector \(X\) for \(i = 1, \ldots, N\).
These convex model proposed above can be very efficiently solved by operator-splitting algorithms, which were also very stable with regard to initialization and parameters due to the convexity. However, it has been observed in many applications [23, 31, 32, 37, 45] that the non-convex models were able to produce better results than convex ones with proper reformulation or initialization. Namely for the AA model, in order to apply the difference of convex algorithm (DCA) [17, 18], Li et al. [20] considered the decomposition: \( M(u) = G(u) - H(u) \), where \( G(u) = TV(u) + \langle f, Au \rangle \) and \( H(u) = -\lambda (\ln(Au), 1) \), then the model can be efficiently solved by the iterative algorithm obtained from DCA.

3. Proposed algorithms

In this section, we will introduce the proposed algorithms NEMA and NEMA_f, both of which are designed to solve the original AA model. Here, the scaled ADMM is used for both inner and outer iterations.

The main idea of designing the new algorithm starts from the original forward model (1.1). One can rewrite (2.1) as the following equivalent constrained optimization problem:

\[
\min_{u, \eta} J(\eta) + TV(u), \quad \text{s.t.} \quad f = (Au) \circ \eta, \quad (3.1)
\]

where \( J(\eta) := \lambda (\ln \eta + \eta, 1) \). If using operator-splitting algorithms, the subproblem with regard to the variable \( \eta \) has a closed-form solution, for the objective function of the sum of \( J(\eta) \) and a quadratic penalization of the constraint, that essentially solves algebraic equations element-wisely. After such a simple observation, we will establish two algorithms based on ADMM.

3.1. NEMA

By introducing the multiplier \( \Theta_{\eta} \), the augmented Lagrangian of the above constrained optimization problem in (3.1) is given below

\[
L_{\alpha\eta}(u, \eta; \Theta_{\eta}) = J(\eta) + TV(u) + \alpha\eta \langle \Theta_{\eta}, f - (Au) \circ \eta \rangle + \frac{\alpha\eta}{2} \|f - (Au) \circ \eta\|^2 \quad (3.2)
\]

with the penalization parameter \( \alpha_{\eta} > 0 \). Then consequently a saddle point problem has to be optimized as

\[
\max_{\Theta_{\eta}} \min_{u, \eta} L_{\alpha\eta}(u, \eta; \Theta_{\eta}).
\]

In order to solve the saddle point problem, given the previous iterative solution \((u_k, \eta_k)\), the scaled ADMM updates the sequence \((u_{k+1}, \eta_{k+1})\) by solving two subproblems with regard to \( u, \eta \), and multiplier updates, which is given below

\[
u_{k+1} = \arg \min_u L_{\alpha\eta}(u, \eta_k; \Theta_{\eta_k}),
\]

\[
\eta_{k+1} = \arg \min_{\eta} L_{\alpha\eta}(u_{k+1}, \eta; \Theta_{\eta_k}),
\]
\[ \Theta_{\eta,k+1} = \Theta_{\eta,k} + f - (A u_{k+1}) \circ \eta_{k+1}. \] (3.3c)

First, we consider the \( u \)-subproblem. From (3.3a), we can get
\[
u_{k+1} = \arg \min_u \left\{ TV(u) + \frac{\alpha}{2} \| f + \Theta_{\eta,k} - (A u) \circ \eta_k \|_2^2 \right\}. \tag{3.4}
\]

Since (3.4) is convex, many first-order algorithms can be adopted to solve such an optimization problem [6]. This optimization problem does not have a closed form solution, and one has to develop iterative algorithms as an inner loop. Hence, by introducing an auxiliary variable \( p := (p_1, p_2) \in \mathbb{R}^{N,2} \) with constraint condition \( p[i] = \nabla u[i] \), the \( u \)-subproblem reduced to the following optimization problem:
\[
\min_{u,p} \left\{ \sum_{i=1}^{N} \| p[i] \|_2 + \frac{\alpha}{2} \| f + \Theta_{\eta,k} - (A u) \circ \eta_k \|_2^2 \right\}, \tag{3.5}
\]
subject to \( p[i] = \nabla u[i], \quad \forall \ 1 \leq i \leq N. \)

Similarly we consider the scaled ADMM. By introducing the multiplier \( \Theta_p \), the augmented Lagrangian of (3.5) with the penalization parameters \( \alpha_p > 0 \) is given below
\[
L_{\alpha_p}(u,p; \Theta_p) = \sum_{i=1}^{N} |p[i]|_2 + \frac{\alpha}{2} \| f + \Theta_{\eta,k} - (A u) \circ \eta_k \|_2^2 \\
+ \alpha_p (\Theta_p - p - \nabla u) + \frac{\alpha_p}{2} \| p - \nabla u \|_2^2. \tag{3.6}
\]

Given the previous iterative solution \( (u^n, p^n) \), then the ADMM updates the sequence \( (u^{n+1}, p^{n+1}) \) by solving two subproblems with regard to \( u, p \), and multipliers update. We further introduce a proximal term to the \( u \)-subproblem, and then the update scheme is given below
\[
u^{n+1} = \arg \min_u L_{\alpha_p}(u, p^n; \Theta_p^n) + \frac{1}{2} \left\| u - u^n \right\|_{M^n}^2, \tag{3.7a}
\]
\[p^{n+1} = \arg \min_p L_{\alpha_p}(u^{n+1}, p; \Theta_p^n), \tag{3.7b}
\]
\[\Theta_p^{n+1} = \Theta_p^n + p^{n+1} - \nabla u^{n+1}. \tag{3.7c}
\]

where \( \| x \|_{M^n} = \langle M^n x, x \rangle^{\frac{1}{2}} \). The matrix \( M^n \in \mathbb{R}^{N \times N} \) is a positive definite matrix updated as the iteration goes, and its specific forms will be given in the following part.

From (3.7a), we can get
\[
u^{n+1} = \arg \min_u \left\{ \frac{\alpha_p}{2} \left\| f + \Theta_{\eta,k} - (A u) \circ \eta_k \right\|_2^2 + \frac{\alpha_p}{2} \left\| p^n + \Theta_p^n - \nabla u \right\|_2^2 \\
+ \frac{1}{2} \left\| u - u^n \right\|_{M^n}^2 \right\}. \tag{3.8}
\]
We can easily obtain its optimality condition as
\[ \alpha_n A^T (\eta_k^2 \circ (Au)) + Mn u - \alpha_p \Delta u = b^n \]  
(3.9)
with
\[ b^n := \alpha_n A^T ((f + \Theta_{\eta,k}) \circ \eta_k) - \alpha_p \nabla \cdot p^n - \alpha_p \nabla \cdot \Theta^n_p + Mn u^n. \]
Here \( \eta_k^2 \) means that the element-wise square of \( \eta_k \). In order to get rid of solving the linear system iteratively, choose the matrix \( Mn \) as
\[ Mn := \alpha_n A^T \left( (\mu_k - \eta_k^2) \circ (Au) \right), \quad \forall u \in \mathbb{R}^N \]
with a small positive scalar \( \mu_k = (1 + \epsilon) \max_i (\eta_k[i])^2 \) and \( \epsilon > 0 \). Hence the first two left terms of (3.9) reduce to \( \alpha_n \mu_k A^T A - \alpha_p \Delta \)
by fast Fourier transform (FFT).
Then, we consider the \( p \)-subproblem
\[ p^{n+1} = \arg\min_p \left\{ \sum_{i=1}^N \|p[i]\|_2 + \frac{\alpha_n}{2} \|p + \Theta^n_p - \nabla u^{n+1}\|_2^2 \right\}. \]  
(3.11)
The solution of this optimization problem is exactly the soft thresholding of \( \nabla u^{n+1} - \Theta^n_p \), i.e.
\[ p^{n+1} = \text{Thresh}_{\frac{\alpha_n}{2}} (\nabla u^{n+1} - \Theta^n_p) \]  
(3.12)
with
\[ \text{Thresh}_\delta(p) := \left( \max \{0, |p| - \delta\} \circ \left( \frac{p_1}{|p|} \right), \max \{0, |p| - \delta\} \circ \left( \frac{p_2}{|p|} \right) \right), \]
where
\[ |p[i]| = \sqrt{|p_1[i]|^2 + |p_2[i]|^2}, \quad \forall 1 \leq i \leq N \]
with \( p := (p_1, p_2) \in \mathbb{R}^{N,2} \).
Secondly, we consider the \( \eta \)-subproblem, from (3.3b), we have
\[ \eta_{k+1} = \arg\min_{\eta} \left\{ J(\eta) + \frac{\alpha_n}{2} \|f + \Theta_{\eta,k} - (Au_{k+1}) \circ \eta\|_2^2 \right\}. \]  
(3.13)
Then one can easily get the solution to the above convex differential problem by solving
the optimality condition below: for all \( 1 \leq i \leq N \),
\[ \alpha_n ((Au_{k+1})[i])^2 (\eta[i])^2 \]
\[ - \left( \alpha_n ((Au_{k+1})[i]) f[i] + \alpha_n ((Au_{k+1})[i]) \Theta_{\eta,k}[i] - \lambda \right) \eta[i] - \lambda = 0. \]  
(3.14)
Readily we can get $\eta_{k+1}[i] = 1$, for $i \in \{i : (Au_{k+1})[i] = 0\}$. Otherwise, according to the root-finding formula of the quadratic equation of one variable, we can get

$$
\eta_{k+1}[i] = \frac{\Phi_k[i]}{2\alpha_\eta((Au_{k+1})[i])^2},
$$

where

$$
\Phi_k[i] = \psi_k[i] + \sqrt{\left(\psi_k[i]\right)^2 + 4\lambda\alpha_\eta((Au_{k+1})[i])^2},
$$

and

$$
\psi_k[i] = \alpha_\eta((Au_{k+1})[i])f[i] + \alpha_\eta((Au_{k+1})[i])\Theta_{\eta,k}[i] - \lambda.
$$

Therefore, we have

$$
\eta_{k+1}[i] = \begin{cases} 
1, & \text{if } (Au_{k+1})[i] = 0, \\
\frac{\Phi_k[i]}{2\alpha_\eta((Au_{k+1})[i])^2}, & \text{otherwise}.
\end{cases} \tag{3.15}
$$

Algorithm 3.1 summarizes the overall NEMA algorithm.

**Algorithm 3.1 NEMA**

**Input:** $maxInIter$, $maxOutIter$, $tol_{out}$, $A$, and parameters $\lambda$, $\alpha_\eta$, $\alpha_p$, $u_0 = f$.

**Output:** the reconstructed image $u_{k+1}$.

1: for $maxOutIter \leftarrow 1$ do
2: for $maxInIter \leftarrow 1$ do
3: Solve $u^{n+1}$ by using FFT for (3.10).
4: Solve $p^{n+1}$ by (3.12).
5: Update the multiplier by

$$
\Theta_p^{n+1} = \Theta_p^n + p^{n+1} - \nabla u^{n+1}.
$$

6: Solve $\eta_{k+1}$ by (3.15).
7: Update the multiplier by

$$
\Theta_{\eta,k+1} = \Theta_{\eta,k} + f - (Au_{k+1}) \circ \eta_{k+1}.
$$

8: if $\frac{\|u_{k+1} - u_k\|}{\|u_k\|} < tol_{out}$ then break.

return $u_{k+1}$.

3.2. NEMA$f$

In order to get a fully splitting algorithm without inner loop, we introduce one more auxiliary variable $p$ satisfying the relation as $p = \nabla u$. And then, we rewrite (3.1) as the following equivalent constrained problem:

$$
\min_{u,\eta,p} J(\eta) + F(p), \tag{3.16a}
$$
New Splitting Algorithms for Multiplicative Noise Removal

\[ s.t. \quad f = (Au) \circ \eta, \quad p = \nabla u \] (3.16b)

with \( F(p) := \sum_{i=1}^{N} |p[i]|^2 \).

By introducing the multipliers \( \Lambda_\eta, \Lambda_p \), the augmented Lagrangian of the above constrained problem with the penalization parameters \( \beta_\eta, \beta_p > 0 \) is given below

\[
L_{\beta_\eta, \beta_p}(u, \eta, p; \Lambda_\eta, \Lambda_p) = J(\eta) + F(p) + \beta_\eta \langle \Lambda_\eta, f - (Au) \circ \eta \rangle + \frac{\beta_p}{2} \|p - \nabla u\|^2_2.
\] (3.17)

Given the previous iterative solution \((u_k, \eta_k, p_k)\), then generalized ADMM with additional proximal term for \( u \)-subproblem, updates the sequence \((u_{k+1}, \eta_{k+1}, p_{k+1})\) by solving three subproblems with regard to \( u, \eta, p \), and multipliers update, which is given below

\[
u_{k+1} = \arg \min_u L_{\beta_\eta, \beta_p}(u, \eta_k, p_k; \Lambda_{\eta, k}, \Lambda_{p, k}) + \frac{1}{2\beta_\eta} \|u - u_k\|^2_{M_k}, \]
(3.18a)

\[
\eta_{k+1} = \arg \min_\eta L_{\beta_\eta, \beta_p}(u_{k+1}, \eta, p_k; \Lambda_{\eta, k}, \Lambda_{p, k}), \]
(3.18b)

\[
p_{k+1} = \arg \min_p L_{\beta_\eta, \beta_p}(u_{k+1}, \eta_{k+1}, p; \Lambda_{\eta, k}, \Lambda_{p, k}), \]
(3.18c)

\[
\Lambda_{\eta, k+1} = \Lambda_{\eta, k} + f - (Au_{k+1}) \circ \eta_{k+1}, \]
(3.18d)

\[
\Lambda_{p, k+1} = \Lambda_{p, k} + p_{k+1} - \nabla u_{k+1}.
\]
(3.18e)

The overall iterative scheme above is a full splitting form of NEMA, named NEMA_f. Specific solving process is given below and the techniques used are similar to NEMA.

For \( u \)-subproblem, similarly, from (3.18a) we can get its optimality condition

\[
\beta_\eta A^T(\eta_k^2 \circ (Au)) + M_k u - \beta_p \Delta u = c_k
\]
(3.19)

with

\[
c_k := \beta_\eta A^T((f + \Lambda_{p, k}) \circ \eta_k) - \beta_p \nabla \cdot p_k - \beta_p \nabla \cdot \Lambda_{\eta, k} + M_k u_k.
\]

Similarly to the previous subsection, choose the \( M_k \) as

\[
M_k u := \beta_\eta A^T((\mu_k - \eta_k^2) \circ (Au)),
\]

and therefore, FFT can be used to explicitly solve the problem

\[
u = (\beta_\eta \mu_k A^T A - \beta_p \Delta)^{-1} c_k.
\]
(3.20)

For \( \eta \) and \( p \) subproblems, one gets

\[
\eta_{k+1}[i] = \begin{cases} 
1, & \text{if } (Au_{k+1})[i] = 0, \\
\frac{\phi_k[i]}{2\beta_\eta((Au_{k+1})[i])^2}, & \text{otherwise,}
\end{cases}
\]
(3.21)
where
\[
\bar{\Phi}_k[i] = \bar{\psi}_k[i] + \sqrt{(\bar{\psi}_k[i])^2 + 4\lambda \beta \eta ((Au_{k+1})[i])^2},
\]
\[
\bar{\psi}_k[i] = \beta \eta ((Au_{k+1})[i]) f[i] + \beta \eta ((Au_{k+1})[i]) \Lambda_{\eta,k}[i] - \lambda,
\]
and
\[
p_{k+1} = \text{Thresh}_{\frac{1}{\beta p}}(\nabla u_{k+1} - \Lambda_{p,k}). \tag{3.22}
\]

Algorithm 3.2 summarizes the overall NEMA algorithm.

Algorithm 3.2 NEMA_f
\begin{align*}
\textbf{Input:} & \ maxOutIter, tol_{out}, A \text{ and parameters } \lambda, \beta \eta, \beta p, u_0 = f. \\
\textbf{Output:} & \ \text{the reconstructed image } u_{k+1}. \\
& \textbf{1: } \text{for } maxOutIter \leftarrow 1 \text{ do} \\
& \quad 2: \ \text{Solve } u_{k+1} \text{ by using FFT for (3.20).} \\
& \quad 3: \ \text{Solve } \eta_{k+1} \text{ by (3.21).} \\
& \quad 4: \ \text{Solve } p_{k+1} \text{ by (3.22).} \\
& \quad 5: \ \text{Update the multipliers by} \\
& \quad \quad \Lambda_{\eta,k+1} = \Lambda_{\eta,k} + f - (Au_{k+1}) \circ \eta_{k+1}, \\
& \quad \quad \Lambda_{p,k+1} = \Lambda_{p,k} + p_{k+1} - \nabla u_{k+1}. \\
& \textbf{6: } \text{if } \frac{\|u_{k+1} - u_k\|_2}{\|u_k\|_2} < tol_{out} \text{ then break.} \\
& \textbf{return } u_{k+1}. \\
\end{align*}

One can see that the ADMM is frequently used to solve many linear constrained optimization problems. The convergence of ADMM for nonconvex optimization problems can be guaranteed with some special setting. For example, Wang et al. [42] analyzed the convergence of the ADMM for a nonconvex and possibly nonsmooth optimization problem with coupled linear equality constraints. In the field of compressive sensing, Lou and Yan [22] showed the convergence of ADMM for $\ell_1 - \ell_2$ minimization problem. Hajinezhad-Shi [15] and Zhang et al. [47] proposed an algorithm based on ADMM for nonconvex optimization problems with bilinear constraints and rigorously analyzed its convergence properties.

Readily the existing convergence theorem cannot directly be applied to the proposed algorithms due to the multiplicative constraint. The introduced constraint by estimating the noise variable simplifies the iterative scheme. However, to get a strict convergence analysis, the lower bound of variable $u$ (far away from the origins) is difficult to prove, thus causing the main challenge to the convergence theory, as well as the non-differentiation of the traditional TV term in AA model. Therefore, we will leave the analysis for ADMM under the multiplicative constraint as a future work.
4. Numerical experiments

In this section, we will evaluate the performance of our proposed methods for image denoising and deblurring. We will also compare with other popular variational methods, including the nonconvex methods such as gradient projection algorithm (AA) [1] and AA-DCA [20] for the original AA model, and three convex methods such as HNW [16], ZWN [48], and DZ [10].

Setting the degraded image as the initial guess for all compared algorithms, we stop the proposed algorithms when either the iteration number reaches $\text{maxOutIter} = 1000$ or the relative error between the successive iterations

$$\text{SE} := \frac{\|u_{k+1} - u_k\|_2}{\|u_k\|_2} < \text{tol}_{\text{out}}$$

where $\text{tol}_{\text{out}} = 5 \times 10^{-4}$. All the experiments are operated using MATLAB on a laptop with Intel(R) Core(TM) i7-5600U CPU and 16GB RAM.

For the sake of fairness, we fix the model parameters for our proposed algorithms and two compared algorithms as AA, AA-DCA, which directly solve the AA model. In order to evaluate the quality of the reconstructed images, we introduce the SNR as

$$\text{SNR}(u, u_{\text{truth}}) = -10 \times \log_{10} \frac{\|u - u_{\text{truth}}\|_2^2}{\|u\|_2^2},$$

where $u$ and $u_{\text{truth}}$ denote the reconstructed image and the ground truth respectively.

We take three images including Cameraman, Parrot and Syn as the test images (Fig. 1). For convenience, scale the image intensities range from $[0, 255]$ to $[0.03, 0.9]$. In numerical experiments, the images are corrupted by random Gamma noise with mean 1, which is generated by MATLAB function Gamma and its probability density function is

$$g_{\eta}(\eta, \theta, L) = \frac{1}{\theta L \Gamma(L)} \eta^{L-1} e^{-\frac{\eta}{\theta}},$$

where $\theta$ is the scale parameter, $L$ is the number of looks and $\Gamma$ is the classical Gamma function with $\Gamma(L) = (L - 1)!$. According to the assumption in the AA model, the mean

![Figure 1: Test images.](image)
is $\theta L = 1$, variance is $\theta^2 L = \frac{1}{L}$. Therefore, in this paper, we can use only one parameter $L$ to control the noise level and a high number $L$ corresponds to low noise level. Here consider $L = 6, 10$ and 20. To gain optimal performance, we tune the parameters for all compared algorithms by hand.

For the proposed NEMA, the inner iterations will increase the computational consumption. We first show how different numbers of inner iterations affect the performance of proposed NEMA. We conduct such test with results shown in Fig. 2, where one readily sees that the SNRs of recovered images change slightly with different inner iterations. Hence, we fix the inner iteration as 3 in order to reduce the overall computational cost.

In the following parts, we will conduct the experiments for both image denoising and deblurring. Since HNW can only be used in purely denoising problem, we do not consider this method in Section 4.2.

### 4.1. Image denoising

This section is devoted to showing the image restoration capabilities of our proposed algorithms. Our empirical evidences indicate that when $\alpha_\eta$ and $\beta_\eta$ varies in $[100, 300]$ and $[10, 200]$ respectively, we can obtain the optimal restoration result with further adjustment the parameters slightly according to the ratio of $\frac{\alpha_\eta}{\alpha_p}$ and $\frac{\beta_\eta}{\beta_p}$. Specifically, the ratios $\frac{\alpha_\eta}{\alpha_p}$ for NEMA in all test images are 0.83 and 1 for $L = 6$ and $L = 10$, respectively.
respectively. When $L = 20$, the ratio is 2 for Cameraman (Fig. 1(a)) and Parrot (Fig. 1(b)), and is 1 for Syn (Fig. 1(c)). As for NEMA$_f$, in Cameraman (Fig. 1(a)), the ratio is 1 for $L = 6$ and 0.5 for $L = 10, 20$. In Parrot (Fig. 1(b)), the ratios are 3, 8 and 12 for $L = 6, 10$ and 20, respectively. In Syn (Fig. 1(c)), the ratios are 1, 2.4 and 3 for $L = 6, L = 10$ and $L = 20$, respectively.

We first consider the same AA model with different algorithms. The restoration results are shown in Fig. 3, which demonstrate that there is no obvious advantage in visual output. In Table 1, the SNR values of NEMA give the best result, while for NEMA$_f$, the SNR value is almost the highest for $L = 6$ and 10 with acceptable difference from AA-DCA at $L = 10$. Figs. 4(a)-(c) give the histories of SNR changes with respect to CPU time, where one can see that our proposed algorithms gain higher SNR values much faster than other two compared algorithms. Further, the computational

<table>
<thead>
<tr>
<th>Images</th>
<th>$L$</th>
<th>Noisy</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cameraman</td>
<td></td>
<td></td>
<td>AA</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>SNR</td>
<td>7.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0693</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>SNR</td>
<td>10.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0620</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>SNR</td>
<td>13.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0678</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>SNR</td>
<td>10.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0664</td>
</tr>
<tr>
<td>Parrot</td>
<td></td>
<td>SNR</td>
<td>7.82</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Time</td>
<td>0.0735</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>SNR</td>
<td>10.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0643</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>SNR</td>
<td>13.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0634</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>SNR</td>
<td>10.31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0671</td>
</tr>
<tr>
<td>Syn</td>
<td></td>
<td>SNR</td>
<td>7.83</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Time</td>
<td>0.0702</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>SNR</td>
<td>10.34</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0690</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>SNR</td>
<td>13.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0677</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>SNR</td>
<td>10.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0690</td>
</tr>
</tbody>
</table>
Figure 3: Result of different methods for multiplicative noise removal. Column (a): noisy image; column (b) to (e): restored image with different methods. Row 1 to 3: noise level (row 1: \(L = 6\); row 2: \(L = 10\); row 3: \(L = 20\)). The test image is Parrot (Fig. 1(b)).

Figure 4: (a)-(c): histories of SNR changes v.s. CPU time (in seconds) of proposed algorithms and AA, AA-DCA for multiplicative noise removal. (d)-(f): \(\text{SE}(\|u_{k+1} - u_k\|)\) (in log scale) changes v.s. iteration number of proposed algorithms and AA, AA-DCA for multiplicative noise removal. Column 1 to 3: noise level (column 1: \(L = 6\), column 2: \(L = 10\), column 3: \(L = 20\)). The test image is Syn (Fig. 1(c)).
New Splitting Algorithms for Multiplicative Noise Removal

Figure 5: (a)-(c): total energy of optimization problem \(E_{OP} := E(u)\) changes v.s. iteration number of different methods for multiplicative noise removal. (d)-(f): total energy of Augmented Lagrangian \(E_{AL} := L_{\alpha, \eta}(u, \eta; \Theta_\eta)\) of NEMA changes v.s. iteration number of different methods for multiplicative noise removal. (g)-(i): total energy of Augmented Lagrangian \(E_{AL} := L_{\beta, \eta, p}(u, \eta, p; \Lambda_\eta, \Lambda_p)\) of NEMA changes v.s. iteration number of different methods for multiplicative noise removal. Column 1 to 3: noise level (column 1: \(L = 6\); column 2: \(L = 10\); column 3: \(L = 20\)). The test image is Syn (Fig. 1(c)).

Time of NEMA\textsubscript{f} is much lower than that of NEMA. In addition, Figs. 4(d)-(f) show the successive errors change v.s. the iteration number, where one can also notice that the proposed algorithms converge the fastest among all compared algorithms. More results related to the objective functions and augmented Lagrangian are further provided in Fig. 5. Again they demonstrate that objective function values by the proposed algorithms decrease the fastest. We remark that AA reaches the given tolerance faster than others as shown in Table 1, however, the SNRs of recovered images by AA are the worst.

Then we compare the proposed algorithms with algorithms for different convex models including HNW, ZWN and DZ. Fig. 6 presents the degraded images and restoration results, while corresponding SNR values and computational time are recorded in Table 2. Figs. 7(a)-(c) show the histories changes of SNR and successive error changes
Figure 6: Result of different methods for multiplicative noise removal. Column (a): noisy image; column (b) to (e): restored image with different methods. Row 1 to 3: noise level (row 1: \( L = 6 \); row 2: \( L = 10 \); row 3: \( L = 20 \)). The test image is Cameraman (Fig. 1(a)).

Figure 7: (a)-(c): histories of SNR changes v.s. CPU time (in seconds) of proposed algorithms and HNW, DZ, ZWN for multiplicative noise removal. (d)-(f): \( \text{SE}(\|u_{k+1} - u_k\|/\|u_k\|) \) (in log scale) changes v.s. iteration number of proposed algorithms and HNW, DZ, ZWN for multiplicative noise removal. Column 1 to 3: noise level (column 1: \( L = 6 \), column 2: \( L = 10 \), column 3: \( L = 20 \)). The test image is Syn (Fig. 1(c)).

respectively. Results in Fig. 6 demonstrate that there are no obvious differences visually for all compared algorithms. In Table 2, the SNR values of NEMA and NEMA \( f \) are almost the highest with \( L = 6, 10 \) with acceptable difference from DZ at \( L = 10 \). In Figs. 7(a)-(c) one can see that our proposed algorithms with higher SNR values con-
Table 2: The SNR (in dB) values and CPU-time (in seconds) of the algorithms for different models (i.e. HNW, ZWN, DZ and AA) in the denoising case.

<table>
<thead>
<tr>
<th>Images</th>
<th>$L$</th>
<th>Noisy</th>
<th>Methods</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SNR</td>
<td>HNW</td>
<td>DZ</td>
<td>ZWN</td>
<td>NEMA</td>
<td>NEMA_f</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cameraman</td>
<td>6</td>
<td>7.81</td>
<td>18.55</td>
<td>18.98</td>
<td>18.79</td>
<td>19.22</td>
<td>19.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0693</td>
<td></td>
<td>0.85</td>
<td>4.25</td>
<td>2.26</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10.02</td>
<td>19.68</td>
<td>20.21</td>
<td>19.85</td>
<td>20.32</td>
<td>20.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0620</td>
<td></td>
<td>0.79</td>
<td>4.74</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0678</td>
<td></td>
<td>0.56</td>
<td>4.65</td>
<td>0.56</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>SNR</td>
<td>10.29</td>
<td>19.93</td>
<td>20.37</td>
<td>19.91</td>
<td>20.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0664</td>
<td></td>
<td>0.73</td>
<td>4.55</td>
<td>1.24</td>
</tr>
<tr>
<td>Parrot</td>
<td>6</td>
<td>7.82</td>
<td>18.13</td>
<td>18.70</td>
<td>18.27</td>
<td>18.86</td>
<td>18.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0735</td>
<td></td>
<td>0.87</td>
<td>5.44</td>
<td>1.73</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10.07</td>
<td>19.22</td>
<td>20.06</td>
<td>19.59</td>
<td>20.04</td>
<td>19.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0643</td>
<td></td>
<td>0.76</td>
<td>4.74</td>
<td>0.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0634</td>
<td></td>
<td>0.48</td>
<td>5.67</td>
<td>0.63</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>SNR</td>
<td>10.31</td>
<td>19.48</td>
<td>20.17</td>
<td>19.58</td>
<td>20.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0671</td>
<td></td>
<td>0.70</td>
<td>5.29</td>
<td>0.98</td>
</tr>
<tr>
<td>Syn</td>
<td>6</td>
<td>7.83</td>
<td>19.28</td>
<td>19.52</td>
<td>18.47</td>
<td>20.12</td>
<td>19.93</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0702</td>
<td></td>
<td>1.51</td>
<td>3.26</td>
<td>1.58</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0690</td>
<td></td>
<td>1.29</td>
<td>3.63</td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>13.01</td>
<td>22.64</td>
<td>23.67</td>
<td>22.28</td>
<td>23.01</td>
<td>22.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0677</td>
<td></td>
<td>0.90</td>
<td>4.62</td>
<td>0.82</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>SNR</td>
<td>10.29</td>
<td>20.91</td>
<td>21.46</td>
<td>20.33</td>
<td>21.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td>0.0690</td>
<td></td>
<td>1.23</td>
<td>3.84</td>
<td>1.06</td>
</tr>
</tbody>
</table>

verge much faster than HNW, ZWN and DZ, especially when $L = 6$, this advantage is the most obvious. In addition, the successive error changes with respect to iteration number in Figs. 7(d)-(f) are also comparable.

### 4.2. Image deblurring

In this section, we consider image deblurring with multiplicative noise. Here the test images are corrupted by a $7 \times 7$ Gaussian blur with a standard deviation of 1 and further degraded by multiplicative noise with $L = 6, 10$ and 20, respectively.

Heuristically in NEMA, when $\alpha_r$ varies in [50,200] the ratios $\alpha_r / \alpha_p$ in all test images are 1.3 and 1 for $L = 6$ and 10, respectively. When $L = 20$, for Cameraman (Fig. 1(a))
Table 3: The SNR (in dB) value and CPU time (in seconds) of the algorithms for AA model in the denoising and deblurring case.

<table>
<thead>
<tr>
<th>Images</th>
<th>L</th>
<th>Noisy</th>
<th>Methods</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
|        |    | SNR         | AA            | AA-DCA| NEMA  | NEMA  | NEMA
|        |    | Time        |               |       |       |       | f    |
| Cameraman | 6  | SNR 7.68    | 17.44         | 17.87 | 17.92 | 17.89 |
|         |    | Time 0.0940 | 1.52          | 5.92  | 7.09  | 3.78  |
|         | 10 | SNR 9.75    | 17.97         | 18.25 | 18.36 | 18.32 |
|         |    | Time 0.1020 | 1.25          | 5.34  | 7.09  | 3.19  |
|         | 20 | SNR 12.43   | 18.60         | 18.80 | 19.03 | 19.01 |
|         |    | Time 0.0942 | 0.82          | 4.38  | 6.79  | 3.32  |
| Average|    | SNR 9.96    | 18.00         | 18.30 | 18.44 | 18.41 |
|         |    | Time 0.0974 | 1.20          | 5.21  | 6.99  | 3.43  |
| Parrot | 6  | SNR 7.73    | 17.30         | 17.76 | 17.84 | 17.80 |
|         |    | Time 0.0958 | 2.12          | 5.78  | 7.76  | 3.95  |
|         | 10 | SNR 9.84    | 17.94         | 18.25 | 18.48 | 18.45 |
|         |    | Time 0.0980 | 1.50          | 5.25  | 7.62  | 3.71  |
|         | 20 | SNR 12.49   | 18.72         | 19.00 | 19.31 | 19.31 |
|         |    | Time 0.1007 | 1.22          | 4.10  | 5.07  | 2.33  |
| Average|    | SNR 10.02   | 17.99         | 18.33 | 18.55 | 18.52 |
|         |    | Time 0.0982 | 1.16          | 5.04  | 6.82  | 3.33  |
| Syn    | 6  | SNR 7.75    | 18.13         | 19.48 | 19.82 | 19.75 |
|         |    | Time 0.1085 | 1.42          | 7.50  | 8.54  | 4.26  |
|         | 10 | SNR 9.85    | 18.90         | 20.37 | 20.72 | 20.52 |
|         |    | Time 0.0978 | 1.14          | 7.20  | 7.26  | 3.13  |
|         | 20 | SNR 12.61   | 20.02         | 21.75 | 21.96 | 21.77 |
|         |    | Time 0.1005 | 0.38          | 12.52 | 5.56  | 2.46  |
| Average|    | SNR 10.07   | 19.02         | 20.53 | 20.83 | 20.68 |
|         |    | Time 0.1023 | 1.13          | 9.07  | 7.12  | 3.32  |

and Parrot (Fig. 1(b)), the ratio is 0.1 and for Syn (Fig. 1(c)) is 2.5. For NEMA_f, when \( \beta_\eta \) varies in [10, 100], both in Cameraman (Fig. 1(a)) and Parrot (Fig. 1(b)), the ratios \( \frac{\beta_\eta}{\beta_p} \) are 5, 2, and 0.2 for \( L = 6, 10 \) and 20, respectively. In Syn (Fig. 1(c)), the ratio is 4.5 for \( L = 6, 10 \) and 6 for \( L = 20 \).

We conduct two experiments similarly to Section 4.1. In Fig. 8, we show the blurry and restoration images for proposed algorithms and AA model based algorithms AA, AA-DCA. Related SNR values with respect to CPU time in Table 3 show that NEMA and NEMA_f give the best SNR values in all cases and NEMA_f computes much faster than AA-DCA and NEMA. In addition, Figs. 9(a)-(c) show that SNR values of our methods changes much faster than compared one. Not only that, the rate of convergence is the fastest in Figs. 9(d)-(f) and the objective function values are also decrease the fastest in Fig. 10.
Table 4: The SNR (in dB) values and CPU-time (in seconds) of the algorithms for different models (i.e. ZWN, DZ and AA) in the denoising and deblurring case.

<table>
<thead>
<tr>
<th>Images</th>
<th>$L$</th>
<th>SNR</th>
<th>Time</th>
<th>SNR</th>
<th>Time</th>
<th>SNR</th>
<th>Time</th>
<th>SNR</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Noisy</td>
<td></td>
<td>Methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ZWN</td>
<td>DZ</td>
<td>NEMA</td>
<td>NEMA$f$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cameraman</td>
<td>6</td>
<td>7.68</td>
<td>17.59</td>
<td>17.94</td>
<td><strong>18.00</strong></td>
<td>17.98</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0940</td>
<td><strong>1.40</strong></td>
<td>6.50</td>
<td>8.27</td>
<td>3.63</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>9.75</td>
<td>18.39</td>
<td>18.55</td>
<td><strong>18.61</strong></td>
<td>18.60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1020</td>
<td><strong>1.00</strong></td>
<td>7.17</td>
<td>5.14</td>
<td>1.88</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0962</td>
<td><strong>0.99</strong></td>
<td>2.98</td>
<td>3.13</td>
<td>1.35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>9.96</td>
<td>18.39</td>
<td>18.58</td>
<td><strong>18.73</strong></td>
<td>18.72</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0974</td>
<td><strong>1.13</strong></td>
<td>5.55</td>
<td>5.51</td>
<td>2.29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parrot</td>
<td>6</td>
<td>7.73</td>
<td>17.22</td>
<td>17.82</td>
<td><strong>17.90</strong></td>
<td>17.85</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0958</td>
<td><strong>1.38</strong></td>
<td>7.29</td>
<td>5.82</td>
<td>3.29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>7.84</td>
<td>18.26</td>
<td>18.57</td>
<td><strong>18.67</strong></td>
<td>18.61</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0980</td>
<td><strong>1.19</strong></td>
<td>8.57</td>
<td>3.65</td>
<td>1.86</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>12.49</td>
<td>19.36</td>
<td>19.67</td>
<td><strong>19.81</strong></td>
<td>19.76</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1007</td>
<td><strong>1.03</strong></td>
<td>10.98</td>
<td>3.10</td>
<td>1.22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>10.02</td>
<td>18.28</td>
<td>18.69</td>
<td><strong>18.79</strong></td>
<td>18.74</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0982</td>
<td><strong>1.20</strong></td>
<td>8.95</td>
<td>4.19</td>
<td>2.12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Syn</td>
<td>6</td>
<td>7.75</td>
<td>18.12</td>
<td>19.67</td>
<td><strong>19.75</strong></td>
<td>19.71</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0185</td>
<td><strong>1.29</strong></td>
<td>6.29</td>
<td>5.66</td>
<td>3.38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>9.85</td>
<td>19.58</td>
<td>20.56</td>
<td><strong>20.64</strong></td>
<td>20.58</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0978</td>
<td><strong>1.24</strong></td>
<td>4.71</td>
<td>5.14</td>
<td>2.71</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>12.61</td>
<td>21.44</td>
<td><strong>22.17</strong></td>
<td>22.08</td>
<td>22.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1005</td>
<td><strong>1.38</strong></td>
<td>4.86</td>
<td>3.96</td>
<td>2.21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>10.07</td>
<td>19.71</td>
<td>20.80</td>
<td><strong>20.82</strong></td>
<td>20.77</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1023</td>
<td><strong>1.30</strong></td>
<td>5.29</td>
<td>4.92</td>
<td>2.77</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The above advantages are also reflected in the comparison with different convex models ZWN and DZ. Related results are shown in Table 4, Figs. 11 and 12. More specifically, the SNR values of NEMA and NEMA$f$ in Table 4 are the highest in all cases except to comparison to DZ when $L = 20$ in Syn and the CPU time of NEMA$f$ is much fewer than DZ and NEMA. In addition, the speed of SNR changes in Figs. 12(a)-(c) are more faster with higher value and the successive error changes with respect to iteration number in Figs. 12(d)-(f) are also comparable.

### 4.3. Impact by the parameters

In this section, numerical results are given to illustrate how algorithm parameters influence the capability of proposed algorithms by fixing the model parameter $\lambda$. The idea is to vary one of the algorithm parameter, while keeping other parameters
Figure 8: Result of different methods for multiplicative noise and blur removal. Column (a): noisy and blurry image; column (b) to (e): restored image with different methods. Row 1 to 3: noise level (row 1: $L = 6$; row 2: $L = 10$; row 3: $L = 20$). The test image is Parrot (Fig. 1(b)).

Figure 9: (a)-(c): histories of SNR changes v.s. CPU time (in seconds) of proposed algorithms and AA, AA-DCA for multiplicative noise and blur removal. (d)-(f): $SE(\frac{1}{\|u_{k+1}-u_{k}\|})$ (in log scale) changes v.s. iteration number of proposed algorithms and AA, AA-DCA for multiplicative noise and blur removal. Column 1 to 3: noise level (column 1: $L = 6$, column 2: $L = 10$, column 3: $L = 20$). The test image is Syn (Fig. 1(c)).
New Splitting Algorithms for Multiplicative Noise Removal

Figure 10: (a)-(c): total energy of optimization problem \( E_{OP} := E(u) \) changes v.s. iteration number of different methods for multiplicative noise and blur removal. (d)-(f): total energy of Augmented Lagrangian \( E_{AL} := L_{\alpha,\eta}(u, \eta; \Theta_{\eta}) \) of NEMA changes v.s. iteration number of different methods for multiplicative noise and blur removal. (g)-(i): total energy of Augmented Lagrangian \( E_{AL} := L_{\beta,\eta}(u, \eta, p; \Lambda_{\eta}, \Lambda_{p}) \) of NEMA\(_f\) changes v.s. iteration number of different methods for multiplicative noise and blur removal. Column 1 to 3: noise level (column 1: \( L = 6 \); column 2: \( L = 10 \); column 3: \( L = 20 \)). The test image is Syn (Fig. 1(c)).

unchanged. Figs. 13 and 14 show the performance of NEMA and NEMA\(_f\). Figs. 13(a)-(d) are in denoising case, while Figs. 14(a)-(d) further consider Gaussian blur. In Fig. 13(a), \( \alpha_{\eta} \) varies from 13 to 300 with interval 20 and in Fig. 13(b), \( \alpha_{p} \) varies from 6 to 300 with same interval. In Fig. 13(c), \( \beta_{\eta} \) varies from 18 to 300 with interval 20 and \( \beta_{p} \) varies from 9 to 150 with interval 10 in Fig. 13(d). In Fig. 14(a) \( \alpha_{\eta} \) valued from 10 to 300 with step size 20, and \( \alpha_{p} \) valued from 1 to 300 with same step in Fig. 14(b). In Fig. 14(c), \( \beta_{\eta} \) valued from 17 to 300 with step size 20 and in Fig. 14(d), \( \beta_{p} \) valued from 3 to 150 with step size 10. All results above demonstrate that NEMA and NEMA\(_f\) are quite robust.

Furthermore in Fig. 15, we also show how model parameters affect the results of image restoration. Empirically, we set \( \lambda \) varies in \([0, 2]\) with interval 0.2 in the denoising
Figure 11: Result of different methods for multiplicative noise and blur removal. Column (a): noisy and blurry image; column (b) to (e): restored image with different methods. Row 1 to 3: noise level (row 1: $L = 6$; row 2: $L = 10$; row 3: $L = 20$). The test image is Cameraman (Fig. 1(a)).

Figure 12: (a)-(c): histories of SNR changes v.s. CPU time (in seconds) of proposed algorithms and ZWN, DZ for multiplicative noise and blur removal. (d)-(f): $SE\left(\frac{u^{k+1} - u^k}{\|u^k\|}\right)$ (in log scale) changes v.s. iteration number of proposed algorithms and ZWN, DZ for multiplicative noise and blur removal. Column 1 to 3: noise level (column 1: $L = 6$, column 2: $L = 10$, column 3: $L = 20$). The test image is Syn (Fig. 1(c)).
and deblurring case, while in the denoising case, set to $[0, 1]$ with an interval of 0.1.
The results show that model parameter is critical to the image restoration effect, and at present, we need to manually fine-tune to achieve the optimal result. Therefore, we will try to explore some automatic parameter adjustment methods in the future.

5. Conclusion

In this paper, we have proposed two algorithms, which are specially designed to solve the original AA model for multiplicative noise and blur removal. Under the noise estimate constraint, all subproblems of the proposed algorithms can be solved effectively. We have conducted numerical experiments in order to evaluate the performance of the proposed algorithms. Numerical results demonstrate the convergence of the proposed algorithms. However, due to the non-differentiability of the TV term and lack of the lower boundedness of the original variable, the current convergence theory cannot be directly applied to such algorithms. Hence we will leave the convergence analysis as a future work.

Acknowledgments

The authors would like to thank the editor and the three referees for their valuable comments. The work of YG and HC was partially supported by the NSFC (Grants
Figure 14: (a)-(b): the performance of NEMA with regard to $\alpha_{\eta}$, $\alpha_{p}$ for multiplicative noise and blur removal. (c)-(d): the performance of NEMA with regard to $\beta_{\eta}$, $\beta_{p}$ for multiplicative noise and blur removal. The test image is Syn (Fig. 1(c)) and $L = 6$.

Figure 15: (a)-(b): the performance of proposed algorithms ((a): NEMA, (b): NEMA$_{f}$) with regard to $\lambda$ for multiplicative noise removal. (c)-(d): the performance of proposed algorithms ((c) NEMA, (d): NEMA$_{f}$) with regard to $\lambda$ for multiplicative noise and blur removal. The test image is Syn (Fig. 1(a)) and $L = 6$. 
11871372, 11501413) and by the Natural Science Foundation of Tianjin (Grant 18JCY-BJC16600).

References


