EFFICIENT BELTRAMI IMAGE FILTERING VIA VECTOR EXTRAPOLATION METHODS[*]

GUY ROSMAN[†], LORINA DASCAL[†], AVRAM SIDI[†], AND RON KIMMEL[†]

Abstract. The Beltrami image flow is an effective non-linear filter, often used in color image processing. It was shown to be closely related to the median, total variation, and bilateral filters. It treats the image as a 2D manifold embedded in a hybrid spatial-feature space. Minimization of the image surface area yields the Beltrami flow. The corresponding diffusion operator is anisotropic and strongly couples the spectral components. Thus, there is so far no implicit, nor operator-splitting-based numerical scheme for the partial differential equation that describes the Beltrami flow in color. Usually, this flow is implemented by explicit schemes, which are stable only for very small time steps and therefore require many iterations. At the other end, vector extrapolation techniques accelerate the convergence of vector sequences, without explicit knowledge of the sequence generator. In this paper, we propose to use vector extrapolation techniques for accelerating the convergence of the explicit schemes for the Beltrami flow. Experiments demonstrate fast convergence and efficiency compared to explicit schemes.

Key words. Diffusion, PDE, Beltrami, extrapolation, filtering, denoising

AMS subject classifications. 60J60,58J35,65B05

1. Introduction.

The Beltrami framework, introduced in [38, 39, 44], is based on a nonlinear flow that was applied as an edge preserving denoising and deblurring algorithm for signals and especially multi-channel images, see for example [3]. Unlike related nonlinear filters such as the total variation filter [23, 1, 8], which can be computed efficiently using semi-implicit schemes [43], the Beltrami flow is usually implemented by an explicit finite difference approximation of the characterizing partial differential equation. Standard explicit finite difference schemes require small time-steps for stability that lead to a large number of iterations required for convergence to the desired solution. So far, there is no implicit scheme for the Beltrami flow, due to the strong coupling of the color components and its anisotropic nature. Our goal is to accelerate the slow convergence of the explicit schemes, for which we propose to employ vector extrapolation techniques.

As an alternative to the explicit scheme, an approximation using the short time kernel for the Beltrami operator was suggested in [40]. This method is still computationally demanding, since computing the kernel operation involves geodesic distance computation around each pixel. A semi-implicit scheme has been devised in [11] for an approximation of the Beltrami flow. This approximation is not, however, consistent with the PDE characterizing the Beltrami flow. Rather, it discretizes a slightly different PDE. The Beltrami flow is also strongly linked to similar diffusion processes defined on non-flat manifolds [41], and formulations of it were solved on manifolds as well as images [40, 36].

Strongly related to the Beltrami operator, the bilateral operator was studied in different contexts (see for example [35], [42], [37], [14], [4]), and can be shown to be an approximation of the Beltrami kernel. This filter has later been extended to the the non-local means filter [6].

[*]This work was partly supported by United States -Israel Binational Science Foundation grant No. 2004274 and by the Ministry of Science grant No. 3-3414, and in part by Elias Fund for Medical Research.

[†]Department of Computer Science, Technion, Israel Institute of Technology, Haifa, 32000 Israel (`rosman,lorina,asidi,ron@cs.technion.ac.il`)

In this paper, we propose to apply vector extrapolation methods to accelerate the convergence rate of standard explicit schemes for the Beltrami flow in color. Specifically, we use the minimal polynomial extrapolation (MPE) method of Cabay and Jackson [7] and the reduced rank extrapolation (RRE) method of Mešina and Eddy [20, 13]. Because both MPE and RRE take as their input only a vector sequence obtained from a fixed-point iterative procedure, they can be applied not only to linearly generated sequences, but also to nonlinear ones. This allows us to employ them for accelerating the convergence of the vector sequences generated by the explicit finite-difference schemes for the Beltrami geometric flow. This approach for efficient Beltrami filtering was introduced in [10], and we elaborate upon it in this paper, further detailing the theory and practice of vector extrapolation methods. We demonstrate the efficiency and accuracy of MPE and RRE in color image processing applications, such as scale-space analysis, denoising, and deblurring.

This paper is organized as follows: Section 2 gives a brief summary of the Beltrami framework. In Section 3, we review approximations based on standard explicit finite-difference schemes. In Section 4, we present a detailed review of the MPE and RRE methods, which were only briefly discussed before in the context of image processing [10]. This review includes the derivation of the methods, computationally efficient and stable algorithms for their implementation, their known convergence theory, and a mode of application known as *cycling*. In Section 5, we apply RRE to our Beltrami color flow and demonstrate the resulting speed-up. Section 6 concludes the paper.

2. The Beltrami Framework.

Let us briefly review the Beltrami framework for non-linear diffusion in computer vision [18, 38, 39, 44]. For a more complete introduction to the Riemannian geometry tools used, henceforth, we refer the reader to [12].

We represent images as embedding maps of Riemannian manifolds in a higher dimensional space. Denote such a map by $X : \Sigma \to M$, where Σ is a two-dimensional surface, with (σ^1, σ^2) denoting coordinates on it. We denote by M the spatial-feature manifold, embedded in $\mathbb{R}^{d+2}$, where d is the number of image channels. For example, a gray-level image can be represented as a 2D surface embedded in $\mathbb{R}^3$. The map X in this case is $X(\sigma^1, \sigma^2) = (\sigma^1, \sigma^2, I(\sigma^1, \sigma^2))$, where I is the image intensity. For color images, X is given by $X(\sigma^1, \sigma^2) = (\sigma^1, \sigma^2, I^1(\sigma^1, \sigma^2), I^2(\sigma^1, \sigma^2), I^3(\sigma^1, \sigma^2))$, where I^1, I^2, I^3 are the three components of the color vector (for example, red, green, blue for the RGB color space).

Next, we choose a Riemannian metric on this surface. Its components are denoted by g_{ij}. The canonical choice of coordinates in image processing is Cartesian (we denote them here by x^1 and x^2). For such a choice, which we follow for the rest of the paper, we identify $\sigma^1 = x^1$ and $\sigma^2 = x^2$. In this case, σ^1 and σ^2 are the image coordinates. We denote the elements of the inverse of the metric by superscripts g^{ij}, and the determinant by $g = \det(g_{ij})$. Once images are defined as embedding of Riemannian manifolds, it is natural to look for a measure on this space of embedding maps.

Denote by (Σ, g) the image manifold and its metric, and by (M, h) the space-feature manifold and its metric. Then, the functional $S[X]$ attaches a real number to a map $X : \Sigma \to M$,

$$S[X, g_{ij}, h_{ab}] = \int d^m\sigma \sqrt{g} ||dX||^2_{g,h}, \tag{2.1}$$

where m is the dimension of Σ, g is the determinant of the image metric, and the range of indices is $i, j = 1, 2, ..., \dim(\Sigma)$ and $a, b = 1, 2, ..., \dim(M)$. The integrand $||dX||^2_{g,h}$

is expressed in a local coordinate system, by $||dX||^2_{g,h} = (\partial_{x_i} I^a) g^{ij} (\partial_{x_j} I^b) h_{ab}$. We have used here Einstein summation convention: identical indices that appear up and down are summed over. This functional, for $\dim(\Sigma) = 2$ and $h_{ab} = \delta_{ab}$, was first proposed by Polyakov [22] in the context of high energy physics, in the theory known as *string theory*.

The elements of the induced metric for color images with Cartesian color coordinates are

$$G = (g_{ij}) = \begin{pmatrix} 1+\beta^2 \sum_{a=1}^3 (I^a_{x_1})^2 & \beta^2 \sum_{a=1}^3 I^a_{x_1} I^a_{x_2} \\ \beta^2 \sum_{a=1}^3 I^a_{x_1} I^a_{x_2} & 1+\beta^2 \sum_{a=1}^3 (I^a_{x_2})^2 \end{pmatrix},$$

where a subscript of I denotes a partial derivative and the parameter $\beta > 0$ determines the ratio between the spatial and spectral (color) distances. Using standard methods in the calculus of variations, the Euler-Lagrange equations minimizing S with respect to the embedding (assuming Euclidean embedding space) are

$$0 = -\frac{1}{\sqrt{g}} h^{ab} \frac{\delta S}{\delta I^b} = \underbrace{\frac{1}{\sqrt{g}} \operatorname{div}(D \nabla I^a)}_{\Delta_g I^a}, \tag{2.2}$$

where the matrix $D = \sqrt{g} G^{-1}$. See [38] for explicit derivation. The operator that acts on I^a is the natural generalization of the Laplacian from flat spaces to manifolds, it is called the Laplace-Beltrami operator, and is denoted by Δ_g.

The parameter β, in the elements of the metric g_{ij}, determines the nature of the flow. At the limits, where $\beta \to 0$ and $\beta \to \infty$, we obtain respectively a linear diffusion flow and a nonlinear flow, akin to the TV flow for the case of grey-level images (see [39] for details).

The Beltrami scale-space emerges as a gradient descent minimization process

$$I^a_t = -\frac{1}{\sqrt{g}} \frac{\delta S}{\delta I^a} = \Delta_g I^a, \tag{2.3}$$

with reflective boundary conditions and a smooth initial solution $I^a|_{t=0} = I^a_0$. For Euclidean embedding, the functional in Eq. 2.1 reduces to

$$S(X) = \int \sqrt{g}\, dx^1\, dx^2,$$

where

$$g = \det(g_{ij}) = 1 + \beta^2 \sum_{a=1}^{3} |\nabla I^a|^2 + \frac{1}{2}\beta^4 \sum_{a,b=1}^{3} |\nabla I^a \times \nabla I^b|^2. \tag{2.4}$$

The role of the cross product term $\sum_{a,b=1}^3 |\nabla I^a \times \nabla I^b|^2$ in the minimization was explored in [18]. It enforces the Lambertian model of image formation by penalizing misalignments of the gradient directions in the various color channels. Accordingly, taking large values of β makes sense as we would expect $\sum_{a,b=1}^3 |\nabla I^a \times \nabla I^b|^2$ to vanish in natural images and $\sum_{a=1}^3 |\nabla I^a|^2$ to be small.

The geometric functional S can be used as a regularization term for color image processing. In the variational framework, the reconstructed image is the minimizer of a cost-functional. Functionals using the Beltrami flow for regularization can be written in the general form

$$\Psi = \frac{\alpha}{2}\sum_{a=1}^{3} ||KI^a - I_0^a||^2 + S(X),$$

where K is a bounded linear operator. In the denoising case, K is the identity operator $Ku = u$, and in the deblurring case, $Ku = k{*}u$, k is the blurring kernel (often assumed to be Gaussian), and $\bar{k}(x, y) = k(-x, -y)$. The parameter α controls the smoothness of the solution.

The modified Euler-Lagrange equations as a gradient descent process for each case are

i) for denoising:

$$I_t^a = -\frac{1}{\sqrt{g}}\frac{\delta\Psi}{\delta I^a} = -\frac{\alpha}{\sqrt{g}}(I^a - I_0^a) + \Delta_g I^a. \tag{2.5}$$

ii) for deblurring:

$$I_t^a = -\frac{1}{\sqrt{g}}\frac{\delta\Psi}{\delta I^a} = -\frac{\alpha}{\sqrt{g}}\bar{k} * (k * I^a - I_0^a) + \Delta_g I^a. \tag{2.6}$$

The Laplace-Beltrami operator in Eqs. (2.5) and (2.6) provides us with an adaptive smoothing mechanism. In areas with large gradients (edges), the fidelity term is suppressed and the regularizing term becomes dominant. At homogenous regions with low-gradient magnitude, the fidelity term takes over and controls the flow.

3. Standard Explicit Finite Difference Scheme. Our goal is to speed-up the convergence of the explicit scheme in Beltrami color processing. In this section, we detail the standard explicit scheme. The applications we address are the Beltrami-based smoothing, Beltrami-based denoising, and Beltrami-based deblurring.

We work on a rectangular grid with step sizes Δt in time and Δx in space. The spatial units are normalized such that $\Delta x = 1$. For each channel I^a, $a \in \{1, 2, 3\}$, we define the discrete approximation $(I^a)_{ij}^n$ by

$$(I^a)_{ij}^n \approx I^a(i\Delta x, j\Delta x, n\Delta t).$$

On the boundary of the image we impose the Neumann boundary condition.

The explicit finite difference scheme is written in a general form as

$$(I^a)_{ij}^{n+1} = (I^a)_{ij}^n + \Delta t O_{ij}^n(I^a), \tag{3.1}$$

where O_{ij}^n is the discretization of the right hand side of the relevant continuous equation (2.3), (2.5), or (2.6). Below, we give the exact form of the operator O_{ij}^n for each of the above cases.

- *Beltrami-based smoothing.*
  The explicit scheme (3.1) for discretizing Equation 2.3 takes the form

$$(I^a)_{ij}^{n+1} = (I^a)_{ij}^n + \Delta t L_{ij}^n(I^a), \tag{3.2}$$

  where $L_{ij}^n(U^a)$ denotes a discretization of the Laplace-Beltrami operator $\Delta_g U^a$, for example, using a backward-forward approximation.

- *Beltrami-based denoising.*
  The explicit scheme (3.1) is given in this case by

$$(I^a)^{n+1}_{ij} = (I^a)^n_{ij} + \Delta t \left(L^n_{ij}(I^a) + \frac{\alpha}{\sqrt{g}}((I^a_0)^n_{ij} - (I^a)^n_{ij}) \right). \tag{3.3}$$

- *Beltrami-based deblurring.*
  Similarly, in the deblurring case, the explicit scheme (3.1) reads

$$(I^a)^{n+1}_{ij} = (I^a)^n_{ij} + \Delta t \left(L^n_{ij}(I^a) + \frac{\alpha}{\sqrt{g}}\bar{k}^n_{ij} * \left((I^a_0)^n_{ij} - k^n_{ij} * (I^a)^n_{ij}\right) \right). \tag{3.4}$$

Due to stability requirements (see [9], [17]), explicit schemes limit the time-step Δt and usually require a large number of iterations in order to converge. We propose to use vector extrapolation techniques in order to accelerate the convergence of these explicit schemes.

4. MPE/RRE Acceleration Techniques. The minimal polynomial extrapolation (MPE) [7] and the reduced rank extrapolation (RRE) [20, 13] are two vector extrapolation methods that have proved to be very efficient in accelerating the convergence of vector sequences arising from fixed-point iteration schemes for nonlinear, as well as linear, large and sparse systems of equations. For a review of these methods and others, covering the relevant developments until mid '80s, see [34]. The brief review we present here covers the various developments that have taken place since the publication of [34].

Both methods are derived by considering vector sequences $\mathbf{x}_0, \mathbf{x}_1, \ldots,$ generated via a linear fixed-point iteration process, namely,

$$\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n + \mathbf{b}, \quad n = 0, 1, \ldots, \tag{4.1}$$

where $\mathbf{A}$ is a fixed $N \times N$ matrix and $\mathbf{b}$ is a fixed N-dimensional vector and $\mathbf{x}_0$ is an initial vector chosen by the user. Clearly, this sequence has a limit $\mathbf{s}$ that is the unique solution to the linear system

$$\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{b}, \tag{4.2}$$

provided $\rho(\mathbf{A}) < 1$, where $\rho(\mathbf{A})$ is the spectral radius of $\mathbf{A}$. [Note that the system in (4.2) can also be written as $(\mathbf{I} - \mathbf{A})\mathbf{x} = \mathbf{b}$, and the uniqueness of the solution $\mathbf{s}$ follows from our assumption that $\rho(\mathbf{A}) < 1$, which guarantees that the matrix $\mathbf{I} - \mathbf{A}$ is nonsingular since 1 is not an eigenvalue of $\mathbf{A}$.]

We now turn to simple derivations of MPE and RRE, that are based on those given in [34]. Other derivations, based on the Shanks–Schmidt transformation [26, 25] have been given in [30]. For a detailed treatment of this transformation, see [29, Chapter 16].

Given the sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots,$ generated as in (4.1), let

$$\mathbf{u}_n = \Delta\mathbf{x}_n = \mathbf{x}_{n+1} - \mathbf{x}_n, \quad n = 0, 1, \ldots,$$

and define the error vectors $\boldsymbol{\epsilon}_n$ as in

$$\boldsymbol{\epsilon}_n = \mathbf{x}_n - \mathbf{s}, \quad n = 0, 1, \ldots . \tag{4.3}$$

Making also use of the fact that $\mathbf{s} = \mathbf{A}\mathbf{s} + \mathbf{b}$, one can relate the error in step n to the initial error via

$$\boldsymbol{\epsilon}_n = (\mathbf{A}\mathbf{x}_{n-1} + \mathbf{b}) - (\mathbf{A}\mathbf{s} + \mathbf{b}) = \mathbf{A}(\mathbf{x}_{n-1} - \mathbf{s}) = \mathbf{A}\boldsymbol{\epsilon}_{n-1}, \tag{4.4}$$

which, by induction, gives

$$\boldsymbol{\epsilon}_n = \mathbf{A}^n \boldsymbol{\epsilon}_0, \quad n = 0, 1, \dots \ . \tag{4.5}$$

We now seek to approximate $\mathbf{s}$ by a "weighted average" of $k+1$ consecutive $\mathbf{x}_i$'s as in

$$\mathbf{s}_{n,k} = \sum_{i=0}^{k} \gamma_i \mathbf{x}_{n+i}; \quad \sum_{i=0}^{k} \gamma_i = 1. \tag{4.6}$$

Substituting Equation (4.3) in Equation (4.6), and making use of the fact that $\sum_{i=0}^{k} \gamma_i = 1$, we obtain

$$\mathbf{s}_{n,k} = \sum_{i=0}^{k} \gamma_i (\mathbf{s} + \boldsymbol{\epsilon}_{n+i}) = \mathbf{s} + \sum_{i=0}^{k} \gamma_i \boldsymbol{\epsilon}_{n+i} \tag{4.7}$$

which, by (4.5), becomes

$$\mathbf{s}_{n,k} = \mathbf{s} + \sum_{i=0}^{k} \gamma_i \mathbf{A}^{n+i} \boldsymbol{\epsilon}_0. \tag{4.8}$$

From this expression, it is clear that we must choose the scalars γ_i to make the vector $\sum_{i=0}^{k} \gamma_i \mathbf{A}^{n+i} \boldsymbol{\epsilon}_0$, the weighted sum of the error vectors $\boldsymbol{\epsilon}_{n+i}$, $i = 0, 1, \dots, k$, as small as possible. As we show next, we can actually make this vector vanish by choosing k and the γ_i appropriately.

Now, given a nonzero $N \times N$ matrix $\mathbf{B}$ and an arbitrary nonzero N-dimensional vector $\mathbf{u}$, it is known that there exists a unique monic polynomial $P(z)$ of smallest degree (that is at most N) that annihilates the vector $\mathbf{u}$, that is, $P(\mathbf{B})\mathbf{u} = \mathbf{0}$. This polynomial is called the *minimal polynomial of* $\mathbf{B}$ *with respect to the vector* $\mathbf{u}$. It is also known that $P(z)$ divides the minimal polynomial of $\mathbf{B}$, which divides the characteristic polynomial of $\mathbf{B}$. Thus, the zeros of $P(z)$ are some or all the eigenvalues of $\mathbf{B}$.

Thus, if the minimal polynomial of $\mathbf{A}$ with respect to $\boldsymbol{\epsilon}_n$ is

$$P(z) = \sum_{i=0}^{k} c_i z^i; \quad c_k = 1,$$

then

$$P(\mathbf{A})\boldsymbol{\epsilon}_n = \mathbf{0}.$$

By (4.5), this means that

$$\sum_{i=0}^{k} c_i \mathbf{A}^i \boldsymbol{\epsilon}_n = \sum_{i=0}^{k} c_i \boldsymbol{\epsilon}_{n+i} = \mathbf{0}. \tag{4.9}$$

This is a set of N linear equations in the k unknowns $c_0, c_1, \ldots, c_{k-1}$, with $c_k = 1$. In addition, these equations are consistent and have a unique solution because $P(z)$ is unique. From these equations, it seems, however, that we need to know the vector $\boldsymbol{\epsilon}_n = \mathbf{x}_n - \mathbf{s}$, hence the solution $\mathbf{s}$. Fortunately, this is not the case, and we can obtain the c_i solely from our knowledge of the vectors $\mathbf{x}_i$. This is done as follows: Multiplying Equation (4.9) by $\mathbf{A}$, and recalling (4.4), we have

$$\mathbf{0} = \sum_{i=0}^{k} c_i \mathbf{A}\boldsymbol{\epsilon}_{n+i} = \sum_{i=0}^{k} c_i \boldsymbol{\epsilon}_{n+i+1}.$$

Subtracting from this Equation (4.9), we obtain

$$\mathbf{0} = \sum_{i=0}^{k} c_i(\boldsymbol{\epsilon}_{n+i+1} - \boldsymbol{\epsilon}_{n+i}) = \sum_{i=0}^{k} c_i(\mathbf{x}_{n+i+1} - \mathbf{x}_{n+i}),$$

hence the linear system

$$\sum_{i=0}^{k} c_i \mathbf{u}_{n+i} = \mathbf{0}. \tag{4.10}$$

Once $c_0, c_1, \ldots, c_{k-1}$ have been determined from this linear system, we set $c_k = 1$ and let $\gamma_i = c_i / \sum_{j=0}^{k} c_j$, $i = 0, 1, \ldots, k$. This is allowed because $\sum_{j=0}^{k} c_j = P(1) \neq 0$ by the fact that $\mathbf{I} - \mathbf{A}$ is not singular and hence $\mathbf{A}$ does not have 1 as an eigenvalue. Summing up, we have shown that if k is the degree of the minimal polynomial of $\mathbf{A}$ with respect to $\boldsymbol{\epsilon}_n$, then there exist scalars $\gamma_0, \gamma_1, \ldots, \gamma_k$, satisfying $\sum_{i=0}^{k} \gamma_i = 1$, such that $\sum_{i=0}^{k} \gamma_i \mathbf{x}_{n+i} = \mathbf{s}$.

At this point, we note that, $\mathbf{s}$ is the solution to $(\mathbf{I} - \mathbf{A})\mathbf{x} = \mathbf{b}$, whether $\rho(\mathbf{A}) < 1$ or not. Thus, with the γ_i as determined above, $\mathbf{s} = \sum_{i=0}^{k} \gamma_i \mathbf{x}_{n+i}$, whether $\lim_{n\to\infty} \mathbf{x}_n$ exists or not.

In the sequel, we shall use the notation

$$\mathbf{U}_s^{(j)} = [\,\mathbf{u}_j \,|\, \mathbf{u}_{j+1} \,|\, \ldots \,|\, \mathbf{u}_{j+s}\,]. \tag{4.11}$$

Thus, $\mathbf{U}_s^{(j)}$ is an $N \times (s+1)$ matrix. In this notation, Equation (4.9) reads

$$\mathbf{U}_k^{(n)}\mathbf{c} = \mathbf{0}; \quad \mathbf{c} = [c_0, c_1, \ldots, c_k]^{\mathrm{T}}. \tag{4.12}$$

Of course, dividing Equation (4.12) by $\sum_{i=0}^{k} c_i$, we also have

$$\mathbf{U}_k^{(n)}\boldsymbol{\gamma} = \mathbf{0}; \quad \boldsymbol{\gamma} = [\gamma_0, \gamma_1 \ldots, \gamma_k]^{\mathrm{T}}. \tag{4.13}$$

4.1. Derivation of MPE. As we already know, the degree of the minimal polynomial of $\mathbf{A}$ with respect to $\boldsymbol{\epsilon}_n$ can be as large as N. This makes the process we have just described a prohibitively expensive one, since we have to save all the vectors $\mathbf{x}_{n+i}$ $i = 0, 1, \ldots, k+1$, which is a problem when N is very large. In addition, we also do not have a way to know this degree. Given these facts, we modify the approach we have just described as follows: We choose k to be an arbitrary positive integer that is normally (much) smaller than the degree of the minimal polynomial of $\mathbf{A}$ with respect to $\boldsymbol{\epsilon}_n$. With this k, the linear system in Equation (4.10) is not consistent, hence does not have a solution for $c_0, c_1, \ldots, c_{k-1}$, with $c_k = 1$, in the ordinary sense. Therefore, we solve this system in the least squares sense. Following that, we compute $\gamma_0, \gamma_1, \ldots, \gamma_k$ precisely as described following Equation (4.10), and then compute the vector $\mathbf{s}_{n,k} = \sum_{i=0}^{k} \gamma_i \mathbf{x}_{n+i}$ as our approximation to $\mathbf{s}$. The resulting method is known as the minimal polynomial extrapolation (MPE) method. Clearly, MPE takes as its input only the integers k and n and the vectors $\mathbf{x}_n, \mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+k+1}$, hence can be employed whether these vectors are generated by a linear or nonlinear iterative process.

We can summarize the definition of MPE through the following steps:
1. Choose the integers k and n, and input the vectors $\mathbf{x}_n, \mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+k+1}$.
2. Form the $N \times k$ matrix $\mathbf{U}_k^{(n)}$.
3. Solve the overdetermined linear system $\mathbf{U}_{k-1}^{(n)}\mathbf{c}' = -\mathbf{u}_{n+k}$ by least squares. Here $\mathbf{c}' = [c_0, c_1, \ldots, c_{k-1}]^{\mathrm{T}}$.
Set $c_k = 1$, and $\gamma_i = c_i / \sum_{i=0}^{k} c_i$, $i = 0, 1, \ldots, k$.
4. Compute the vector $\mathbf{s}_{n,k} = \sum_{i=0}^{k} \gamma_i \mathbf{x}_{n+i}$ as approximation to $\lim_{i\to\infty} \mathbf{x}_i = \mathbf{s}$.

4.2. Derivation of RRE. Again, we choose k to be an arbitrary positive integer that is normally (much) smaller than the degree of the minimal polynomial of $\mathbf{A}$ with respect to $\boldsymbol{\epsilon}_n$. With this k, the linear system in Equation (4.13) is not consistent, hence does not have a solution for $\gamma_0, \gamma_1, \ldots, \gamma_k$, in the ordinary sense. Therefore, we solve this system in the least squares sense, subject to the constraint $\sum_{i=0}^{k} \gamma_i = 1$. Following that, we compute the vector $\mathbf{s}_{n,k} = \sum_{i=0}^{k} \gamma_i \mathbf{x}_{n+i}$ as our approximation to $\mathbf{s}$. The resulting method is known as the reduced rank extrapolation (RRE) method. Clearly, RRE, just as MPE, takes as its input only the integers k and n and the vectors $\mathbf{x}_n, \mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+k+1}$, hence can be employed whether these vectors are generated by a linear or nonlinear iterative process.

We can summarize the definition of RRE through the following steps:
1. Choose the integers k and n, and input the vectors $\mathbf{x}_n, \mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+k+1}$.
2. Form the $N \times k$ matrix $\mathbf{U}_k^{(n)}$.
3. Form the $N \times (k+1)$ matrix $\mathbf{U}_k^{(n)}$, and solve the overdetermined linear system $\mathbf{U}_k^{(n)}\boldsymbol{\gamma} = \mathbf{0}$ by least squares, subject to the constraint $\sum_{i=0}^{k} \gamma_i = 1$. Here $\boldsymbol{\gamma} = [\gamma_0, \gamma_1, \ldots, \gamma_k]^{\mathrm{T}}$.
4. Compute the vector $\mathbf{s}_{n,k} = \sum_{i=0}^{k} \gamma_i \mathbf{x}_{n+i}$ as approximation to $\lim_{i\to\infty} \mathbf{x}_i = \mathbf{s}$.

4.3. Treatment of Nonlinear Equations. We now turn to the treatment of nonlinear equations, such as those based on the Beltrami framework, by vector extrapolation methods. Assume that the system of nonlinear equations in question has been written in the (possibly preconditioned) form

$$\mathbf{x} = \mathbf{F}(\mathbf{x}), \tag{4.14}$$

where $\mathbf{F}(\mathbf{x})$ is an N-dimensional vector-valued function and $\mathbf{x}$ is the N-dimensional vector of unknowns, such as the column-stacked vector form of the discretized image. Let the sequence of approximations $\mathbf{x}_n$ to the solution $\mathbf{s}$ be generated via

$$\mathbf{x}_{n+1} = \mathbf{F}(\mathbf{x}_n), \quad n = 0, 1, \ldots, \tag{4.15}$$

and assume that this sequence converges to the solution vector $\mathbf{s}$. In our case, $\mathbf{F}$ is the right-hand side of the explicit discretization equation for the Beltrami color flow (given in a general form in Equation 3.1).

$$\mathbf{F}((I^a)_{ij}) = (I^a)_{ij} + \Delta t O_{ij}^n(I^a),$$

For $\mathbf{x}$ close to $\mathbf{s}$, $\mathbf{F}(\mathbf{x})$ can be expanded in a Taylor series in the form

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}(\mathbf{s}) + \mathbf{F}'(\mathbf{s})(\mathbf{x}-\mathbf{s}) + O(\|\mathbf{x}-\mathbf{s}\|^2) \quad \text{as } \mathbf{x} \to \mathbf{s}.$$

Here $\mathbf{F}'(\mathbf{x})$ is the Jacobian matrix of the vector-valued function $\mathbf{F}(\mathbf{x})$. Recalling also that $\mathbf{F}(\mathbf{s}) = \mathbf{s}$, this expansion can be put in the form

$$\mathbf{F}(\mathbf{x}) = \mathbf{s} + \mathbf{F}'(\mathbf{s})(\mathbf{x}-\mathbf{s}) + O(\|\mathbf{x}-\mathbf{s}\|^2) \quad \text{as } \mathbf{x} \to \mathbf{s}.$$

By the assumption that the sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots,$ converges to $\mathbf{s}$ [which takes place provided $\rho(\mathbf{F}'(\mathbf{s})) < 1$], it follows that $\mathbf{x}_n$ is close to $\mathbf{s}$ for all large n, and hence

$$\mathbf{x}_{n+1} = \mathbf{s} + \mathbf{F}'(\mathbf{s})(\mathbf{x}_n-\mathbf{s}) + O(\|\mathbf{x}_n-\mathbf{s}\|^2) \quad \text{as } n \to \infty.$$

Rewriting this in the form

$$\mathbf{x}_{n+1} - \mathbf{s} = \mathbf{F}'(\mathbf{s})(\mathbf{x}_n-\mathbf{s}) + O(\|\mathbf{x}_n-\mathbf{s}\|^2) \quad \text{as } n \to \infty,$$

we realize that, for all large n, the vectors $\mathbf{x}_n$ behave as if they were generated by a linear system of the form $(\mathbf{I}-\mathbf{A})\mathbf{x} = \mathbf{b}$ via

$$\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n + \mathbf{b}, \quad n = 0, 1, \ldots, \tag{4.16}$$

where $\mathbf{A} = \mathbf{F}'(\mathbf{s})$ and $\mathbf{b} = [\mathbf{I} - \mathbf{F}'(\mathbf{s})]\mathbf{s}$. This suggests that the extrapolation methods MPE and RRE [that were designed by considering vector sequences generated by a linear fixed-point iterative process as in (4.1)] can be applied to sequences of vectors obtained from nonlinear fixed-point iterative methods. Indeed, methods such as MPE and RRE have been applied with success to the numerical solution of large and sparse nonlinear systems of equations arising in various areas of science and engineering, such as computational fluid dynamics, semiconductor research, and computerized tomography.

4.4. Efficient Implementation of MPE and RRE. In subsections 4.1 and 4.2, we gave the definitions of MPE and RRE. These definitions actually form the basis for the implementations of MPE and RRE that have been presented in [28]. The most important aspect of these implementations is the accurate solution of the relevant least-squares problems and minimization of computing time and storage requirements. The implementations we give in the sequel, were developed in [28], where a FORTRAN 77 code is also included.

In these implementations, the least-squares problems are solved by using QR factorizations of the matrices $\mathbf{U}_k^{(n)}$, as in

$$\mathbf{U}_k^{(n)} = \mathbf{Q}_k\mathbf{R}_k.$$

Here $\mathbf{Q}_k$ is an $N \times (k+1)$ unitary matrix satisfying $\mathbf{Q}_k^*\mathbf{Q}_k = \mathbf{I}_{(k+1)\times(k+1)}$. Thus, $\mathbf{Q}_k$ has the columnwise partition

$$\mathbf{Q}_k = [\,\mathbf{q}_0 \,|\, \mathbf{q}_1 \,|\, \cdots \,|\, \mathbf{q}_k\,], \tag{4.17}$$

such that the columns $\mathbf{q}_i$ form an orthonormal set of N-dimensional vectors, that is, $\mathbf{q}_i^*\mathbf{q}_j = \delta_{ij}$. The matrix $\mathbf{R}_k$ is a $(k+1)\times(k+1)$ upper triangular matrix with positive diagonal elements. Thus,

$$\mathbf{R}_k = \begin{bmatrix} r_{00} & r_{01} & \cdots & r_{0k} \\ & r_{11} & \cdots & r_{1k} \\ & & \ddots & \vdots \\ & & & r_{kk} \end{bmatrix}; \quad r_{ii} > 0, \quad i = 0, 1, \ldots, k. \tag{4.18}$$

This factorization can be carried out easily and accurately using the modified Gram–Schmidt orthogonalization process (MGS). See, for example, [16] and [28]. For completeness, we give here the steps of MGS as applied to the matrix $\mathbf{U}_k^{(n)}$:

```
1. Compute $r_{00} = \|\mathbf{u}_n\|$ and $\mathbf{q}_0 = \mathbf{u}_n/r_{00}$.
2. For $i = 1, \ldots, k$ do
       Set $\mathbf{u}_i^{(0)} = \mathbf{u}_{n+i}$
       For $j = 0, \ldots, i-1$ do
           $r_{jk} = \mathbf{q}_j^* \mathbf{u}_i^{(j)}$ and $\mathbf{u}_i^{(j+1)} = \mathbf{u}_i^{(j)} - r_{jk}\mathbf{q}_j$
       end
       Compute $r_{ii} = \|\mathbf{u}_i^{(i)}\|$ and $\mathbf{q}_i = \mathbf{u}_i^{(i)}/r_{ii}$.
   end
```

Here, $\|\mathbf{x}\| = \sqrt{\mathbf{x}^*\mathbf{x}}$. In addition, the vector $\mathbf{u}_i^{(j+1)}$ overwrites $\mathbf{u}_i^{(j)}$, so that the vectors $\mathbf{u}_{n+i}$, $\mathbf{u}_i^{(j)}$ and $\mathbf{q}_i$ all occupy the same storage location.

Note that $\mathbf{Q}_k$ is obtained from $\mathbf{Q}_{k-1}$ by appending to the latter the vector $\mathbf{q}_k$ as the $(k+1)$th column. Similarly, $\mathbf{R}_k$ is obtained from $\mathbf{R}_{k-1}$ by appending to the latter the $\mathbf{0}$ vector as the $(k+1)$th row and then the vector $[r_{0k}, r_{1k}, \ldots, r_{kk}]^{\mathrm{T}}$ as the $(k+1)$th column.

An important point we wish to emphasize is that, when forming the matrix $\mathbf{U}_k^{(n)}$, we overwrite the vector $\mathbf{x}_{n+i}$ with $\mathbf{u}_{n+i} = \Delta\mathbf{x}_{n+i}$ as soon as the latter is computed, for $i = 1, \ldots, k$. We save only $\mathbf{x}_n$. Next, when computing the matrix $\mathbf{Q}_k$, we overwrite $\mathbf{u}_{n+i}$ with $\mathbf{q}_i$, $i = 0, 1, \ldots, k$. This means that, at all stages of the computation of $\mathbf{Q}_k$ and $\mathbf{R}_k$, we are keeping only $k+2$ vectors in memory. The vectors $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+k+1}$ need not be saved.

With the QR factorization of $\mathbf{U}_k^{(n)}$ (hence of $\mathbf{U}_{k-1}^{(n)}$) available we can give algorithms for MPE and RRE within a unified framework as follows:

Algorithms for MPE and RRE

1. Input: k and n and the vectors $\mathbf{x}_n, \mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+k+1}$.
2. Compute the vectors $\mathbf{u}_{n+i} = \Delta\mathbf{x}_{n+i}$, $i = 0, 1, \ldots, k$, and form the $N \times (k+1)$ matrix

$$\mathbf{U}_k^{(n)} = [\,\mathbf{u}_n \,|\, \mathbf{u}_{n+1} \,|\, \cdots \,|\, \mathbf{u}_{n+k}\,],$$

   and form its QR factorization, namely, $\mathbf{U}_k^{(n)} = \mathbf{Q}_k\mathbf{R}_k$, with $\mathbf{Q}_k$ and $\mathbf{R}_k$ as in (4.17) and (4.18).
3. Determination of the γ_i:
   - *For MPE*
     With $\boldsymbol{\rho}_k = [r_{0k}, r_{1k}, \ldots, r_{k-1,k}]^{\mathrm{T}}$, solve the $k \times k$ upper triangular system

$$\mathbf{R}_{k-1}\mathbf{c}' = -\boldsymbol{\rho}_k; \quad \mathbf{c}' = [c_0, c_1, \ldots, c_{k-1}]^{\mathrm{T}}.$$

     Set $c_k = 1$, and $\gamma_i = c_i / \sum_{i=0}^{k} c_i$, $i = 0, 1, \ldots, k$.
   - *For RRE*
     With $\mathbf{e} = [1, 1, \ldots, 1]^{\mathrm{T}}$, solve the $(k+1) \times (k+1)$ linear system

$$\mathbf{R}_k^*\mathbf{R}_k\mathbf{d} = \mathbf{e}; \quad \mathbf{d} = [d_0, d_1, \ldots, d_k]^{\mathrm{T}}.$$

     This amounts to solving two triangular systems: first $\mathbf{R}_k^*\mathbf{a} = \mathbf{e}$ for $\mathbf{a}$, and, following that, $\mathbf{R}_k\mathbf{d} = \mathbf{a}$ for $\mathbf{d}$. Next, compute $\lambda = 1/\sum_{i=0}^{k} d_i$; λ is always positive (it becomes zero only when $\mathbf{s}_{n,k} = \mathbf{s}$ in the linear case). Next, set $\boldsymbol{\gamma} = \lambda\mathbf{d}$, that is $\gamma_i = \lambda d_i$, $i = 0, 1, \ldots, k$.

4. With the γ_i computed, compute $\boldsymbol{\xi} = [\xi_0, \xi_1, \ldots, \xi_{k-1}]^{\mathrm{T}}$ via

$$\xi_0 = 1 - \gamma_0; \quad \xi_j = \xi_{j-1} - \gamma_j, \quad j = 1, \ldots, k-1.$$

5. Compute

$$\boldsymbol{\eta} = [\eta_0, \eta_1, \ldots, \eta_{k-1}]^{\mathrm{T}} = \mathbf{R}_{k-1}\boldsymbol{\xi}.$$

Then compute

$$\mathbf{s}_{n,k} = \mathbf{x}_n + \mathbf{Q}_{k-1}\boldsymbol{\eta} = \mathbf{x}_n + \sum_{i=0}^{k-1} \eta_i \mathbf{q}_i.$$

4.5. Residual Estimation. One common way of assessing the quality of the approximation $\mathbf{s}_{n,k}$ is by looking at the residual vector $\mathbf{r}(\mathbf{s}_{n,k})$ associated with it. When the $\mathbf{x}_m$ are generated by the iterative process $\mathbf{x}_{m+1} = \mathbf{F}(\mathbf{x}_m)$, we have $\mathbf{r}(\mathbf{x}) = \mathbf{F}(\mathbf{x}) - \mathbf{x}$, and

$$\begin{aligned} \mathbf{r}(\mathbf{s}_{n,k}) &= \mathbf{U}_k^{(n)}\boldsymbol{\gamma} \quad \text{if } \mathbf{F}(\mathbf{x}) \text{ is linear } [\mathbf{F}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}], \\ \mathbf{r}(\mathbf{s}_{n,k}) &\approx \mathbf{U}_k^{(n)}\boldsymbol{\gamma} \quad \text{if } \mathbf{F}(\mathbf{x}) \text{ is nonlinear.} \end{aligned}$$

Therefore,

$$\begin{aligned} \|\mathbf{r}(\mathbf{s}_{n,k})\| &= \|\mathbf{U}_k^{(n)}\boldsymbol{\gamma}\| \quad \text{if } \mathbf{F}(\mathbf{x}) \text{ is linear,} \\ \|\mathbf{r}(\mathbf{s}_{n,k})\| &\approx \|\mathbf{U}_k^{(n)}\boldsymbol{\gamma}\| \quad \text{if } \mathbf{F}(\mathbf{x}) \text{ is nonlinear.} \end{aligned}$$

In addition, no matter how the $\mathbf{x}_m$ are generated, $\|\mathbf{U}_k^{(n)}\boldsymbol{\gamma}\|$ can be computed without having to compute $\mathbf{s}_{n,k}$ and $\mathbf{F}(\mathbf{s}_{n,k})$ explicitly, and at no cost, via

$$\|\mathbf{U}_k^{(n)}\boldsymbol{\gamma}\| = \begin{cases} r_{kk}|\gamma_k| & \text{for MPE,} \\ \sqrt{\lambda} & \text{for RRE.} \end{cases}$$

Here, r_{kk} is the last diagonal element of the matrix $\mathbf{R}_k$ and λ is the parameter computed in Step 3 of the algorithms in the preceding subsection. For details, see [28].

4.6. Cycling with MPE and RRE. The convergence acceleration properties of MPE and RRE have been studied in [27, 30, 32, 34, 35] as these methods are applied to a linearly generated vector sequence $\{\mathbf{x}_m\}$. When the matrix $\mathbf{A}$ in $\mathbf{F}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$ is diagonalizable, then the $\mathbf{x}_m$ are necessarily of the form $\mathbf{x}_m = \mathbf{s} + \sum_{i=1}^{p} \mathbf{v}_i \lambda_i^m$, where $(\lambda_i, \mathbf{v}_i)$ are some or all of the eigenpairs of $\mathbf{A}$, with distinct nonzero eigenvalues. With λ_i ordered as $|\lambda_1| \geq |\lambda_2| \geq |\lambda_3| \geq \cdots$, the following important asymptotic performance is achieved by both MPE and RRE when $|\lambda_k| > |\lambda_{k+1}|$:

$$\mathbf{s}_{n,k} - \mathbf{s} = O(\lambda_{k+1}^n) \quad \text{as } n \to \infty.$$

(This clearly shows that the sequence $\{\mathbf{s}_{n,k}\}_{n=0}^{\infty}$ converges to $\mathbf{s}$ faster than the sequence $\{\mathbf{x}_m\}$.) It can be shown that the same holds for RRE also when $|\lambda_k| = |\lambda_{k+1}|$.

This means that for large n, the fixed-point iterations reduce the contributions of the smaller λ_i to the error $\mathbf{s}_{n,k} - \mathbf{s}$, while MPE and RRE reduce the contributions of the k largest λ_i, that is, of $\lambda_1, \dots, \lambda_k$. The end result is, of course, that $\mathbf{s}_{n,k} - \mathbf{s}$ is smaller than each of $\mathbf{x}_{n+i} - \mathbf{s}$, $i = 0, 1, \dots, k$, when n is large.

Obviously, there is no way of letting n go to infinity in practice. It also follows from the asymptotic results just mentioned that increasing k generally makes MPE and RRE converge faster. However, we have no way of increasing k indefinitely either, because this would increase the storage requirements and also increase the computational cost tremendously.

In case we are solving the system $\mathbf{x} = \mathbf{F}(\mathbf{x})$ and the vectors $\mathbf{x}_i$ are generated via the fixed-point iterative procedure $\mathbf{x}_{n+1} = \mathbf{F}(\mathbf{x}_n)$, we can employ a mode of application called *cycling* or *restarting* in which n and k are held fixed. Here are the steps of this mode.

C0. Choose integers n, k, and an initial vector $\mathbf{x}_0$.
C1. Compute the vectors $\mathbf{x}_i$, $1 \leq i \leq n + k + 1$, [via $\mathbf{x}_{n+1} = \mathbf{F}(\mathbf{x}_n)$], and save $\mathbf{x}_{n+i}$, $0 \leq i \leq k + 1$.
C2. Apply MPE or RRE to the vectors $\mathbf{x}_{n+i}$, $0 \leq i \leq k+1$, precisely as described in subsection (4.4), with end result $\mathbf{s}_{n,k}$.
C3. If $\mathbf{s}_{n,k}$ satisfies accuracy test, stop.
Otherwise, set $\mathbf{x}_0 = \mathbf{s}_{n,k}$, and go to Step C1.

We will call each application of Steps C1–C3 a *cycle*. We will also denote the $\mathbf{s}_{n,k}$ that is computed in the ith cycle $\mathbf{s}^{(i)}_{n,k}$.

A discussion of the error in this mode of usage – in case of linear $\mathbf{F}(x)$, i.e., when $\mathbf{F}(x) = \mathbf{A}\mathbf{x} + \mathbf{b}$ – is given in [31] and [32]. The relevant errors can be shown to have upper bounds that are expressible in terms of Jacobi polynomials for certain types of spectra of the matrix $\mathbf{A}$, and these bounds turn out to be quite tight. They also indicate that, with even moderate n, $\mathbf{s}^{(i)}_{n,k}$ can closely approximate the solution $\mathbf{s}$ with small k, resulting in small storage requirements and few iterations. Another advantage of applying MPE and RRE in this mode (that is, with $n > 0$) is that it prevents stagnation in the cases where methods such as the generalized minimal residuals (GMRES) stagnate. (See the numerical examples in [31, 32].) Numerical experiments confirm that this is indeed the case. Furthermore, this is the case for nonlinear systems of equations as well, even though the analysis of [31, 32] does not apply to this case in a straightforward manner.

The analysis of MPE and RRE as these are applied to nonlinear systems in the cycling mode has been considered in the works [33, 34]. What can be said heuristically is that, when k in the ith cycle is chosen to be k_i, the degree of the minimal polynomial of the matrix $\mathbf{F}'(\mathbf{s})$ with respect to $\boldsymbol{\epsilon}_0 = \mathbf{x}_0 - \mathbf{s}$, the sequence $\{\mathbf{s}^{(i)}_{n,k_i}\}_{i=0}^{\infty}$ converges to $\mathbf{s}$ quadratically. However, we must add that, since the k_i can be as large as N and are not known exactly, this usage of cycling is not useful practically for the large-scale problems we are interested in solving. In other words, trying to achieve quadratic convergence from MPE and RRE via cycling may not be realistic. With even moderate values of n and k, we may achieve linear but fast convergence nevertheless. This turns out to be the case even when $\mathbf{x}_n$ is far from the solution $\mathbf{s}$.

4.7. Connection with Krylov Subspace Methods. When applied to linearly generated sequences, MPE and RRE are very closely related to the method of Arnoldi [2] and to GMRES [24], two well known Krylov subspace methods. The following result is stated in [27].

Theorem 1. *Consider the linear system* $(\mathbf{I} - \mathbf{A})\mathbf{x} = \mathbf{b}$. *With* $\mathbf{x}_0$ *as the initial vector,* *let the vector sequence $\{\mathbf{x}_n\}$ be generated via $\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n + \mathbf{b}$, and let $\mathbf{s}_{0,k}^{\mathrm{MPE}}$ and $\mathbf{s}_{0,k}^{\mathrm{RRE}}$ be obtained from this sequence by applying, respectively, MPE and RRE. Let also the vectors $\mathbf{s}_k^{\mathrm{Arnoldi}}$ and $\mathbf{s}_k^{\mathrm{GMRES}}$ be the vectors obtained by applying, respectively, k steps of the method of Arnoldi and GMRES to $(\mathbf{I} - \mathbf{A})\mathbf{x} = \mathbf{b}$, with $\mathbf{x}_0$ as the starting vector. Then $\mathbf{s}_{0,k}^{\mathrm{MPE}} = \mathbf{s}_k^{\mathrm{Arnoldi}}$ and $\mathbf{s}_{0,k}^{\mathrm{RRE}} = \mathbf{s}_k^{\mathrm{GMRES}}$.*

We also recall that the method of Arnoldi becomes the method of conjugate gradients when $\mathbf{A}$ is a Hermitian matrix.

It must be noted that the equivalence of MPE and RRE to the method of Arnoldi and to GMRES is mathematical and not algorithmic. The algorithms (computational procedures) are different.

Note: Another method which exploits previous descent directions in order to efficiently minimize a cost function is the sequential subspace optimization (SESOP) [15] method. Also aimed at solving nonlinear problems, this method extends the nonlinear conjugate gradients method [21], and can be highly efficient for problems of a certain sparse structure. While this method can be applied to the discretization of the functional $S(X)$, along with additional terms, using SESOP in this context raises several questions. Firstly, $S(X)$ is relatively costly to compute, and cannot be expressed in a relatively sparse manner as is often done in order to use SESOP effectively [15]. Secondly, the discretization used in computing the gradient of $S(X)$ may not completely agree with that used to compute $S(X)$. This can slow down or halt convergence. We have experimented with SESOP and compared it to the methods we suggest here. Specifically we have tried using SESOP with a truncated Newton, using conjugate gradients for inverting the Hessian. We did not, however, find SESOP as efficient for the problem of Beltrami-based denoising compared to the methods considered in this paper, namely constant step size gradient descent with extrapolation. Other variations on using SESOP for Beltrami-based image processing remain a topic for future work.

5. Experimental Results.

We have applied the MPE and the RRE to the problems described in Section 2 of this work. In this section, we proceed to demonstrate experimental results of the Beltrami scale-space and restoration of color images processed by the explicit and the MPE- and RRE-accelerated schemes, specifying the CPU runtime and the resulting speed-up. In all of the examples, an Intel® Core™2 Duo 1.83 GHz processor with 2GB of RAM was used. Although in each application we display results with respect to a single image, the behavior exhibited is similar for other input data.

We recall that the sequence of vectors $\mathbf{x}_1, \mathbf{x}_2, ...$ that form the input for RRE are generated according to

$$\mathbf{x}_{n+1} = \mathbf{F}(\mathbf{x}_n), \quad n = 0, 1, ...$$

where $\mathbf{F}(\mathbf{x})$ are those vector-valued functions that result from the finite-difference solution of the relevant Beltrami equations discussed in Section 2.

In our experiments, we apply the MPE and the RRE in the cycling mode. The MPE- and RRE-accelerated schemes allow us to reduce the number of explicit iterations by at least a factor of 10, in order to reach the same residual norm value. Experiments demonstrate that the MPE and RRE schemes remain stable as the number of iterations increases.

Figure 5.1 top row depicts the scale-space behavior of the Beltrami color flow, obtained using Equation 3.2. At the bottom right, it shows the speed-up obtained by using the RRE scheme for the Beltrami-based scale-space analysis. The speed-up gain is up to 40, as can be seen in the graph of the residual norm.

We have measured the approximation error of the accelerated sequence using l^2-norm values for the application of scale-space analysis. Comparison is done by running the explicit scheme and the RRE scheme simultaneously. After each RRE iteration, we advance the explicit sequence, starting from the previous result until it diverges from the RRE result. l^2-norm values indicate that the images obtained by the explicit and RRE techniques are "numerically" the same. The maximum l^2-norm relative error value observed during scale-space evolution was 0.241%, with a mean value of 0.122% and a standard deviation of 0.058%. The result for scale-space analysis is specifically important because of the obvious existence of a continuum of global minimizers for the functional without fidelity terms. Indeed, every constant valued image is a global minimizer of $S(X)$, while for scale-space analysis we wish to follow a specific gradient descent process. In applications involving a fidelity term or a deblurring term, the errors detected in the computed steady state solution were even smaller. These results validate numerically the convergence of the scheme. For these applications one would be more interested in the algebraic error of the solution with respect to the steady-state solution. A graph measuring the l^2-norm of the algebraic error, relative to the energy of the image, is shown in Figure 5.2 for both the MPE and the RRE methods, as well as for the explicit scheme. The steady-state solution is approximated by the numerical solution using a small step-size explicit scheme, as for natural images we do not have an analytical solution of Equation 2.5.

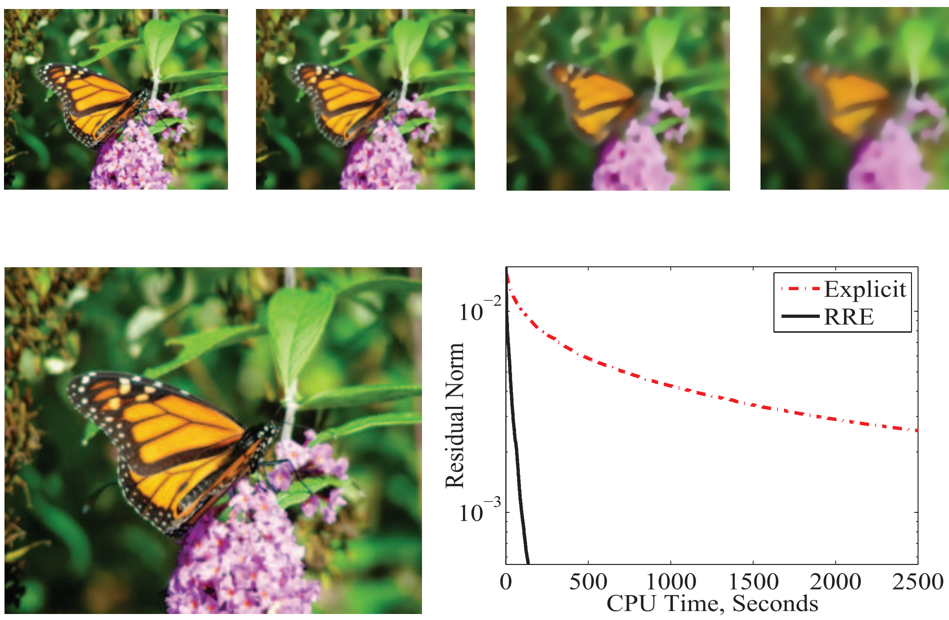


FIG. 5.1. *Top (left to right): RRE iterations:* 50, 150, 300, 414. *Bottom: left: Original picture. right: Comparison of the residual norms versus CPU time. Parameters:* $\beta = \sqrt{1000} \simeq 31.62$, $\Delta t = 0.001$.

5.1. Beltrami-based Denoising. Figure 5.3 displays the restoration of an image from its noisy version, corrupted by additive Gaussian noise, by applying Equation 3.3. The speed-up factor in this case is about 10.

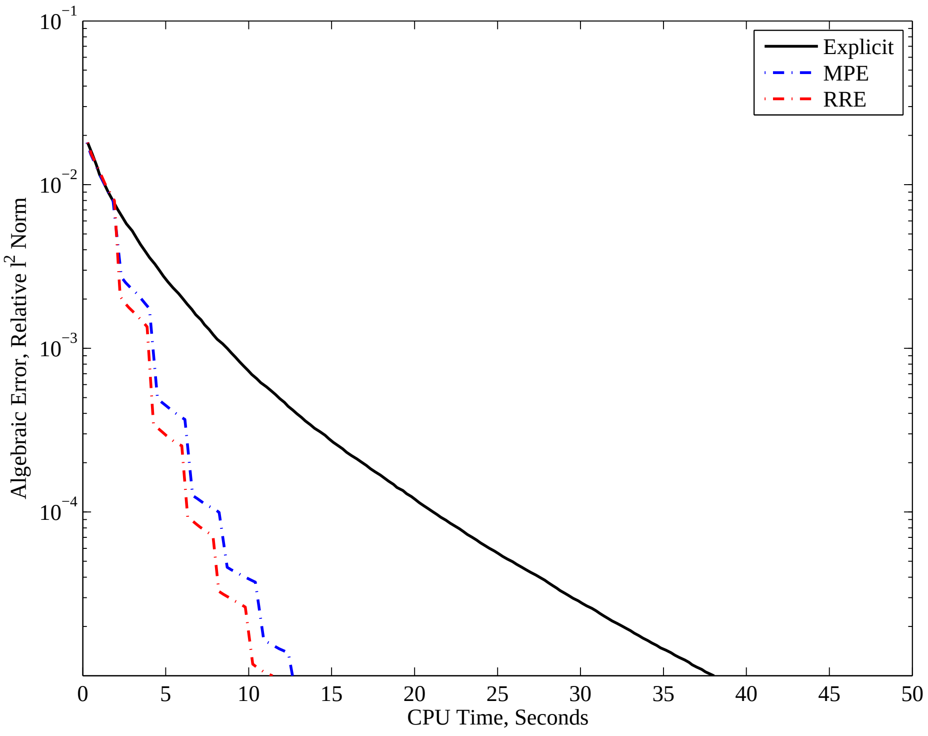
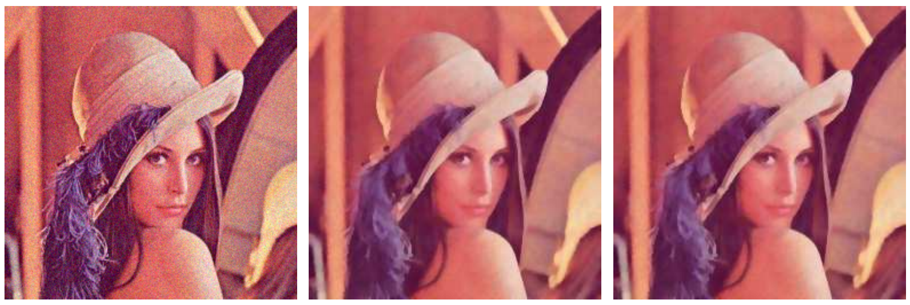


FIG. 5.2. *l^2-norm error between the sequences generated using the explicit scheme without acceleration, and the explicit scheme accelerated by MPE and RRE, with respect to the steady state solution. The acceleration used $k = 6, n = 0$, which gave the fastest convergence. Parameters: $\beta = \sqrt{200} \approx 14.14$, $\Delta t = 0.1$.*

5.2. Beltrami-based Deblurring. In the next example the original image was blurred by a Gaussian kernel, as shown in Figure 5.4 top-left. The image was restored using Equation 3.4. A significant speed-up is obtained in this case, as seen in Figure 5.4 bottom.

5.3. Denoising 3D images. Another application for the Beltrami flow is denoising of 3D images such as volume images found in medical application, or denoising of movie sequences. In Figure 5.5 we show an example of denoising of a medical CT image taken from the Stanford volume data archive [19]. While the data, being a gray-level image, can be processed more efficiently using various methods such as operator splitting schemes [43] or multigrid schemes, perhaps in combination with extrapolation techniques [5], we will not go into detailing usage of such methods so as to avoid deviating from the main topic of this paper.

5.4. Robustness to the Choice of Parameters. A natural question, regarding accelerations of nonlinear processes, is how does the speedup obtained depend on both parameters of the flow, and the parameter of the extrapolation algorithm. In practice, the MPE and the RRE algorithms seem to be quite robust to various choices of these parameters involved. For example, for relatively small time-steps, such as those warranted by the CFL condition, one can find, in practice, an optimal value of k, the length of the cycle. In our experiments, increasing n, the number of preconditioning iterations, did not result in an increasing speedup for the parameters



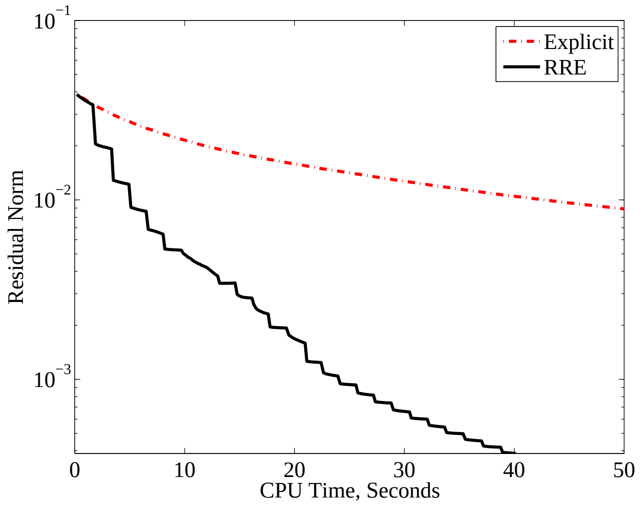


FIG. 5.3. *Beltrami-based denoising. Top left: Noisy image. Middle: Denoised image obtained by the RRE (901 iterations). Right: Denoised image obtained by the explicit scheme (11541 iterations). Bottom: Comparison of the residual norms versus CPU time. Parameters:* $\beta = \sqrt{1000} \approx 31.63$, $\lambda = 0.02$, $\Delta t = 0.0021/\beta^2$.

we have tested. For larger time-steps, slightly below instability sets in, however, it is harder to determine a clear trend in the speedup obtained as a function of n and k. We note that the speedup is still clearly apparent, as can be seen in Figure 5.6. Looking at the speedup obtained as a function of λ and β, we have noticed the speedup is relatively stable with respect to changes in β and decreases as λ becomes larger. The speedup remains significant for relevant values of λ, as can be seen in Figure 5.7.

6. Concluding Remarks.

Due to its anisotropic nature and non-separability, there is no implicit scheme, nor operator-splitting-based numerical scheme for the PDE characterizing the Beltrami flow in color. This flow is usually performed by means of explicit schemes. Low computational efficiency limits their use in practical applications. We accelerated the convergence of the explicit scheme using vector extrapolation methods. Experiments of denoising and deblurring of color images based on RRE have demonstrated the efficiency of the method. This makes vector extrapolation methods useful and attractive to the Beltrami filter and potentially other image processing applications.

REFERENCES

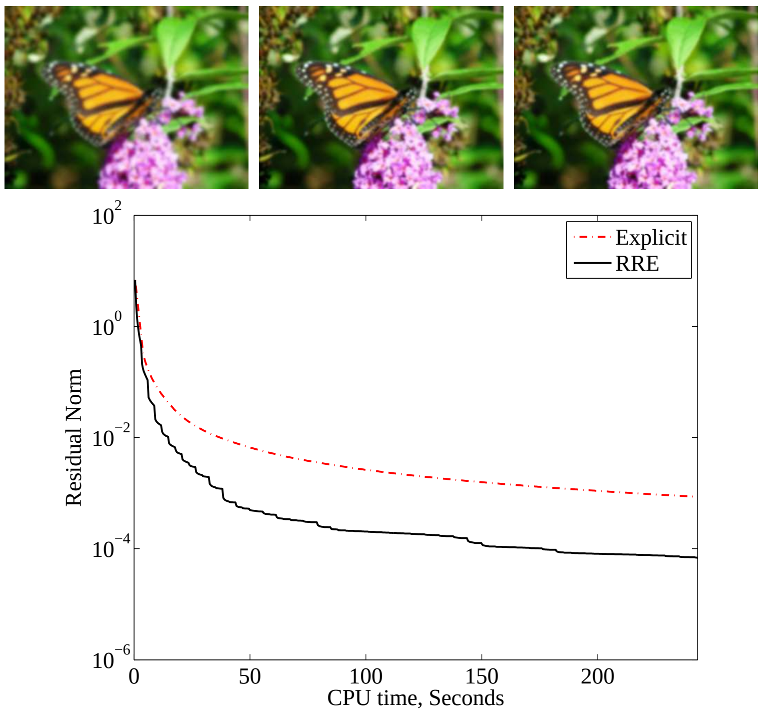


FIG. 5.4. *Beltrami-based deblurring. Top left: Blurred image. Middle: Deblurred image obtained by the RRE scheme (*1301 *iterations). Right: Deblurred image obtained by the explicit scheme (*196608 *iterations). Bottom: Comparison of the residual norms versus CPU time. Parameters:* $\beta = \sqrt{2000} \simeq 44.72$*,* $\Delta t = 0.0021/\beta^2$*,* $\lambda = 0.03$.

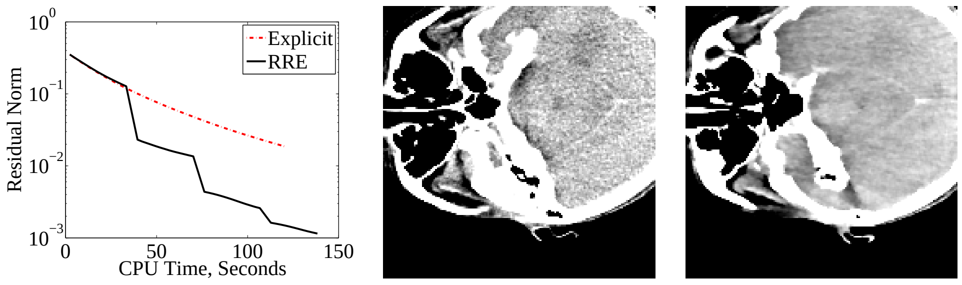


FIG. 5.5. *Beltrami-based denoising of a 3D image. Left: A comparison of the explicit and RRE-accelerated Beltrami schemes. Middle: A slice from the original volume image. Right: The same slice, denoised using the accelerated Beltrami flow. Parameters:* $\beta = 0.1$*,* $\Delta t = 0.02$*,* $\lambda = 1.5$.

[1] L. Alvarez, P.-L. Lions, and J.-M. Morel. Image selective smoothing and edge detection by nonlinear diffusion. II. *SIAM J. Numer. Anal.*, 29(3):845–866, 1992.

[2] W. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[3] L. Bar, A. Brook, N. A. Sochen, and N. Kiryati. Color image deblurring with impulsive noise. In N. Paragios, O. D. Faugeras, T. Chan, and C. Schnörr, editors, *Proceedings of the International Conference on Variational, Geometry and Level Sets Methods in Computer Vision*, volume 3752 of *Lecture Notes on Computer Science*, pages 49–60. Springer, 2005.

[4] D. Barash. A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(6):844–847, 2002.

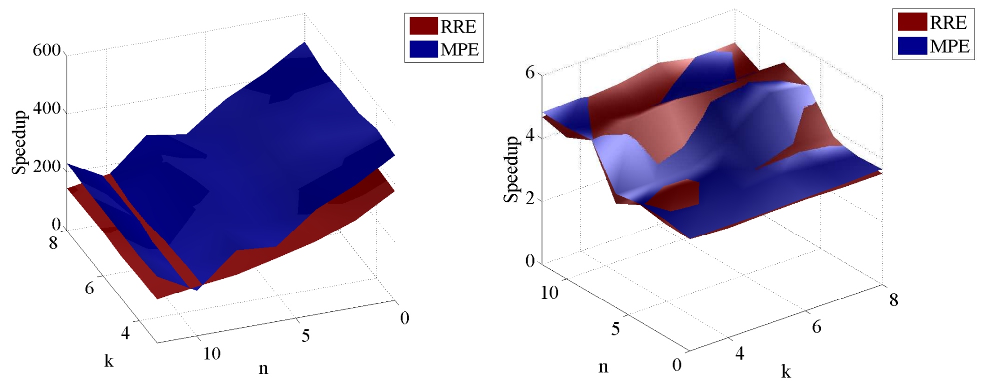


FIG. 5.6. *The speedup obtained for various values of n, the number of preconditioning iterations before each cycle and k, the cycle length. Left: The speedup obtained for a denoising problem such as the one shown in Figure 5.3, with $dt = 0.0042/\beta^2$. Right: the speedup for the same evolution with $dt = 0.1$. Parameters: $\beta = \sqrt{150} \approx 12.25$, $\lambda = 0.1$*

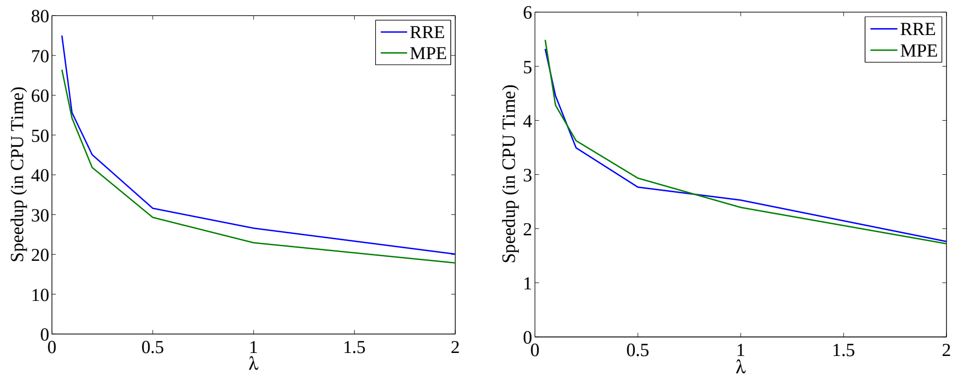


FIG. 5.7. *The speedup obtained for various values of λ, the fidelity factor. Left: The speedup obtained for $dt = 0.005$. Right: the speedup for the same evolution with $dt = 0.1$. Parameters: $\beta = 10$, $n = 0, k = 5$*

[5] A. Brandt and V. Mikulinsky. On recombining iterants in multigrid algorithms and problems with small islands. *SIAM J. Numer. Anal.*, 16(1):20–28, 1995.
[6] A. Buades, B. Coll, and J.-M. Morel. Nonlocal image and movie denoising. *International Journal of Computer Vision*, 76(2):123–139, February 2008.
[7] S. Cabay and L. Jackson. A polynomial extrapolation method for finding limits and antilimits of vector sequences. *SIAM J. Numer. Anal.*, 13:734–752, 1976.
[8] F. Catte, P. L. Lions, J. M. Morel, and T. Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM J. Numer. Anal.*, 29(1):182–193, 1992.
[9] R. Courant, K. Friedrichs, and H. Lewy. Uber die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100(1):32–74, 1928.
[10] L. Dascal, G. Rosman, and R. Kimmel. Efficient Beltrami filtering of color images via vector extrapolation. In *Scale Space and Variational Methods in Computer Vision*, volume 4485 of *Lecture Notes on Computer Science*, pages 92–103. Springer, 2007.
[11] T. Deschamps, R. Malladi, and I. Ravve. Fast evolution of image manifolds and application to filtering and segmentation in 3d medical images. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):525–535, 2004.
[12] M. P. do Carmo. *Riemannian Geometry.* Birkhäuser Verlag, Boston, MA, 1992.
[13] R. Eddy. Extrapolating to the limit of a vector sequence. In P. Wang, editor, *Information Linkage Between Applied Mathematics and Industry*, pages 387–396, New York, 1979. Academic Press.

[14] M. Elad. On the bilateral filter and ways to improve it. *IEEE Trans. Image Process.*, 11(10):1141–1151, 2002.

[15] M. Elad, B. Matalon, , and M. Zibulevsky. Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization. *Applied and Computational Harmonic Analysis*, 2007.

[16] G. Golub and C. V. Loan. *Matrix Computations.* The Johns Hopkins University Press, London, third edition, 1996.

[17] B. Gustafsson, H. Kreiss, and J. Oliger. *Time dependent problems and difference methods.* Wiley, New York, 1995.

[18] R. Kimmel, R. Malladi, and N. Sochen. Images as embedding maps and minimal surfaces: Movies, color, texture, and volumetric medical images. *International Journal of Computer Vision*, 39(2):111–129, 2000.

[19] M. Levoy. The stanford volume data archive, 2001.

[20] M. Mešina. Convergence acceleration for the iterative solution of the equations $X = AX + f$. *Comput. Methods Appl. Mech. Engrg.*, 10:165–173, 1977.

[21] G. Narkiss and M. Zibulevsky. Sequential subspace optimization method for large-scale unconstrained problems. Technical Report CCIT 559, EE Dept., Technion, 2005.

[22] A. M. Polyakov. Quantum geometry of bosonic strings. *Physics Letters*, 103 B(3):207–210, 1981.

[23] L. Rudin, S. Osher, and E. Fatemi. Non linear total variation based noise removal algorithms. *Physica D Letters*, 60:259–268, 1992.

[24] Y. Saad and M. Schultz. GMRES: A generalized minimal residual method for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7:856–869, 1986.

[25] J. Schmidt. On the numerical solution of linear simultaneous equations by an iterative method. *Phil. Mag.*, 7:369–383, 1941.

[26] D. Shanks. Nonlinear transformations of divergent and slowly convergent sequences. *J. Math. and Phys.*, 34:1–42, 1955.

[27] A. Sidi. Extrapolation vs. projection methods for linear systems of equations. *J. Comp. Appl. Math.*, 22:71–88, 1988.

[28] A. Sidi. Efficient implementation of minimal polynomial and reduced rank extrapolation methods. *J. Comp. Appl. Math.*, 36:305–337, 1991. Originally appeared as NASA TM-103240 ICOMP-90-20.

[29] A. Sidi. *Practical Extrapolation Methods: Theory and Applications.* Number 10 in Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, 2003.

[30] A. Sidi, W. Ford, and D. Smith. Acceleration of convergence of vector sequences. *SIAM J. Numer. Anal.*, 23:178–196, 1986. Originally appeared as NASA TP-2193, (1983).

[31] A. Sidi and Y. Shapira. Upper bounds for convergence rates of vector extrapolation methods on linear systems with initial iterations. Technical Report 701, Computer Science Department, Technion–Israel Institute of Technology, 1991. Appeared also as NASA Technical memorandum 105608, ICOMP-92-09, (1992).

[32] A. Sidi and Y. Shapira. Upper bounds for convergence rates of acceleration methods with initial iterations. *Numer. Algorithms*, 18:113–132, 1998.

[33] S. Skelboe. Computation of the periodic steady-state response of nonlinear networks by extrapolation methods. *IEEE Trans. Circuits and Systems*, 27:161–175, 1980.

[34] D. Smith, W. Ford, and A. Sidi. Extrapolation methods for vector sequences. *SIAM Rev.*, 29:199–233, 1987.

[35] S. M. Smith and J. Brady. SUSAN - a new approach to low level image processing. *International Journal of Computer Vision*, 23:45–78, 1997.

[36] N. Sochen, R. Deriche, and L. Lopez Perez. The Beltrami flow over implicit manifolds. In *IEEE International Conference on Computer Vision*, pages 832–839, 2003.

[37] N. Sochen, R. Kimmel, and A. M. Bruckstein. Diffusions and confusions in signal and image processing. *Journal of Mathematical Imaging and Vision*, 14(3):195–209, 2001.

[38] N. Sochen, R. Kimmel, and R. Maladi. From high energy physics to low level vision. In *Proceedings of the First International Conference on Scale Space Theory in Computer Vision*, volume 1252 of *Lecture Notes on Computer Science*, pages 237–247, July 1997.

[39] N. Sochen, R. Kimmel, and R. Maladi. A general framework for low level vision. *IEEE Trans. Image Process.*, 7(3):310–318, 1998.

[40] A. Spira, R. Kimmel, and N. Sochen. Efficient Beltrami flow using a short time kernel. *IEEE Trans. Image Process.*, 16:1628–1636, 2007.

[41] B. Tang, G. Sapiro, and V. Caselles. Diffusion of general data on non-flat manifolds via harmonic maps theory: The direction diffusion case. *International Journal of Computer*

Vision, 36(2):149–161, 2000.

[42] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *IEEE International Conference on Computer Vision*, pages 836–846, 1998.

[43] J. Weickert. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE Trans. Image Process.*, 7:398–410, 1998.

[44] A. J. Yezzi. Modified curvature motion for image smoothing and enhancement. *IEEE Trans. Image Process.*, 7(3):345–352, 1998.