1. Introduction

Digital images generated by various digital devices or used in practical applications usually contain some noises, so denoising is necessary before analyzing the images [1]. Over the last decade, a number of image denoising techniques preserving important image information such as edges have been proposed to obtain the true image from the Noisy (known) image. These methods include filtering technique [2], wavelet techniques [3, 4], and variational methods [5–9].

Denoising based on linear filters normally does not perform satisfactorily since both noise and edges contain high frequencies, while denoising based on the nonlinear models has been found to be successful. To preserve image edges, Rudin and Osher [8] and Rudin et al. [9] introduced a TV- (Total Variation-) based denoising model, which is based on a variational problem using the TV norm as a nonlinear nondifferentiable functional. Also Rudin et al. [9] used a nonlinear parabolic PDE (Partial Differential Equation) as a solution procedure for solving the Euler-Lagrange equation arising from the TV-based denoising model and proposed TM (Time Marching) method for solving the nonlinear parabolic PDE. This model is found to be a successful tool for image denoising and edge enhancement. High-order denoising models [10, 11] improving the TV denoising model have been proposed, but Euler-Lagrange equation associated with the TV functional for those models is a nonlinear PDE [12]. Due to the difficulties in obtaining analytic solutions of the nonlinear PDE, there has been many researches in developing implicit or explicit methods for the nonlinear PDE. Also, many good methods for image denoising problems such as high-order TV method [13, 14], total generalized variation (TGV) method, block-matching and 3D filtering (BM3D) method [15, 16], four-directional TV-based method [17], and an adaptive method based on Tikhonov and TV regularization [18] have been recently proposed.

For the last 15 years, HPM (homotopy perturbation method), first introduced by He [19], has been used by many mathematicians and engineers to solve various functional equations. In this method, the solution is considered as the sum of an infinite series which converges rapidly to the exact solutions. Using homotopy technique in topology, a homotopy is constructed with an embedding parameter \( p \in [0, 1] \) which is considered as a "small parameter." Many researches have been conducted in applying the HPM to a class of linear and nonlinear equations. In addition, HPM was further applied to nonlinear oscillators with discontinuities [20], nonlinear wave equations [21], boundary value problem
[22], and many other subjects [23–26]. Recently, Ghanbari et al. [27] proposed RHAM (restarted homotopy analysis method), based on HAM [28], for a nonlinear denoising model problem, which is a main motivation for this paper. The purpose of this paper is to propose a restarted homotopy perturbation method (RHPM), based on the HPM, for solving a nonlinear PDE problem and then to apply RHPM to the TV-based image denoising problem.

This paper is organized as follows. In Section 2, we review TM method for TV-based image denoising problem. In Section 3, we propose a restarted homotopy perturbation method (RHPM) for solving a nonlinear PDE problem. In Section 4, we present an application of RHPM to TV-based image denoising problem. The main difficulty in applying RHPM is settled by using binomial series techniques. In Section 5, numerical experiments for several test images are carried out to demonstrate the feasibility, efficiency, and reliability of RHPM by comparing the performance of RHPM with that of existing TM and recently proposed RHAM method. Lastly, some conclusions are drawn.

2. Review of TM Method for TV-Based Image Denoising Problem

In this section, we only review TM method for solving a TV-based image denoising problem (see [27] for details of the recently proposed RHAM method). TM method has been proven to be a successful tool for image denoising and has been used extensively in many practical applications.

Let \( \Omega \) be a bounded domain, let an intensity function \( z(x, y) \) be the pixel values of an observed image, and let \( \eta(x, y) \) be an unknown noise function. Then, we would like to construct the desired clean image \( u(x, y) \) from the observed image \( z(x, y) \) such that

\[
z(x, y) = u(x, y) + \eta(x, y).
\]

Image denoising problem is to reconstruct \( u(x, y) \) from \( z(x, y) \) which contains the additive noise \( \eta(x, y) \). It is well known that image denoising problem is an ill-conditioned problem, so regularization techniques should be used to approximate \( u(x, y) \) from \( z(x, y) \).

Rudin et al. [9] proposed the use of the following TV-based minimization problem:

\[
\min_u \int_{\Omega} |\nabla u(x, y)|\, dx\, dy + \frac{\lambda}{2} \int_{\Omega} (u(x, y) - z(x, y))^2\, dx\, dy, \tag{2}
\]

where \( \lambda > 0 \) is a regularization parameter. The first term is the TV of \( u(x, y) \), a regularizer, while the second term is a fidelity term ensuring that the denoised image \( u(x, y) \) will be close to the given image \( z(x, y) \). The regularization parameter \( \lambda \) is important for balancing denoising and smoothing. That is, it can be observed that the parameter \( \lambda \) may be small in the presence of high noise and \( \lambda \) may be large for little noise. The Euler-Lagrange equation corresponding to (2) leads to a nonlinear elliptic PDE with homogeneous Neumann boundary conditions as follows:

\[
g(u(x, y)) = \nabla \cdot \left( \frac{\nabla u(x, y)}{\|\nabla u(x, y)\|} \right) - \lambda (u(x, y) - z(x, y)) = 0, \quad (x, y) \in \Omega,
\]

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

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

To avoid division by zero in numerical implementation, we replace the nondifferentiable term \( |\nabla u(x, y)| \) by a smooth approximation as follows:

\[
|\nabla u(x, y)| = \sqrt{|\nabla u(x, y)|^2 + \beta^2}
\]

for some small \( \beta > 0 \).

Hence, one obtains

\[
g(u(x, y)) = \nabla \cdot \left( \frac{\nabla u(x, y)}{\|\nabla u(x, y)\|} \right) - \lambda (u(x, y) - z(x, y)) = 0, \quad (x, y) \in \Omega,
\]

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

In order to solve the Euler-Lagrange equation (5), Rudin et al. [9] proposed TM method for solving the following time-dependent nonlinear parabolic PDE

\[
\frac{\partial u(x, y; t)}{\partial t} = \nabla \cdot \left( \frac{\nabla u(x, y; t)}{\|\nabla u(x, y; t)\|} \right) - \lambda (u(x, y) - z(x, y)) , \quad t > 0, \quad (x, y) \in \Omega,
\]

\[
\nabla u \cdot \vec{n} = 0, \quad (x, y) \in \partial \Omega, \quad \text{where } u(x, y, 0) \text{ is given.}
\]

For numerical implementation, let us assume that the domain \( \Omega \) has been split into \( n \times m \) cells, where the grid points are located at \( (x_i = ih_x, y_j = jh_y), \) \( 1 \leq i \leq m, 1 \leq j \leq n, t_k = k\Delta t, \) where \( \Delta t \) 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 \( u^{k}_{ij} \) and \( u^{k}_{ij} = z(x_i, y_j). \)

Without loss of generality, we can assume that \( h_x = h_y = 1. \) Then, the nonlinear PDE (6) can be approximated by the following finite difference formula:

\[
u^{k+1}_{ij} = u^{k}_{ij} + \Delta t \cdot \left( \kappa (u^{k}_{ij}) - \lambda \left( u^{k}_{ij} - z_{ij} \right) \right), \tag{7}
\]
where
\[
\kappa(u_{ij}^k) = \left( \nabla \cdot \left( \frac{\nabla u}{|\nabla u|_\beta} \right) \right)_{ij}^k = \left[ \frac{\partial}{\partial x} \left( \frac{u_x}{|\nabla u|_\beta} \right) \right]_{ij}^k
\]
+ \frac{\partial}{\partial y} \left( \frac{u_y}{|\nabla u|_\beta} \right)_{ij}^k
\]
+ \frac{\partial}{\partial z} \left( \frac{u_z}{|\nabla u|_\beta} \right)_{ij}^k
\]
(8)

with
\[
\Delta^x u_{ij}^k = u_{i+1,j}^k - u_{ij}^k,
\]
\[
\Delta^y u_{ij}^k = -u_{i,j+1}^k + u_{ij}^k,
\]
\[
\Delta^z u_{ij}^k = u_{i+1,j+1}^k - u_{ij}^k,
\]
\[
\Delta^x u_{ij}^k = u_{i,j+1}^k - u_{ij}^k,
\]
and boundary conditions are
\[
u_{ij}^k = u_{ij}^k,
\]
\[
u_{i+1,j}^k = u_{ij}^k,
\]
\[
u_{i,j}^k = u_{ij}^k,
\]
\[
u_{i,j+1}^k = u_{ij}^k,
\]
for \(1 \leq i \leq m, 1 \leq j \leq n\).

Hence, we obtain TM method, called Algorithm 1, for solving (6), where tol is the tolerance for stopping criterion, maxit is the maximum number of iterations, z is the Noisy image, v is the true image, and PSNR is the peak signal-to-noise ratio between the true and restored images which is defined in Section 5.

### 3. Description of Restarted Homotopy Perturbation Method (RHPM)

In this section, we first describe the homotopy perturbation method (HPM) for solving nonlinear Partial Differential Equation

\[
A(u) - f(r) = 0, \quad r \in \Omega
\]
(11)
with the boundary conditions
\[
B\left( u, \frac{\partial u}{\partial n} \right) = 0, \quad r \in \Gamma,
\]
(12)

where \(A\) is a general differential operator, \(B\) is a boundary operator, \(f(r)\) is a known analytic functions, and \(\Gamma\) is the boundary of the domain \(\Omega\). The operator \(A\) can be divided in two parts \(L\) and \(N\), where \(L\) is linear and \(N\) is nonlinear. Therefore, (11) can be written as

\[
L(u) + N(u) - f(r) = 0.
\]
(13)

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

\[
H(v, p) = (1 - p) [L(v) - L(u_0)] + p [A(v) - f(r)]
\]
(14)
or, equivalently,

\[
H(v, p) = L(v) - L(u_0) + pL(u_0)
\]
\[
+ p [N(v) - f(r)] = 0,
\]
(15)
where \(p \in [0, 1]\) is an embedding parameter and \(u_0\) is the initial approximation of (11) which satisfies the boundary conditions. Clearly, we have

\[
H(v, 0) = L(v) - L(u_0) = 0,
\]
\[
H(v, 1) = A(v) - f(r) = 0.
\]
(16)

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 \(0 \leq p \leq 1\) is considered as a “small parameter” applying the classical perturbation technique, then the solution of (15) can be expressed as a power series in \(p\); that is,

\[
v = v(r, p) = v_0 + p v_1 + p^2 v_2 + \cdots.
\]
(17)

Setting \(p \to 1\), the solution \(u\) of (11) can be expressed as the sum of an infinite series

\[
u = \lim_{p \to 1} v(r, p) = v_0 + v_1 + v_2 + \cdots.
\]
(18)

Further description and convergence for the HPM can be found in [19–22, 26].

Before giving a general description of the restarted homotopy perturbation method (RHPM), we first provide a simple example of how RHPM can be applied to a PDE problem.

**Example 1.** Consider two-dimensional heat equation with variable coefficients of the form

\[
u_t - \frac{y^2}{2}u_{xx} - \frac{x^2}{2}u_{yy} = 0
\]
(19)

with the initial condition \(u(x, y, 0) = y^2\).

**Method 1 (HPM).** We first solve (19) using the traditional HPM. The operators \(L\), \(N\), and \(f\) in (13) can be viewed as

\[
L(u) = u_t,
\]
\[
N(u) = -\frac{y^2}{2}u_{xx} - \frac{x^2}{2}u_{yy}
\]
(20)
\[
f(r) = 0.
\]
Let \( u_0 = u(x, y, 0) \). From (15), we readily obtain a homotopy \( v(r, p) \) which satisfies

\[
H(v, p) = v_t - (u_0)_t + p(u_0)_t + p^2[(v_0)_xx + (v_1)_xx + \cdots] + \frac{x^2}{2}((v_0)_yy + (v_1)_yy + \cdots) = (u_0)_t.
\]

Then, the solution \( v \) of (21) can be expressed as in (17). Substituting (17) into (21), one obtains

\[
(v_0)_t + p(v_1)_t + p^2(v_2)_t + \cdots = (u_0)_t - p(u_0)_t + p^2\left[\frac{y^2}{2}((v_0)_xx + (v_1)_xx + \cdots) + \frac{x^2}{2}((v_0)_yy + (v_1)_yy + \cdots)\right] = (u_0)_t.
\]

Comparing coefficients of terms with identical powers of \( p \), we obtain that

\[
p^0: (v_0)_t = (u_0)_t \Rightarrow v_0 = u(x, y, 0) = y^2
\]

\[
p^1: (v_1)_t = -(u_0)_t + \frac{y^2}{2}((v_0)_xx + (v_0)_yy) \Rightarrow v_1 = \int_0^t \left[-(v_0)_t + \frac{1}{2}\left(y^2((v_0)_xx + x^2(v_0)_yy)\right)\right] dt
\]

\[
p^{2n}: (v_{2n})_t = \frac{y^2}{2}(v_{2n-1})_{xx} + \frac{x^2}{2}(v_{2n-1})_{yy} (n \geq 1) \Rightarrow v_{2n} = \int_0^t \left[\frac{y^2}{2}(v_{2n-1})_{xx} + \frac{x^2}{2}(v_{2n-1})_{yy}\right] dt
\]

\[
p^{2n+1}: (v_{2n+1})_t = \frac{y^2}{2}(v_{2n})_{xx} + \frac{x^2}{2}(v_{2n})_{yy} (n \geq 1) \Rightarrow v_{2n+1} = \int_0^t \left[\frac{y^2}{2}(v_{2n})_{xx} + \frac{x^2}{2}(v_{2n})_{yy}\right] dt.
\]

Hence, we have

\[
v_0 = y^2,
\]

\[
v_1 = \int_0^t x^2 dt = tx^2
\]

\[
v_{2n} = \int_0^t \frac{t^{2n-1}}{(2n-1)!} y^2 dt = \frac{t^{2n}}{(2n)!} y^2 (n \geq 1)
\]

\[
v_{2n+1} = \int_0^t \frac{t^{2n+1}}{(2n+1)!} x^2 dt = \frac{t^{2n+1}}{(2n+1)!} x^2 (n \geq 1).
\]

Therefore, we obtain the homotopy solution of (21)

\[
v = y^2 + tx^2 p + \frac{t^2}{2} y^2 p^2 + \frac{t^3}{3!} x^2 p^3 + \frac{t^4}{4!} y^2 p^4 + \frac{t^5}{5!} x^2 p^5 + \cdots
\]

\[
= \sum_{n=0}^{\infty} \frac{t^{2n} p^{2n}}{(2n)!} y^2 + \frac{t^{2n+1} p^{2n+1}}{(2n+1)!} x^2.
\]

Letting \( p \to 1 \), we obtain the exact solution of (19) which is given by

\[
u = \lim_{p \to 1} v = \sum_{n=0}^{\infty} \frac{t^{2n} p^{2n}}{(2n)!} y^2 + \frac{t^{2n+1} p^{2n+1}}{(2n+1)!} x^2
\]

\[
y = y^2 \cosh t + x^2 \sinh t.
\]

Method 2 (RHPM). We propose a new approach for solving (19), called RHPM, which is based on HPM. We choose \( v_0 = u(x, y, 0) = y^2 \) as an initial approximation of (19). Let the operators \( L, N, \) and \( f \) be defined as in (20). Comparing coefficients of terms \( p^0 \) and \( p^1 \) in (22), one obtains

\[
p^0: (v_0)_t = (u_0)_t \Rightarrow v_0 = u_0 = u(x, y, 0) = y^2
\]

\[
p^1: (v_1)_t = -(u_0)_t + \frac{y^2}{2}(v_0)_xx + \frac{x^2}{2}(v_0)_yy \Rightarrow v_1 = \int_0^t \left[-(v_0)_t + \frac{1}{2}\left(y^2((v_0)_xx + x^2(v_0)_yy)\right)\right] dt
\]

\[
p^{2n}: (v_{2n})_t = \frac{y^2}{2}(v_{2n-1})_{xx} + \frac{x^2}{2}(v_{2n-1})_{yy} (n \geq 1) \Rightarrow v_{2n} = \int_0^t \left[\frac{y^2}{2}(v_{2n-1})_{xx} + \frac{x^2}{2}(v_{2n-1})_{yy}\right] dt
\]

\[
p^{2n+1}: (v_{2n+1})_t = \frac{y^2}{2}(v_{2n})_{xx} + \frac{x^2}{2}(v_{2n})_{yy} (n \geq 1) \Rightarrow v_{2n+1} = \int_0^t \left[\frac{y^2}{2}(v_{2n})_{xx} + \frac{x^2}{2}(v_{2n})_{yy}\right] dt.
\]
Then, the first two term approximation of (18) is defined as follows:
\[ u_0^1 = v_0 + v_1 = y^2 + tx^2. \] (28)

From (27), using \( u_0 = u_0^1 \) as an initial approximation, one easily obtains
\[ p^0: v_0^1 = u_0^1 = y^2 + tx^2 \]
\[ p^1: v_1^1 = \int_0^t \left[ -(v_0^1)_t + \frac{1}{2} \left\{ y^2 (v_0^1)_{xx} + x^2 (v_0^1)_{yy} \right\} \right] dt \]
\[ = \frac{t^2}{2} y^2. \] (29)

From (29), the new approximation \( u_0^2 \) is given by
\[ u_0^2 = v_0^1 + v_1^1 = y^2 + tx^2 + \frac{t^2}{2} y^2. \] (30)

From (27) with an initial approximation \( u_0 = u_0^2 \), one obtains
\[ p^0: v_0^2 = u_0^2 = y^2 + tx^2 + \frac{t^2}{2} y^2 \]
\[ p^1: v_1^2 = \int_0^t \left[ -(v_0^2)_t + \frac{1}{2} \left\{ y^2 (v_0^2)_{xx} + x^2 (v_0^2)_{yy} \right\} \right] dt \]
\[ = \frac{t^3}{3!} x^2. \] (31)

Hence, the new approximation \( u_0^3 \) is given by
\[ u_0^3 = v_0^2 + v_1^2 = y^2 + tx^2 + \frac{t^2}{2} y^2 + \frac{t^3}{3!} x^2. \] (32)

Continuing this process, it can be seen that \( u_k \) converges to the exact solution \( u \) of (19), which is the same as the result of Method 1, as \( k \rightarrow \infty \).

Generally speaking, the computation of \( v_k \) in the HPM becomes more complicated as \( k \) increases. To circumvent this problem, RHPM 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 RHPM as compared with HPM. From the idea of RHPM introduced in Example 1, the general procedure of RHPM for solving a nonlinear equation is described below.

Let \( u_0^0 \) be an initial approximation of a nonlinear equation. First, we compute the first \( k \)th term approximation \( u_0^k \) in the following way:
\[ v_0^0 = u_0^0 \]
Compute \( v_1^0, v_2^0, \ldots, v_k^0 \) using (15) and (17) (33)
\[ u_0^k = v_0^0 + v_1^0 + v_2^0 + \cdots + v_k^0. \]

Following the above computational process with \( u_0^0 \) as an initial approximation, a new approximation \( u_0^1 \) is computed as follows:
\[ v_0^1 = u_0^1 \]
Compute \( v_1^1, v_2^1, \ldots, v_k^1 \) using (15) and (17) (34)
\[ u_0^{k+1} = v_0^1 + v_1^1 + v_2^1 + \cdots + v_k^1. \]

By repeating the above process, a new approximation \( u_0^{k+1} \) can be computed from the initial approximation \( u_0^0 \), which was obtained at the \( k \)th step, in the following way:
\[ v_0^k = u_0^k \]
Compute \( v_1^k, v_2^k, \ldots, v_k^k \) using (15) and (17) (35)
\[ u_0^{k+1} = v_0^k + v_1^k + v_2^k + \cdots + v_k^k. \]

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

4. Application of RHPM to TV-Based Image Denoising Problem

In this section, we describe an application of RHPM to a TV-based image denoising problem. Before doing this, we first describe the binomial series which is used for an application of RHPM. The difficulty in applying RHPM to the TV-based image denoising problem comes from (45) (i.e., \( (v_x^2 + v_y^2)^{-1/2} \) and \( (v_x^2 + v_y^2)^{-3/2} \)). In order to handle this difficulty, the following binomial series is used. For \( |x| < 1 \),
\[ (1 + x)^n = \sum_{k=0}^{\infty} \binom{n}{k} x^k \]
\[ = 1 + nx + \frac{n(n-1)}{2!} x^2 + \frac{n(n-1)(n-2)}{3!} x^3 + \cdots. \] (36)

If \( n = -1/2 \) in (36), then
\[ (1 + x)^{-1/2} = \frac{1}{\sqrt{1 + x}} = 1 - \frac{1}{2} x + \frac{3}{8} x^2 - \frac{5}{16} x^3 + \cdots. \] (37)

Letting \( x = \tau - 1 \) in (37), for \( 0 < \tau < 2 \),
\[ \frac{1}{\sqrt{\tau}} = 1 - \frac{1}{2} (\tau - 1) + \frac{3}{8} (\tau - 1)^2 - \frac{5}{16} (\tau - 1)^3 + \cdots \]
\[ = \frac{35}{16} - \frac{21}{16} \tau + \frac{5}{16} \tau^2 - \frac{5}{16} \tau^3 + \cdots. \] (38)
If $\tau > 0$ is large, then, by introducing a small parameter $\varepsilon > 0$ and using (38), $1/\sqrt{\tau}$ can be approximated by

$$\frac{1}{\sqrt{\tau}} \approx \sqrt{\varepsilon} \left( \frac{35}{16} \frac{\varepsilon}{\tau} + \frac{21 \varepsilon^2}{16} \frac{1}{\tau^2} - \frac{5 \varepsilon^3}{16} \frac{1}{\tau^3} \right).$$ (39)

Similarly, if $n = -3/2$ and $x = \tau - 1$ in (36), then, for $|x| < 1$ or $0 < \tau < 2$,

$$\frac{1}{\sqrt{(1 + x)^3}} = (1 + x)^{3/2},$$ (40)

$$\frac{1}{\sqrt{\tau^3}} = \sqrt{\varepsilon} \left( \frac{95}{16} \frac{\varepsilon}{\tau} + \frac{179 \varepsilon^2}{16} \frac{1}{\tau^2} - \frac{35 \varepsilon^3}{16} \frac{1}{\tau^3} + \cdots \right).$$ (41)

If $\tau > 0$ is large, then, by introducing a small parameter $\varepsilon > 0$ and using (40), $1/\sqrt{\tau^3}$ can be approximated by

$$\frac{1}{\sqrt{\tau^3}} = \sqrt{\varepsilon} \left( \frac{95}{16} \frac{\varepsilon}{\tau} + \frac{179 \varepsilon^2}{16} \frac{1}{\tau^2} - \frac{35 \varepsilon^3}{16} \frac{1}{\tau^3} + \cdots \right).$$ (41)

Note that a small parameter $\varepsilon > 0$ is used to guarantee the convergence of the binomial series when $\tau > 0$ is large.

Now, we describe how to apply RHDM to the TV-based image denoising problem (6). For simplicity of exposition, we describe computational process for RHDM(2) (i.e., $k = 2$). Since we only use four terms of the binomial series for numerical computation of (45), division by zero does not occur and thus $\beta = 0$ is used for RHDM (see (39) and (41)). Let the operators $L, N$, and $f$ be defined as follows:

$$L(v) = \frac{\partial v(x, y; t)}{\partial t},$$

$$N(v) = - \nabla \cdot \left( \frac{\nabla v(x, y; t)}{|\nabla v(x, y; t)|} \right),$$

$$f(r) = \lambda \left\{ v(x, y; t) - z(x, y) \right\},$$ (42)

where

$$\nabla v(x, y; t) = \left( \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y} \right)^T,$$

$$|\nabla v(x, y; t)| = \sqrt{(v_x)^2 + (v_y)^2}.$$ (43)

From (15), one obtains

$$L(v) = L(u_0) - p L(u_0) - p \left[ N(v) - f(r) \right] = \frac{\partial u_0}{\partial t}$$

$$- p \frac{\partial u_0}{\partial t} - p \left[ -\nabla \cdot \left( \frac{\nabla v(x, y; t)}{|\nabla v(x, y; t)|} \right) \right]$$

$$- \lambda \left\{ v(x, y; t) - z(x, y) \right\} = \frac{\partial u_0}{\partial t} - \frac{\partial u_0}{\partial t} + p \nabla$$

$$\cdot \left( \frac{\nabla v(x, y; t)}{|\nabla v(x, y; t)|} \right) + p \lambda \left\{ v(x, y; t) - z(x, y) \right\},$$ (44)

where

$$\nabla \cdot \left( \frac{\nabla v}{|\nabla v|} \right) = \frac{v_{xx} + v_{yy}}{\sqrt{(v_x)^2 + (v_y)^2}}$$

$$- \frac{(v_x)^2 v_{xx} + (v_y)^2 v_{yy}}{\sqrt{((v_x)^2 + (v_y)^2)^3}}.$$ (45)

Since the solution $v$ of (44) is of the form (17), by simple computation

$$v_x = (v_0)_x + p (v_1)_x + p^2 (v_2)_x + \cdots,$$

$$v_{xx} = (v_0)_{xx} + p (v_1)_{xx} + p^2 (v_2)_{xx} + \cdots,$$

$$v_y = (v_0)_y + p (v_1)_y + p^2 (v_2)_y + \cdots,$$

$$v_{yy} = (v_0)_{yy} + p (v_1)_{yy} + p^2 (v_2)_{yy} + \cdots,$$ (46)

$$(v_x)^2 = (v_0)^2 + 2 [(v_0)_x (v_1)_x] p + \cdots,$$

$$(v_y)^2 = (v_0)^2 + 2 [(v_0)_y (v_1)_y] p + \cdots.$$ From these equations, one easily obtains

$$(v_x)^2 + (v_y)^2 = \left[ (v_0)^2 + (v_0)_x \right]$$

$$+ 2 [(v_0)_x (v_1)_x + (v_0)_y (v_1)_y] p + \cdots$$

$$= (D_1) + 2 (D_2) p + \cdots,$$

$$(v_x)^2 + (v_y)^2 = (D_1)^2 + 4 (D_1) (D_2) p + \cdots,$$ (47)

$$(v_x)^2 + (v_y)^2 = (D_1)^3 + 6 (D_1)^2 (D_2) p + \cdots,$$

$$v_{xx} + v_{yy} = \left[ (v_0)_{xx} + (v_0)_{yy} \right]$$

$$+ \left[ (v_1)_{xx} + (v_1)_{yy} \right] p + \cdots$$

$$= (D_3) + (D_4) p + \cdots,$$

$$(v_x)^2 v_{xx} + (v_y)^2 v_{yy} = (D_5) + (D_6) p + \cdots.$$
where
\[
(D_1) = (v_0)_x^2 + (v_0)_y^2,
\]
\[
(D_2) = (v_0)_x (v_1)_x + (v_0)_y (v_1)_y,
\]
\[
(D_3) = (v_1)_x (v_0)_x + (v_1)_y (v_0)_y,
\]
\[
(D_4) = (v_0)_x x + (v_0)_y y,
\]
\[
(D_5) = (v_0)_x x + (v_0)_y y,
\]
\[
(D_6) = (v_0)_x x + (v_0)_y y,
\]
\[
\text{(48)}
\]

The \( L(v) \) and \( p\lambda(v - z) \) in (44) can be expressed as
\[
L(v) = \frac{\partial v}{\partial t} = \frac{\partial v_0}{\partial t} + p \frac{\partial v_1}{\partial t} + p^2 \frac{\partial v_2}{\partial t} + \cdots,
\]
\[
p\lambda(v - z) = \lambda p (v_0 - z) + \lambda p^2 v_1 + \lambda p^3 v_2 + \cdots.
\]

Substituting \( \tau = (v_x)^2 + (v_y)^2 \) into (39), one obtains
\[
\frac{1}{|V|^3} = \frac{\sqrt{\varepsilon}}{\sqrt{\varepsilon^2 + \varepsilon^2}} = \sqrt{\frac{35}{16}} + \frac{35e}{16} \left\{ (v_x)^2 + (v_y)^2 \right\} + \frac{5e^3}{16} \left\{ (v_x)^2 + (v_y)^2 \right\} + \frac{21e^2}{16} \left\{ (v_x)^2 + (v_y)^2 \right\} + \cdots
\]
\[
\text{(50)}
\]

Hence, the first term of the right hand side in (45) can be written as
\[
\frac{v_{xx} + v_{yy}}{|V|^3} = (A_1) (D_3) + (A_2) (D_4) p + \cdots.
\]

Substituting \( \tau = (v_x)^2 + (v_y)^2 \) into (41), one obtains
\[
(A_1) = \sqrt{\frac{35}{16}} \left\{ \frac{35e}{16} (D_1) + \frac{21e^2}{16} (D_2)^2 - \frac{5e^3}{16} (D_3)^3 \right\}
\]
\[
\text{(51)}
\]
\[
(B_1) = \varepsilon \sqrt{\frac{35}{16}} \left\{ -\frac{35e}{16} (D_1) - \frac{15e^2}{8} (D_2)^2 \right\}.
\]

Hence, the second term of the right hand side in (45) can be written as
\[
\frac{(v_x)^2 v_{xx} + (v_y)^2 v_{yy}}{|V|^3} = (A_2) (D_3) + (B_2) (D_3) p + \cdots.
\]

\[
\text{(55)}
\]
From (45), (52), and (55), (44) can be expressed as

\[
\frac{\partial v_0}{\partial t} + p \frac{\partial v_1}{\partial t} + p^2 \frac{\partial v_2}{\partial t} + \cdots = \frac{\partial u_0}{\partial t} - p \frac{\partial u_0}{\partial t} + p \left( \frac{v_{xx} + v_{yy}}{|\nabla v|} - \frac{(v_x)^2 v_{xx} + (v_y)^2 v_{yy}}{|\nabla v|^3} \right) + p \lambda (v_0 - z) + p^2 \lambda v_1 + \cdots
\]

\[
= \frac{\partial u_0}{\partial t} - p \left( \frac{\partial u_0}{\partial t} - \lambda (v_0 - z) \right) + p^2 \lambda v_1 + \cdots + (A_1)(D_3) p - (A_2)(D_5) p + (A_1)(D_4) p^2 + (B_1)(D_2)(D_3) p^2 - (B_2)(D_2)(D_5) p^3 - (A_2)(D_6) p^3 + \cdots.
\]

Comparing the coefficients of \( p^0 \), \( p^1 \), and \( p^2 \) in (56),

\[
p^0: \frac{\partial v_0}{\partial t} = \frac{\partial u_0}{\partial t} \implies v_0 = u_0 \tag{57}
\]

\[
u_0 = z: \text{ given an initial approximation}
\]

\[
p^1: \frac{\partial v_1}{\partial t} = - \frac{\partial v_0}{\partial t} + \lambda (v_0 - z) + (A_1)(D_3)
\]

\[\quad - (A_2)(D_5) \tag{58}\]

\[
p^2: \frac{\partial v_2}{\partial t} = \lambda v_1 + (A_1)(D_4)
\]

\[\quad + [(B_1)(D_3) - (B_2)(D_5)](D_2)
\]

\[\quad - (A_2)(D_6). \tag{59}\]

Applying the inverse operator of \( L \) to (58), one obtains

\[
v_1 = -(v_0 - z) + \lambda (v_0 - z) t + (A_1)(D_3) t
\]

\[\quad - (A_2)(D_5) t = -F + (\lambda F + G) t, \tag{60}\]
where \( F = v_0 - z \) and \( G = (A_1)(D_2) - (A_2)(D_3) \). From (60), the partial derivatives of \( v_1 \) are

\[
\begin{align*}
(v_1)_x &= -F_x + t (\lambda F + G)_x = -F_x + (\lambda F_x + G_x) t \\
&= -F_x + (H_1) t, \\
(v_1)_y &= -F_y + t (\lambda F + G)_y = -F_y + (\lambda F_y + G_y) t \\
&= -F_y + (H_2) t, \\
(v_1)_{xx} &= -F_{xx} + t (\lambda F + G)_{xx} \\
&= -F_{xx} + (\lambda F_{xx} + G_{xx}) t = -F_{xx} + (H_3) t, \\
(v_1)_{yy} &= -F_{yy} + t (\lambda F + G)_{yy} \\
&= -F_{yy} + (\lambda F_{yy} + G_{yy}) t = -F_{yy} + (H_4) t,
\end{align*}
\]

(61)

where \( H_1 = \lambda F_x + G_x \), \( H_2 = \lambda F_y + G_y \), \( H_3 = \lambda F_{xx} + G_{xx} \), and \( H_4 = \lambda F_{yy} + G_{yy} \). From (61), \( D_2 \), \( D_4 \), and \( D_6 \) can be expressed as

\[
\begin{align*}
(D_2) &= (v_0)_x [-F_x + (H_1) t] + (v_0)_y [-F_y + (H_2) t], \\
(D_4) &= - (F_{xx} + F_{yy}) + [(H_3) + (H_4)] t, \\
(D_6) &= (v_0)_x^2 [-F_{xx} + (H_3) t] + (v_0)_y^2 [-F_{yy} + (H_4) t] \\
&\quad + 2 (v_0)_x (v_0)_y [F_{xy} + (H_3)_y + 2 (v_0)_y (v_0)_yy] \\
&\quad \cdot [F_{xy} + (H_4)_x] = - [(v_0)_x^2 F_{xx} + (v_0)_y^2 F_{yy} \\
&\quad + 2 (v_0)_x (v_0)_y F_{xy} + (v_0)_y (v_0)_yy] F_{yx} + [H_3)_x + (v_0)_y (H_4)_y] t \\
&\quad + 2 [(v_0)_x (v_0)_yy (H_1) + (v_0)_y (v_0)_xx (H_2)] t.
\end{align*}
\]
Substituting (60) and (62) into (59) and applying the inverse operator of $L$ to (59), one obtains

$$
\begin{align*}
V_2 &= - \left[ \lambda F + (A_1) \left( F_{xx} + F_{yy} \right) \\
& + \left\{ (B_1) (D_3) - (B_2) (D_3) \right\} \left\{ (v_0)_x F_x + (v_0)_y F_y \right\} \right] t \\
& + (A_2) \left\{ (v_0)_x^2 F_{xx} + (v_0)_y^2 F_{yy} \right\} \\
& + 2 \left\{ (v_0)_x (v_0)_y F_x + (v_0)_y (v_0)_y F_y \right\} t + \lambda (\lambda F) \\
& + G \right\{ \frac{1}{2} t^2 + (A_1) \left\{ (H_3) + (H_4) \right\} \frac{1}{2} t^2 + \left\{ (B_1) (D_3) \\
& - (B_2) (D_3) \right\} \left\{ (v_0)_x (H_1) + (v_0)_y (H_2) \right\} \frac{1}{2} t^2 - (A_2) \\
& \cdot \left\{ (v_0)_x^2 (H_3) \right\} \\
& + (v_0)_y^2 (H_4) + 2 (v_0)_x (v_0)_y (v_0)_xx (H_1) \\
& + 2 (v_0)_y (v_0)_yy (H_2) \right\} \frac{1}{2} t^2.
\end{align*}
$$

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 RHPM(2) method.

For numerical implementation of RHPM(2) method, let us assume that the domain $\Omega$ has been split into $m \times n$ cells, where the grid points are located at $(x_i = i h_x, y_j = j h_y)$, $1 \leq i \leq m, 1 \leq j \leq n$. Let $t_k = k \Delta t$, where $\Delta t$ 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 $(u^k)_{ij}$. Without loss of generality, we can assume that $h = h_x = h_y = 1$. For simplicity of exposition, we define the following notation:

$$
\begin{align*}
v_0^k &= \left\{ (v_0^k)_{ij} = v_0(x_i, y_j; t_k) \mid 1 \leq i \leq m, 1 \leq j \leq n, k \\
&= 1, 2, \ldots \right\}, \\
v_1^k &= \left\{ (v_1^k)_{ij} = v_1(x_i, y_j; t_k) \mid 1 \leq i \leq m, 1 \leq j \leq n, k \\
&= 1, 2, \ldots \right\}, \\
v_2^k &= \left\{ (v_2^k)_{ij} = v_2(x_i, y_j; t_k) \mid 1 \leq i \leq m, 1 \leq j \leq n, k \\
&= 1, 2, \ldots \right\}.
\end{align*}
$$
Figure 4: Example of the Mandrill image: Original image (a), Noisy image (b), TM denoising (c) RHAM2 denoising (d), RHPM(2) denoising (e), and RHPM(1) denoising (f).

Table 1: Numerical results for image denoising problems using TM, RHAM2, RHPM(2), and RHPM(1) methods with Δt = 0.15.

<table>
<thead>
<tr>
<th>Image</th>
<th>Noise</th>
<th>TM PSNR</th>
<th>Iter</th>
<th>CPU</th>
<th>RHAM2 PSNR</th>
<th>Iter</th>
<th>CPU</th>
<th>RHPM(2) PSNR</th>
<th>Iter</th>
<th>CPU</th>
<th>RHPM(1) PSNR</th>
<th>Iter</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cameraman</td>
<td>15%</td>
<td>30.24</td>
<td>134</td>
<td>6.58</td>
<td>30.27</td>
<td>88</td>
<td>17.03</td>
<td>30.14</td>
<td>27</td>
<td>3.58</td>
<td>30.33</td>
<td>22</td>
<td>1.36</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>28.60</td>
<td>188</td>
<td>8.80</td>
<td>28.64</td>
<td>185</td>
<td>35.42</td>
<td>28.69</td>
<td>27</td>
<td>3.54</td>
<td>28.78</td>
<td>22</td>
<td>1.33</td>
</tr>
<tr>
<td>Jet-plane</td>
<td>15%</td>
<td>26.79</td>
<td>203</td>
<td>9.85</td>
<td>26.80</td>
<td>219</td>
<td>42.65</td>
<td>27.08</td>
<td>27</td>
<td>3.62</td>
<td>27.16</td>
<td>22</td>
<td>1.36</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>25.22</td>
<td>287</td>
<td>13.42</td>
<td>25.31</td>
<td>652</td>
<td>123.81</td>
<td>25.54</td>
<td>27</td>
<td>3.55</td>
<td>25.52</td>
<td>22</td>
<td>1.32</td>
</tr>
<tr>
<td>Lake</td>
<td>15%</td>
<td>27.35</td>
<td>134</td>
<td>6.50</td>
<td>27.35</td>
<td>88</td>
<td>16.47</td>
<td>27.63</td>
<td>26</td>
<td>3.47</td>
<td>27.65</td>
<td>21</td>
<td>1.31</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>25.89</td>
<td>191</td>
<td>8.95</td>
<td>25.94</td>
<td>192</td>
<td>35.71</td>
<td>26.33</td>
<td>27</td>
<td>3.52</td>
<td>26.43</td>
<td>22</td>
<td>1.31</td>
</tr>
<tr>
<td>Mandrill</td>
<td>15%</td>
<td>26.18</td>
<td>104</td>
<td>5.08</td>
<td>26.17</td>
<td>56</td>
<td>10.74</td>
<td>26.70</td>
<td>25</td>
<td>3.34</td>
<td>26.70</td>
<td>21</td>
<td>1.31</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>24.62</td>
<td>153</td>
<td>7.31</td>
<td>24.61</td>
<td>113</td>
<td>20.97</td>
<td>25.41</td>
<td>26</td>
<td>3.47</td>
<td>25.38</td>
<td>21</td>
<td>1.26</td>
</tr>
<tr>
<td>Nodules</td>
<td>15%</td>
<td>28.20</td>
<td>195</td>
<td>5.16</td>
<td>28.22</td>
<td>199</td>
<td>19.81</td>
<td>28.69</td>
<td>27</td>
<td>1.97</td>
<td>28.67</td>
<td>22</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>26.60</td>
<td>271</td>
<td>7.03</td>
<td>26.66</td>
<td>549</td>
<td>54.14</td>
<td>26.79</td>
<td>28</td>
<td>2.01</td>
<td>26.73</td>
<td>22</td>
<td>0.74</td>
</tr>
<tr>
<td>Rice</td>
<td>15%</td>
<td>30.62</td>
<td>104</td>
<td>1.34</td>
<td>30.57</td>
<td>57</td>
<td>2.86</td>
<td>31.64</td>
<td>27</td>
<td>0.95</td>
<td>31.85</td>
<td>22</td>
<td>0.41</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>29.01</td>
<td>147</td>
<td>1.86</td>
<td>28.96</td>
<td>105</td>
<td>5.06</td>
<td>30.16</td>
<td>27</td>
<td>0.94</td>
<td>30.14</td>
<td>22</td>
<td>0.39</td>
</tr>
<tr>
<td>Caribou</td>
<td>15%</td>
<td>27.48</td>
<td>125</td>
<td>1.61</td>
<td>27.48</td>
<td>78</td>
<td>3.83</td>
<td>28.42</td>
<td>26</td>
<td>0.94</td>
<td>28.35</td>
<td>22</td>
<td>0.41</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>26.22</td>
<td>183</td>
<td>2.27</td>
<td>26.23</td>
<td>172</td>
<td>8.02</td>
<td>27.06</td>
<td>27</td>
<td>0.94</td>
<td>27.22</td>
<td>22</td>
<td>0.40</td>
</tr>
</tbody>
</table>
(1) \( (u_0^0)_{ij} = z_{ij} \) for all \( i \) and \( j \)
(2) for \( k = 0 \) to \( \text{maxit} \) do
  (3) Set \( v_0^k = u_0^k \)
  (4) Compute \( v_1^k \) using (65) for all \( i \) and \( j \)
  (5) Compute \( v_2^k \) using (66) for all \( i \) and \( j \)
  (6) \( u_0^{k+1} = v_0^k + v_1^k + v_2^k \)
  (7) if \( \| u_0^k - u_0^{k-1} \| / \| u_0^k \| < \text{tol} \) or \( \text{PSNR}(u_0^k) > \text{PSNR}(u_0^{k+1}) \) then
    (8) Stop
  (9) end if
(10) end for

**Algorithm 2: RHPM(2) method.**

(1) \( (u_0^0)_{ij} = z_{ij} \) for all \( i \) and \( j \)
(2) for \( k = 0 \) to \( \text{maxit} \) do
  (3) Set \( v_0^k = u_0^k \)
  (4) Compute \( v_1^k \) using (65) for all \( i \) and \( j \)
  (5) \( u_0^{k+1} = v_0^k + v_1^k \)
  (6) if \( \| u_0^k - u_0^{k-1} \| / \| u_0^k \| < \text{tol} \) or \( \text{PSNR}(u_0^k) > \text{PSNR}(u_0^{k+1}) \) then
    (7) Stop
  (8) end if
(9) end for

**Algorithm 3: RHPM(1) method.**

Boundary conditions are defined the same as those in TM method. Assume that \( (u_0^0)_{ij} = z(x_i, y_j) = z_{ij} \) and \( (u_0^0)_{ij} \) has been computed for all \( i \) and \( j \). Then, \( (v_0^k)_{ij} = (u_0^0)_{ij} \) and discretization of (60) is given by

\[
(v_1^k)_{ij} = -F_{ij} + \left( AF_{ij} + G_{ij} \right) t_k. \tag{65}
\]

Also, discretization of (63) is given by

\[
(v_2^k)_{ij} = -AF_{ij} + (A_1)_{ij} \left( F_{xx} + F_{yy} \right) \\
+ \left[ (B_1)_{ij} \left( D_3 \right)_{ij} - (B_2)_{ij} \left( D_5 \right)_{ij} \right] \\
\cdot \left[ \left( v_0 \right)_{ij} F_x + \left( v_0 \right)_{ij} F_y \right] t_k + (A_2)_{ij} \left[ \left( v_0 \right)_{ij}^2 F_{xx} \\
+ \left( v_0 \right)_{ij}^2 F_{yy} \right] \\
+ 2 \left\{ \left( v_0 \right)_{ij} \left( v_0 \right)_{ij} F_{xx} + \left( v_0 \right)_{ij} \left( v_0 \right)_{ij} F_{yy} \right\} t_k + \frac{1}{2} \left( t_k \right)^2 \\
+ \left\{ \left( v_0 \right)_{ij} \left( H_3 \right)_{ij} - \left( H_4 \right)_{ij} \right\} \left( \frac{1}{2} \left( t_k \right)^2 \right) \\
+ \left( v_0 \right)_{ij} \left( H_2 \right)_{ij} \left( \frac{1}{2} \left( t_k \right)^2 \right) - \left( A_2 \right)_{ij} \left( \frac{1}{2} \left( t_k \right)^2 \right) \left( H_3 \right)_{ij} \\
+ \left( v_0 \right)_{ij} \left( H_4 \right)_{ij} \left( \frac{1}{2} \left( t_k \right)^2 \right) + \left( v_0 \right)_{ij} \left( H_2 \right)_{ij} + 2 \left( v_0 \right)_{ij} \left( H_3 \right)_{ij} \\
+ 2 \left( v_0 \right)_{ij} \left( v_0 \right)_{ij} \left( H_4 \right)_{ij} + \left( v_0 \right)_{ij} \left( v_0 \right)_{ij} + 2 \left( v_0 \right)_{ij} \left( v_0 \right)_{ij} \left( H_2 \right)_{ij} \right. \\
\cdot \left( v_0 \right)_{ij} \left( H_2 \right)_{ij} \left( \frac{1}{2} \left( t_k \right)^2 \right), \tag{66}
\]

where

\[
(v_0)_{ij} = (v_0^k)_{ij} = (v_0^{k-1})_{ij} + (v_0^{k-1})_{ij} + (v_0^{k-1})_{ij} \\
(v_0)_{x} = (v_0^k)_{x} = (v_0^{k-1})_{x} = (v_0^{k-1})_{x} \\
(v_0)_{xx} = (v_0)_{xx} = (v_0)_{xx} = (v_0)_{xx} \\
(v_0)_{y} = (v_0^k)_{y} = (v_0^{k-1})_{y} = (v_0^{k-1})_{y} \\
F_{ij} = (v_0^k)_{ij} - z(i, j), \\
G_{ij} = (A_1)_{ij} \left( D_3 \right)_{ij} - (A_2)_{ij} \left( D_5 \right)_{ij}, \\
F_x = (F_x)_{ij} = F_{i+1} - F_{ij}, \\
F_{xx} = (F_{xx})_{ij} = F_{i+1} - 2F_{ij} + F_{i+1}, \\
F_y = (F_y)_{ij} = F_{ij+1} - F_{ij}, \\
F_{yy} = (F_{yy})_{ij} = F_{ij+1} - 2F_{ij} + F_{ij+1}, \\
(D_1)_{ij} = ((v_0)_{ij+1} - (v_0)_{ij})^2 + ((v_0)_{ij+1} - (v_0)_{ij})^2, \\
(D_3)_{ij} = ((v_0)_{ij-1} - (v_0)_{ij})^2 + ((v_0)_{ij-1} - (v_0)_{ij})^2.
Figure 5: Example of the Nodules image: Original image (a), Noisy image (b), TM denoising (c), RHAM2 denoising (d), RHPM(2) denoising (e), and RHPM(1) denoising (f).

\[
(D_2)_{ij} = \left\{ (v_0)_{i+1,j} - (v_0)_{ij} \right\}^2 \left\{ (v_0)_{i-1,j} - 2(v_0)_{ij} \right\}
\]
\[
+ (v_0)_{i,j+1} \right\} \right\},
\]
\[
(B_2)_{ij} = \sqrt{\varepsilon} \left[ \frac{-179}{8} \right. + \frac{135\varepsilon}{4} \left( D_1 \right)_{ij}
\]
\[
- \frac{105\varepsilon^2}{8} \left\{ \left( D_1 \right)_{ij} \right\}^2 \right],
\]
\[
(H_1)_{ij} = \lambda \left( F_{i+1,j} - F_{ij} \right) + \left( G_{i+1,j} - G_{ij} \right),
\]
\[
(H_2)_{ij} = \lambda \left( F_{i,j+1} - F_{ij} \right) + \left( G_{ij+1} - G_{ij} \right),
\]
\[
(H_3)_{ij} = \lambda \left( F_{i-1,j} - 2F_{ij} + F_{i+1,j} \right) + \left( G_{i-1,j} - 2G_{ij} \right.
\]
\[
+ G_{ij+1} \left\} \right),
\]
\[
(H_4)_{ij} = \lambda \left( F_{i,j-1} - 2F_{ij} + F_{i+1,j} \right) + \left( G_{ij-1} - 2G_{ij} \right.
\]
\[
+ G_{ij+1} \left\} \right). \tag{68}
\]

We now compute \( u_0^{k+1} = u_0^k + u_1^k + u_2^k \). Repeating the above process with \( u_0^{k+1} \) as an initial approximation, we can obtain RHPM(2) method, called Algorithm 2, for solving (6).
Since the computation of $v_k^2$ in RHPM(2) method is so complicated, we also provide RHPM(1) method, called Algorithm 3, whose computational cost is much cheaper than Algorithm 2. In RHPM(2) and RHPM(1) methods, tol, maxit, z, and u are the tolerance for stopping criterion, the maximum number of iterations, the Noisy image, and the true image, respectively. In (68), the coefficients of $(D_1)$ may have large values which cause amplification of some incorrect results. Hence, we have used an intensity range $[0, 1]$ for all test images used in RHPM methods to guarantee numerical stability.

5. Numerical Experiments

In this section, numerical experiments for several test images are carried out to compare the performance of RHPM(2) and RHPM(1) with those of existing TM and recently proposed RHAM2 in [27]. Numerical tests are performed using MATLAB R2014a on a personal computer with a processor at 3.6 GHz and 8 GB RAM. The Noisy images are generated by adding Gaussian white noise to the clean images using the built-in MATLAB function `randn`. For example, adding 15% of Gaussian white noise to the true image $B$ can be done as follows:

$$E = \text{randn}(m, n);$$
$$E = \frac{E}{\|E\|_F};$$
$$\tilde{B} = B + 0.15 \times \|B\|_F \times E,$$

where $\|\cdot\|_F$ refers to the Frobenius norm.

In order to illustrate efficiency and reliability of the proposed denoising algorithms (i.e., Algorithms 2 and 3), we provide experimental results for several images such as artificial, synthetic, and real images. 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

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

where $u$ and $u^k$ are the Original and restored images, respectively. Also $u_{ij}$ stands for the value of Original image.
Table 2: Numerical results for image denoising problems using TM, RHAM2, RHPM(2), and RHPM(1) methods with $\Delta t = 0.10$.

<table>
<thead>
<tr>
<th>Image</th>
<th>Noise</th>
<th>PSNR</th>
<th>TM Iter</th>
<th>CPU</th>
<th>PSNR</th>
<th>RHAM2 Iter</th>
<th>CPU</th>
<th>PSNR</th>
<th>RHPM(2) Iter</th>
<th>CPU</th>
<th>PSNR</th>
<th>RHPM(1) Iter</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cameraman</td>
<td>15%</td>
<td>30.27</td>
<td>200</td>
<td>9.54</td>
<td>30.27</td>
<td>67</td>
<td>13.14</td>
<td>30.19</td>
<td>39</td>
<td>5.14</td>
<td>30.25</td>
<td>31</td>
<td>1.82</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>28.65</td>
<td>282</td>
<td>13.24</td>
<td>28.56</td>
<td>117</td>
<td>22.44</td>
<td>28.76</td>
<td>40</td>
<td>5.18</td>
<td>28.89</td>
<td>32</td>
<td>1.86</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>25.32</td>
<td>431</td>
<td>20.09</td>
<td>25.31</td>
<td>286</td>
<td>54.55</td>
<td>25.63</td>
<td>40</td>
<td>5.17</td>
<td>25.81</td>
<td>32</td>
<td>1.85</td>
</tr>
<tr>
<td>Lake</td>
<td>15%</td>
<td>27.35</td>
<td>201</td>
<td>9.70</td>
<td>27.31</td>
<td>67</td>
<td>12.64</td>
<td>27.60</td>
<td>39</td>
<td>5.07</td>
<td>27.69</td>
<td>31</td>
<td>1.80</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>25.93</td>
<td>287</td>
<td>13.48</td>
<td>25.90</td>
<td>121</td>
<td>22.69</td>
<td>26.36</td>
<td>39</td>
<td>5.07</td>
<td>26.39</td>
<td>31</td>
<td>1.80</td>
</tr>
<tr>
<td>Mandrill</td>
<td>15%</td>
<td>26.15</td>
<td>156</td>
<td>7.58</td>
<td>26.16</td>
<td>48</td>
<td>9.40</td>
<td>26.64</td>
<td>39</td>
<td>4.73</td>
<td>26.56</td>
<td>32</td>
<td>1.86</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>24.61</td>
<td>228</td>
<td>10.75</td>
<td>24.60</td>
<td>82</td>
<td>15.36</td>
<td>25.38</td>
<td>39</td>
<td>4.90</td>
<td>25.35</td>
<td>32</td>
<td>1.85</td>
</tr>
<tr>
<td>Nodules</td>
<td>15%</td>
<td>28.10</td>
<td>289</td>
<td>7.65</td>
<td>28.20</td>
<td>124</td>
<td>13.59</td>
<td>28.80</td>
<td>40</td>
<td>2.89</td>
<td>29.06</td>
<td>32</td>
<td>1.03</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>26.60</td>
<td>407</td>
<td>10.41</td>
<td>26.62</td>
<td>251</td>
<td>25.04</td>
<td>27.07</td>
<td>40</td>
<td>2.84</td>
<td>27.22</td>
<td>32</td>
<td>1.02</td>
</tr>
</tbody>
</table>

Table 3: Numerical results for image denoising problems using TM, RHAM2, RHPM(2), and RHPM(1) methods with $\Delta t = 0.05$.

<table>
<thead>
<tr>
<th>Image</th>
<th>Noise</th>
<th>PSNR</th>
<th>TM Iter</th>
<th>CPU</th>
<th>PSNR</th>
<th>RHAM2 Iter</th>
<th>CPU</th>
<th>PSNR</th>
<th>RHPM(2) Iter</th>
<th>CPU</th>
<th>PSNR</th>
<th>RHPM(1) Iter</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cameraman</td>
<td>15%</td>
<td>30.29</td>
<td>402</td>
<td>19.04</td>
<td>30.16</td>
<td>58</td>
<td>11.73</td>
<td>30.15</td>
<td>76</td>
<td>9.82</td>
<td>30.10</td>
<td>59</td>
<td>3.26</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>28.57</td>
<td>566</td>
<td>26.26</td>
<td>28.63</td>
<td>85</td>
<td>16.35</td>
<td>28.84</td>
<td>77</td>
<td>9.84</td>
<td>28.88</td>
<td>60</td>
<td>3.32</td>
</tr>
<tr>
<td>Jet-plane</td>
<td>15%</td>
<td>26.78</td>
<td>609</td>
<td>28.64</td>
<td>26.82</td>
<td>93</td>
<td>18.16</td>
<td>27.12</td>
<td>77</td>
<td>9.77</td>
<td>27.18</td>
<td>60</td>
<td>3.34</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>25.38</td>
<td>861</td>
<td>40.20</td>
<td>25.29</td>
<td>150</td>
<td>28.78</td>
<td>25.72</td>
<td>78</td>
<td>9.85</td>
<td>25.87</td>
<td>61</td>
<td>3.35</td>
</tr>
<tr>
<td>Lake</td>
<td>15%</td>
<td>27.30</td>
<td>398</td>
<td>18.68</td>
<td>27.35</td>
<td>58</td>
<td>11.06</td>
<td>27.59</td>
<td>73</td>
<td>9.30</td>
<td>27.51</td>
<td>57</td>
<td>3.17</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>25.95</td>
<td>576</td>
<td>26.73</td>
<td>25.93</td>
<td>86</td>
<td>17.44</td>
<td>26.38</td>
<td>76</td>
<td>9.64</td>
<td>26.45</td>
<td>59</td>
<td>3.24</td>
</tr>
<tr>
<td>Nodules</td>
<td>15%</td>
<td>28.21</td>
<td>581</td>
<td>14.90</td>
<td>28.19</td>
<td>88</td>
<td>9.04</td>
<td>28.92</td>
<td>78</td>
<td>5.41</td>
<td>29.15</td>
<td>61</td>
<td>1.84</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>26.68</td>
<td>815</td>
<td>20.58</td>
<td>26.70</td>
<td>139</td>
<td>14.05</td>
<td>27.26</td>
<td>79</td>
<td>5.55</td>
<td>27.67</td>
<td>61</td>
<td>1.82</td>
</tr>
<tr>
<td>Rice</td>
<td>15%</td>
<td>30.61</td>
<td>312</td>
<td>3.76</td>
<td>30.53</td>
<td>46</td>
<td>2.32</td>
<td>31.59</td>
<td>76</td>
<td>2.47</td>
<td>31.55</td>
<td>59</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>28.92</td>
<td>438</td>
<td>5.25</td>
<td>28.91</td>
<td>63</td>
<td>3.20</td>
<td>30.24</td>
<td>77</td>
<td>2.50</td>
<td>30.27</td>
<td>60</td>
<td>0.88</td>
</tr>
<tr>
<td>Caribou</td>
<td>15%</td>
<td>27.49</td>
<td>374</td>
<td>4.53</td>
<td>27.52</td>
<td>55</td>
<td>2.76</td>
<td>28.44</td>
<td>75</td>
<td>2.54</td>
<td>28.42</td>
<td>58</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>26.21</td>
<td>552</td>
<td>6.56</td>
<td>26.21</td>
<td>83</td>
<td>4.04</td>
<td>27.20</td>
<td>77</td>
<td>2.53</td>
<td>27.25</td>
<td>60</td>
<td>0.89</td>
</tr>
</tbody>
</table>

$u$ at the grid point $(i, j)$ and $m \cdot n$ is the total number of pixels. It is clear that the larger PSNR is, the better the quality denoising image is.

We have tested the denoising algorithms for several images with an intensity range $[0, 255]$ for TM and RHAM2 methods and $[0, 1]$ for RHPM(2) and RHPM(1) methods, and Noisy images are generated by adding 15% or 20% of Gaussian white noise to true images. For all of the 4 algorithms, parameters $\text{tol} = 10^{-4}$ and $\text{maxit} = 5000$ have been used. In TM and RHAM2 methods, parameters $\lambda = 10^{-6}$ and $\beta = 10^{-3}$ are used. In RHPM(2) and RHPM(1) methods, parameters $\lambda = 0.35$ and $\epsilon = 10^{-4}$ are used. Note that the parameters are not optimal and are chosen as the best one by tries.

Tables 1–4 provide numerical results for TM, RHAM2, RHPM(2), and RHPM(1) methods corresponding to $\Delta t = 0.15$, 0.10, 0.05, and 0.01, respectively. In Tables 1–4, column PSNR contains the PSNR values, column Iter contains the number of iterations, and column CPU contains the elapsed CPU time in seconds. Figures 1 to 7 show the images denoised by TM, RHAM2, RHPM(2), and RHPM(1) methods with $\Delta t = 0.05$ for Noisy images with 20% of Gaussian white noise. Note that the denoised images for other values of $\Delta t$ and 15% of Gaussian white noise are almost the same. The first row images are Original image and Noisy image with 20% of Gaussian white noise and the image denoised by TM, and the second row images are the images denoised by RHAM2, RHPM(2), and RHPM(1) methods. In Figures 1 to 4, Cameraman, Jet-plane, Lake, and Mandrill images with $512 \times 512$ size are used. In Figure 5, Nodules image with $389 \times 366$ size is used. In Figures 6 and 7, Rice and Caribou images with $256 \times 256$ size are used.
Table 4: Numerical results for image denoising problems using TM, RHAM2, RHPM(2), and RHPM(1) methods with $\Delta t = 0.01$.

<table>
<thead>
<tr>
<th>Image</th>
<th>Noise</th>
<th>TM PSNR</th>
<th>Iter</th>
<th>CPU</th>
<th>RHAM2 PSNR</th>
<th>Iter</th>
<th>CPU</th>
<th>RHPM(2) PSNR</th>
<th>Iter</th>
<th>CPU</th>
<th>RHPM(1) PSNR</th>
<th>Iter</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cameraman</td>
<td>15%</td>
<td>30.18</td>
<td>1994</td>
<td>92.34</td>
<td>30.22</td>
<td>88</td>
<td>17.01</td>
<td>30.00</td>
<td>367</td>
<td>46.31</td>
<td>29.69</td>
<td>275</td>
<td>14.49</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>28.68</td>
<td>2823</td>
<td>129.78</td>
<td>28.61</td>
<td>107</td>
<td>20.70</td>
<td>28.73</td>
<td>376</td>
<td>42.41</td>
<td>28.47</td>
<td>281</td>
<td>14.83</td>
</tr>
<tr>
<td>Jet-plane</td>
<td>15%</td>
<td>26.85</td>
<td>3033</td>
<td>139.50</td>
<td>26.83</td>
<td>113</td>
<td>23.16</td>
<td>27.08</td>
<td>373</td>
<td>46.50</td>
<td>26.83</td>
<td>280</td>
<td>14.76</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>25.35</td>
<td>4292</td>
<td>197.00</td>
<td>25.29</td>
<td>142</td>
<td>27.31</td>
<td>25.77</td>
<td>380</td>
<td>47.58</td>
<td>25.75</td>
<td>284</td>
<td>14.88</td>
</tr>
<tr>
<td>Lake</td>
<td>15%</td>
<td>27.39</td>
<td>2003</td>
<td>92.20</td>
<td>27.33</td>
<td>88</td>
<td>17.78</td>
<td>27.53</td>
<td>358</td>
<td>44.91</td>
<td>27.30</td>
<td>270</td>
<td>14.25</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>24.62</td>
<td>2270</td>
<td>104.40</td>
<td>24.63</td>
<td>95</td>
<td>17.82</td>
<td>25.32</td>
<td>360</td>
<td>45.15</td>
<td>24.93</td>
<td>271</td>
<td>14.23</td>
</tr>
<tr>
<td>Nodules</td>
<td>15%</td>
<td>28.18</td>
<td>2900</td>
<td>72.85</td>
<td>28.10</td>
<td>109</td>
<td>11.05</td>
<td>28.94</td>
<td>379</td>
<td>25.60</td>
<td>28.85</td>
<td>284</td>
<td>8.03</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>26.56</td>
<td>4067</td>
<td>102.17</td>
<td>26.63</td>
<td>136</td>
<td>13.71</td>
<td>27.44</td>
<td>384</td>
<td>25.99</td>
<td>27.77</td>
<td>288</td>
<td>8.14</td>
</tr>
<tr>
<td>Rice</td>
<td>15%</td>
<td>30.62</td>
<td>1550</td>
<td>17.99</td>
<td>30.55</td>
<td>77</td>
<td>3.77</td>
<td>31.46</td>
<td>368</td>
<td>11.53</td>
<td>31.06</td>
<td>275</td>
<td>3.74</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>28.88</td>
<td>2181</td>
<td>25.48</td>
<td>28.98</td>
<td>92</td>
<td>4.44</td>
<td>30.07</td>
<td>376</td>
<td>11.69</td>
<td>29.76</td>
<td>281</td>
<td>3.74</td>
</tr>
<tr>
<td>Caribou</td>
<td>15%</td>
<td>27.52</td>
<td>1865</td>
<td>21.67</td>
<td>27.50</td>
<td>85</td>
<td>4.11</td>
<td>28.45</td>
<td>363</td>
<td>11.42</td>
<td>28.10</td>
<td>273</td>
<td>3.65</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>26.12</td>
<td>2725</td>
<td>31.68</td>
<td>26.22</td>
<td>106</td>
<td>5.10</td>
<td>27.24</td>
<td>373</td>
<td>11.56</td>
<td>27.01</td>
<td>280</td>
<td>3.74</td>
</tr>
</tbody>
</table>

Figure 7: Example of the Caribou image: Original image (a), Noisy image (b), TM denoising (c) RHAM2 denoising (d), RHPM(2) denoising (e), and RHPM(1) denoising (f).
As can be seen in Tables 1–4, \( \Delta t = 0.15 \) yields the best performance of 4 different cases of \( \Delta t \) for TM, RHPM(2), and RHPM(1) methods, but \( \Delta t = 0.05 \) yields the best performance for RHAM2 method. It can also be seen that RHPM methods perform much faster than TM and RHAM2 methods. On the other hand, PSNR values are close to one another. Overall, RHPM methods give a little bit higher PSNR values than TM and RHAM2 methods. In particular, RHPM gives about 1 higher PSNR value than TM and RHAM2 methods for Rice and Caribou images which have small sizes compared to the other images. According to numerical experiments, this phenomenon occurs because of the shape of image, not the size of image. That is, PSNR values are image-dependent, not size-dependent.

Observe that RHPM(1) computing 2 terms \( v_0, v_1 \) every iteration performs more efficiently than RHPM(2) computing 3 terms of \( v_0, v_1, v_2 \) every iteration. The reason why RHPM(2) performs worse than RHPM(1) is as follows: the computational cost for computing \( v_2 \) is expensive as compared with \( v_1 \), and its computation involves several approximation formulas, which use only a few terms of the infinite series, due to the difficulty in handling highly nonlinear terms. To this end, convergence rate of RHPM(2) is slower than RHPM(1).

6. Conclusion

In this paper, we proposed a restarted homotopy perturbation method (RHPM) for solving a nonlinear PDE problem, and we presented an application of RHPM to TV-based image denoising problem. We also provided numerical methods RHPM(2) and RHPM(1) which use binomial series techniques to settle the main difficulty in handling nonlinear terms. Numerical experiments show that RHPM(2) and RHPM(1) are effective, feasible, and reliable for TV-based image denoising problem as compared with TM and RHAM2 methods. Notice that RHPM(1) is very efficient compared to other methods since it has smaller iteration number and less CPU time than others. Also RHPM(1) is much simpler to implement than RHPM(2). So, we recommend the use of RHPM(1) with \( \lambda = 0.35 \) and \( \Delta t = 0.15 \) over RHPM(2) for TV-based image denoising problem. This work only studied the effectiveness, feasibility, and reliability of RHPM. So, future work will include comparison results between the proposed RHPM methods and other good methods in the literature.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank the anonymous referees for their useful comments and for pointing out many related references, which greatly improved this paper. This work was supported by the research Grant of the Agency for Defense Development and the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by Ministry of Education (NRF-2013R1A1A2005722).

References


