Applications of Vector Extrapolation Methods in Image Processing and Dimensionality Reduction

Guy Rosman

April 29, 2008

Contents

1 Introduction **3**

2 Extrapolation Methods **1**
2.1 Derivation of MPE 3
2.2 Derivation of RRE 4
2.3 Treatment of Nonlinear Equations 5
2.4 Efficient Implementation of MPE and RRE 5
2.5 Error Estimation 7
2.6 Error Analysis for MPE and RRE 8
2.7 Cycling with MPE and RRE 9
2.8 Connection with Krylov Subspace Methods 10
2.9 Cycling with a safeguard 11
2.10 An Alternative Motivation for the MPE and RRE methods . . . 11
2.11 Relation to Other Extrapolation Methods 14

3 Multidimensional Scaling **16**
3.1 Multidimensional Scaling 17
3.1.1 Least-squares MDS 17
3.1.2 SMACOF algorithm 18
3.1.3 Classical Scaling 18
3.2 Results 19
3.2.1 Positioning of Random Points 20
3.2.2 A Euclidean Dimensionality Reduction Example 20
3.2.3 Graph Visualization Using Accelerated MDS 20
3.2.4 Isometry Invariant Canonical Forms 22
3.2.5 TCIE using Accelerated MDS 22

4 Topologically Constrained Isometric Embedding **26**
4.1 Topologically Constrained Isometric Embedding 29
4.1.1 Detection of Boundary Points 31
4.1.2 Detection of Inconsistent Geodesics 33
4.1.3 Weighted LS-MDS 35
4.2 Implementation Considerations 35
4.2.1 Numerical Properties and Convergence 36
4.2.2 Convergence Acceleration by Multiscale Optimization . . 36
4.3 Results 37

5 Efficient Beltrami Filtering using Vector Extrapolation **44**
5.1 The Beltrami Framework 45
5.2 Standard Explicit Finite Difference Scheme 47
5.3 Experimental Results 48
5.3.1 Beltrami-based Denoising 49
5.3.2 Beltrami-based Deblurring 50

6 Conclusions **52**

List of Figures

3.1 The visual results using SMACOF and RRE-accelerated SMACOF, obtained for the 1138Bus graph 21
3.2 Left: A facial surface used as input to the canonical forms algorithm. Right: The resulting canonical form 22
3.3 Stress, as a function of CPU time for the canonical forms problem. CPU time is approximated. The second sub-figure is a close-up of the first sub-figure. 23
3.4 Left: Swiss roll surface with a hole in it. Detected boundary points are shown in red. Right: The mapping obtained using the TCIE algorithm 24
3.5 Convergence (in terms of stress value) of basic SMACOF (top, dashed gray), SMACOF with RRE (top, black), SMACOF with multiresolution acceleration (bottom, dashed gray), and SMACOF with both RRE and multiscale (bottom, black), in terms of CPU time, as part of the TCIE algorithm. CPU time is approximated. Convergence at each scale was stopped at the same relative change of stress value. 25

4.1 Left to right: A sampled surface (1815 samples) and the results of its dimensionality reduction into $\mathbb{R}^2$ using PCA, Isomap and the proposed TCIE algorithm. 27
4.2 Example of two points $\mathbf{x}_1$ and $\mathbf{x}_2$, for which the line connecting the points $\mathbb{R}^m$ (dashed black line) is shorter than the geodesic $g_{\mathcal{C}}(\mathbf{x}_1, \mathbf{x}_2)$ (solid black curve), even after flattening due to non-convexity. 30
4.3 An example of two neighborhoods of points in $\mathcal{M}$, one of which is close to the boundary $\partial\mathcal{M}$. In the example, $N = 20000$, and each neighborhood was chosen to include 500 nearest neighbors of a point on the manifold. 32
4.4 Example of a local minimizer of the weighted MDS problem. Flips along the boundaries of the rectangle are marked with arrows. 36
4.5 Left to right: Swiss roll surface without noise, contaminated by Gaussian noise with $\sigma = 0.015$ and $\sigma = 0.05$. Detected boundary points are shown as red crosses. 38
4.6 A planar surface cut as a spiral. Detected boundary points are shown as red crosses. 39

4.7 Embedding of the Swiss roll (without noise), as produced by LLE, Laplacian eigenmaps, Hessian LLE, diffusion maps, Isomap, and our algorithm. Detected boundary points are shown as red crosses. 40
4.8 Embedding of the Swiss roll contaminated by Gaussian noise with $\sigma = 0.015$, as produced by LLE, Laplacian eigenmaps, Hessian LLE, diffusion maps, Isomap, and our algorithm. Detected boundary points are shown as red crosses. 41
4.9 Embedding of a 2D manifold contaminated by Gaussian noise with $\sigma = 0.05$, as produced by LLE, Laplacian eigenmaps, Hessian LLE, diffusion maps, Isomap, and our algorithm. Detected boundary points are shown as red crosses. 42
4.10 Embedding of the spiral manifold, as produced by LLE, Laplacian eigenmaps, Hessian LLE, diffusion maps, Isomap, and our algorithm. Detected boundary points are shown as red crosses. . 43
4.11 Representation of the gaze direction manifold uncovered by our algorithm. 43

5.1 Top (left to right): RRE iterations: $50, 150, 300, 450$. Bottom: left: Original picture. right: Comparison of the residual norms versus CPU time. Parameters: $\beta = \sqrt{1/0.0005} \simeq 44$, $\Delta t = 0.0042/\beta^2$. 49
5.2 l^2-norm error between the RRE iterations sequence and the corresponding explicit iterations images, in the case of scale-space evolution, shown in Figure 5.1. A sequence using $k = 15$ is shown, which gave the worst l^2-norm values in practice. Parameters: $\beta = \sqrt{1/0.001} \simeq 31.62$, $\Delta t = 0.0042/\beta^2$. 49
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: $\beta = \sqrt{1000}$, $\lambda = 0.02$, $\Delta t = 0.0021/\beta^2$. 50
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: $\beta = \sqrt{1/0.0005} \simeq 44$, $\Delta t = 0.0021/\beta^2$, $\lambda = 0.03$. 51

6.1 An illustrations of points x, x', x'' used in the proof of Proposition 1 55

List of Tables

2.1 Definition of the minimal polynomial extrapolation method (MPE) 4
2.2 Definition of the reduced rank extrapolation method (RRE) . . 4
2.3 The modified Gram-Schmidt algorithm 6
2.4 An efficient implementation of MPE and RRE 7
2.5 Using vector extrapolation techniques in cycling mode 10
2.6 Using vector extrapolation techniques for accelerating the SMA-COF algorithm 11

3.1 The resulting speedup, in terms of the rate of decrease in residual norm, obtained using RRE with $n = 5$, $k = 5$ 20
3.2 The resulting speedup, in terms of CPU time, obtained using RRE with $n = 8$, $k = 10$, for the 2-classes Euclidean dimensionality reduction problem 20
3.3 The resulting speed-up, in terms of CPU time, obtained using RRE for the 1138Bus graph, using various n and k parameters . 21

Abstract

Multidimensional scaling attempts to map the coordinates of a set of points given their pairwise distances in a simple low dimensional space. It is used in various applications, ranging from dimensionality reduction of image manifolds to psychology and statistics.

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. Its computation is, however, computationally costly, which is sometimes problematic for image processing applications.

In this thesis we use vector extrapolation techniques in order to accelerate the numerical solution of the multidimensional scaling problem, and efficiently implement the application of the Beltrami filter. Vector extrapolation methods are techniques which are used to accelerate the convergence of fixed-point iterative algorithms. We first review vector extrapolation techniques, detailing the theory available in the case of linear iterative processes, and in the proximity of the fixed point of nonlinear processes. We then review the problem of multidimensional scaling, and the Beltrami framework for image processing. We show several examples of our accelerated solver for multidimensional scaling problems in various applications, and give several examples of accelerating the computation of the Beltrami flow using vector extrapolation.

As one application of multidimensional scaling, we develop a new algorithm for nonlinear dimensionality reduction. Our algorithm uses both local and global distances in order to learn the intrinsic geometry of locally flat manifolds even with non-trivial topology. Unlike existing algorithms that use global distances, we use the topology of the manifold in order to avoid using distance measurements that will distort the resulting mapping. The main idea is to filter out potentially problematic distances between distant feature points based on the properties of the geodesics connecting those points and their relative distance to the edge of the feature manifold. Since the proposed algorithm matches non-local structures, it is robust to a large amount of noise. The algorithm uses our accelerated multidimensional scaling solver in order to compute its mapping. We show experimental results demonstrating the advantages of the proposed approach over state-of-the-art methods.

List of Symbols

$\mathbf{R}^n$	n-dimensional Euclidean space
$\mathcal{M}$	A manifold representing the data
$\mathcal{C}$	The parametrization of the manifold in a lower dimensional space
m	The intrinsic dimension of manifold $\mathcal{M}$
M	The dimension of the embedding space for manifold $\mathcal{M}$
$\mathbf{z}, \mathbf{z}_i$	A vector, or a point in the input data set, sampled from the data manifold $\mathcal{M}$
$\mathbf{x}, \mathbf{x}_i$	A vector, or a point in the data set representation obtained by MDS. In the development of extrapolation methods (Chapter 2) $\mathbf{x}$ denotes the representation of the problem in vectorial form
$\mathbf{X}$	The representation of a configuration of points as a matrix
$d_{\mathcal{C}}(\cdot,\cdot)$	The distance measure between points on $\mathcal{C}$
$c_{\mathcal{C}}(\cdot,\cdot)$	The geodesic curve associated with the distance between points on $\mathcal{C}$
$d_{R^m\|\mathcal{C}}(\cdot,\cdot)$	The distance measure induced from R^m for points on $\mathcal{C}$
$\mathcal{N}(i)$	The neighboring points of point i
$\|\mathcal{N}(i)\|$	The cardinality of the sampled neighborhood of point i
d_{ij}	A distance between points i and j
$d_{ij}(\mathbf{X})$	A distance between points i and j, where i, j are points parameterized by $\mathbf{X}$
δ_{ij}	A distance between points i and j approximated by the MDS algorithm. In most case specifies a geodesic distance
$f(\cdot,\cdot)$	An error cost function between two scalar values
$s(\mathbf{X})$	The stress error resulting from a point configuration given by $\mathbf{X}$
$\mathbf{B}_\Delta$	The gram matrix associated with point configuration $\mathbf{X}$
$B_\epsilon(\mathbf{x})$	A ball of radius ϵ around point $\mathbf{x}$
P	The set of all point pairs with consistent geodesics linking the two points
$\bar{P}_1$	A subset of P defined using criterion 1
$\bar{P}_2$	A subset of P defined using criterion 2
$\mathbf{u}_n$	The difference between iteration vectors n and $n+1$
$\mathbf{U}$	A matrix whose columns are the differences of vector representation of iterative solutions of the problem at hand
$\mathbf{Q}, \mathbf{R}$	The matrices resulting from the QR decomposition of a matrix
$\mathbf{A}, \mathbf{b}$	The matrix and vector characterizing a linear vectorial iterative process
$\gamma_i, \quad i = 1, \ldots, n$	A set of weighting coefficients used to combine vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$
$\rho(\mathbf{A})$	The spectral radius of matrix $\mathbf{A}$
$\mathbf{I}$	The identity matrix
$\mathbf{F}(\cdot)$	A vectorial, nonlinear, process
$\mathbf{s}$	The fixed point vector of a converging vectorial process

$\boldsymbol{\epsilon}_i$	The i-th error vector of a vectorial process
λ_i	The i-th eigenvector of a matrix
$D()$	A determinant expression resulting from Cramer's rule
$\mathbf{H}()$	An ordinary Hankel determinant
$\tilde{\mathbf{H}}()$	A generalized Hankel determinant
(Σ, g)	The image manifold and its metric
(M, h)	The spatial-feature manifold
σ^i	The ith intrinsic coordinate of the image manifold
g_{ij}	The i, j component of the metric g
h_{ab}	The a, b component of the metric h
G	The matrix representing the metric g
X	A map from Σ into M
$S[X, g_{ij}, h_{ab}]$	A functional of the map X
K	A bounded linear operator on the space of images
k	A convolution kernel
k^n_{ij}	A convolution kernel at (i, j) position, at time step n
$\bar{k}$	A convolution kernel flipped with respect to the axes
I^a	The a channel component of a color image
$(I^a)_{ij}$	Pixel i, j of the image channel I^a
$L_{ij}(I^a)$	The discretized Laplace-Beltrami operator acting on channel I^a of image I
$(I^a)^n_{ij}$	Pixel i, j of the image channel I^a at iteration n
β	A parameter setting the relative importance of the spatial vs the intensity term in the image manifold and its metric

Chapter 1

Introduction

Vector extrapolation techniques are a collection of methods which attempt to accelerate the convergence of iterative processes operating on high dimensional data. In this thesis we focus on two main applications of vector extrapolation techniques for computer vision.

The first application is in the field of *dimensionality reduction*. Dimensionality reduction methods such as *principal components analysis* (PCA) [40] and *multidimensional scaling* (MDS) [13] are often used to obtain a low dimensional representation of the data. This is a commonly used pre-processing stage in pattern recognition.

While methods such as PCA assume the existence of a linear map between the data points and the parametrization space, such a map often does not exist. Applying linear dimensionality reduction methods to nonlinear data may result in a distorted representation. *Nonlinear dimensionality reduction* (NLDR) methods attempt to describe a given high-dimensional data set of points as a low dimensional manifold, by a nonlinear map preserving certain properties of the data.

We present a new method for nonlinear dimensionality reduction, *topologically constrained isometric embedding* (TCIE), which relies on MDS when constructing a coordinates system which minimizes the distortion of the representation. The algorithm is different from most existing methods for nonlinear dimensionality reduction in that is uses the topology of the manifold to detect which geodesics can be used to construct a mapping of the data. It then uses this information to combine local and global geodesic length measurements. We use vector extrapolation techniques in order to accelerate the convergence of the MDS algorithm used in the embedding.

We demonstrate the acceleration of MDS computation in various applications in Chapter 3. We then describe in detail the TCIE algorithm and the assumptions it is based on in Chapter 4.

The second application we explore is PDE-based image filtering using the *Beltrami* flow. The Beltrami framework for image filtering regards the image as a manifold embedded in a combined feature/spatial space, and uses the minimization of the surface as a regularization term for image denoising [106], deblurring [3], scale-space analysis and so forth [8].

While this flow has a sound motivation and has been shown to achieve remarkable results, due to its nonlinear and non-separable nature it is usually

computed using explicit finite difference schemes, which tend to be quite slow to compute. This limitation remained despite attempts to alleviate this problem [108]. We show in Chapter 5 how vector extrapolation can accelerate the convergence of the solution to the PDE representing the Beltrami flow, in several model examples in which the Beltrami flow is often used. Chapter 6 concludes this thesis, and highlights the main results.

Chapter 2

Extrapolation Methods

Looking at the patterns of changes exhibited by several nonlinear processes, such as the SMACOF iterations (described in Chapter 3) and the Beltrami flow (described in Chapter 5), it seems that the general direction of the convergence process can be inferred from past iterates. This hints at the possibility of exploiting the behavior of the iterates to speed up the convergence. One possible way to predict the limit of the convergence is by using methods of vector extrapolation, such as those presented in the following section.

Specifically, we focus on two methods found especially useful by us. The minimal polynomial extrapolation (MPE) [21] and the reduced rank extrapolation (RRE) [72, 41] are two vector extrapolation methods that have proved to be very efficient in accelerating the convergence of vector sequences that arise 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 1980s, see [103]. The brief review we present here covers the various developments that have taken place since the publication of [103].

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

$$\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n + \mathbf{b}, \quad n = 0, 1, \ldots, \tag{2.1}$$

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

$$\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{b}, \tag{2.2}$$

provided $\rho(\mathbf{A}) < 1$, where $\rho(\mathbf{A})$ is the spectral radius of $\mathbf{A}$. Note that the system in (2.2) 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 by [103]. Other derivations, based on the Shanks–Schmidt transformation [90, 88] have been given by [98].

Given the sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots$, generated as in (2.20), 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

$$\boldsymbol{\epsilon}_n = \mathbf{x}_n - \mathbf{s}, \quad n = 0, 1, \dots \ . \tag{2.3}$$

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

$$\boldsymbol{\epsilon}_n = (\mathbf{A}\mathbf{x}_{n-1} + \mathbf{b}) - (\mathbf{A}\mathbf{s} + \mathbf{b}) = \mathbf{A}(\mathbf{x}_{n-1} - \mathbf{s}) = \mathbf{A}\boldsymbol{\epsilon}_{n-1}, \tag{2.4}$$

which, by induction, gives

$$\boldsymbol{\epsilon}_n = \mathbf{A}^n \boldsymbol{\epsilon}_0, \quad n = 0, 1, \dots \ . \tag{2.5}$$

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

$$\mathbf{s}_{n,k} = \sum_{i=0}^{k} \gamma_i \mathbf{x}_{n+i}; \quad \sum_{i=0}^{k} \gamma_i = 1. \tag{2.6}$$

Substituting Equation (2.21) in Equation (2.23), and making use of the fact that $\sum_{i=0}^{k} \gamma_i = 1$, we obtain

$$\mathbf{s}_{n,k} = \sum_{i=0}^{k} \gamma_i (\mathbf{s} + \boldsymbol{\epsilon}_{n+i}) = \mathbf{s} + \sum_{i=0}^{k} \gamma_i \boldsymbol{\epsilon}_{n+i} \tag{2.7}$$

which, by (2.22), becomes

$$\mathbf{s}_{n,k} = \mathbf{s} + \sum_{i=0}^{k} \gamma_i \mathbf{A}^{n+i} \boldsymbol{\epsilon}_0. \tag{2.8}$$

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 (2.22), this means that

$$\sum_{i=0}^{k} c_i \mathbf{A}^i \boldsymbol{\epsilon}_n = \sum_{i=0}^{k} c_i \boldsymbol{\epsilon}_{n+i} = \mathbf{0}. \tag{2.9}$$

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 (2.9) by $\mathbf{A}$, and recalling (2.4), 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 (2.9), 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

$$\sum_{i=0}^{k} c_i \mathbf{u}_{n+i} = \mathbf{0}. \tag{2.10}$$

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

$$\mathbf{U}_s^{(j)} = [\, \mathbf{u}_j \,|\, \mathbf{u}_{j+1} \,|\, \ldots \,|\, \mathbf{u}_{j+s} \,]. \tag{2.11}$$

Thus, $\mathbf{U}_s^{(j)}$ is an $N \times (j+1)$ matrix. In this notation, Equation (2.9) reads

$$\mathbf{U}_k^{(n)} \mathbf{c} = \mathbf{0}; \quad \mathbf{c} = [c_0, c_1, \ldots, c_k]^{\mathrm{T}}. \tag{2.12}$$

Of course, dividing Equation (2.12) by $\sum_{i=0}^{k} c_i$, we also have

$$\mathbf{U}_k^{(n)} \boldsymbol{\gamma} = \mathbf{0}; \quad \boldsymbol{\gamma} = [\gamma_0, \gamma_1 \ldots, \gamma_k]^{\mathrm{T}}. \tag{2.13}$$

2.1 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

1	Choose the integers k and n, and input the vectors $\mathbf{x}_n, \mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+k}$.
2	Form the $N \times k+1$ 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 $\mathbf{c}' = [c_0, c_1, \ldots, c_{k-1}]^{\mathrm{T}}$.
4	Set $c_k = 1$, and $\gamma_i = c_i / \sum_{i=0}^{k} c_i$, $i = 0, 1, \ldots, k$.
5	Compute the vector $\mathbf{s}_{n,k} = \sum_{i=0}^{k} \gamma_i \mathbf{x}_{n+i}$ as approximation to $\lim_{i\to\infty} \mathbf{x}_i = \mathbf{s}$.

Table 2.1: Definition of the minimal polynomial extrapolation method (MPE)

1	Choose the integers k and n, and input the vectors $\mathbf{x}_n, \mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+k}$.
2	Form the $N \times k+1$ matrix $\mathbf{U}_k^{(n)}$.
3	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 $\lim_{i\to\infty} \mathbf{x}_i = \mathbf{s}$.

Table 2.2: Definition of the reduced rank extrapolation method (RRE)

polynomial of $\mathbf{A}$ with respect to $\boldsymbol{\epsilon}_n$. With this k, the linear system in Equation (2.10) 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 (2.10), 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 method (MPE). 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 summarize the definition of MPE in Table 2.1.

2.2 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 (2.13) 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 method (RRE). 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 summarize the definition of RRE in Table 2.2.

2.3 Treatment of Nonlinear Equations

We now turn to the treatment of nonlinear equations, such as those accelerated in Chapters 3 and 5, by vector extrapolation methods. Assume that the system of nonlinear equations in question has been written in the (possibly preconditioned) form

$$\mathbf{x} = \mathbf{F}(\mathbf{x}), \tag{2.14}$$

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

$$\mathbf{x}_{n+1} = \mathbf{F}(\mathbf{x}_n), \quad n = 0, 1, \ldots, \tag{2.15}$$

and assume that this sequence converges to the solution vector $\mathbf{s}$. In our case, $\mathbf{F}$ can be the right-hand side of the SMACOF iteration (given in a general form in Equation (3.4)), or an iteration of the explicit scheme for the Beltrami flow. 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

$$\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n + \mathbf{b}, \quad n = 0, 1, \ldots, \tag{2.16}$$

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 (2.20)] 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.

2.4 Efficient Implementation of MPE and RRE

In sections 2.1 and 2.2, we gave the definitions of MPE and RRE. These definitions actually form the basis for efficient implementations of MPE and RRE

```
1 Compute r_00 = ‖u_n‖ and q_0 = u_n/r_00
2 for i = 1, ..., k do
3     Set u_i^(0) = u_{n+i}  for j = 0, ..., i − 1 do
4         r_jk = q_j^* u_i^(j) and u_i^(j+1) = u_i^(j) − r_jk q_j
5     end
6     Compute r_ii = ‖u_i^(i)‖ and q_i = u_i^(i)/r_ii.
7 end
```

Table 2.3: The modified Gram-Schmidt algorithm

[94]. 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 by [94].

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

$$\mathbf{Q}_k = [\, \mathbf{q}_0 \,|\, \mathbf{q}_1 \,|\, \cdots \,|\, \mathbf{q}_k \,], \tag{2.17}$$

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,

$$\mathbf{R}_k = \begin{bmatrix} r_{00} & r_{01} & \cdots & r_{0k} \\ & r_{11} & \cdots & r_{1k} \\ & & \ddots & \vdots \\ & & & r_{kk} \end{bmatrix}; \quad r_{ii} > 0, \quad i = 0, 1, \ldots, k. \tag{2.18}$$

This factorization can be carried out easily and accurately using the modified Gram–Schmidt orthogonalization process (MGS), which is a standard numerical algorithm [49, 94]. For completeness, Table 2.3 describes the steps of MGS as applied to the matrix $\mathbf{U}_k^{(n)}$:

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 $(k+1)$st column. Similarly, $\mathbf{R}_k$ is obtained from $\mathbf{R}_{k-1}$ by appending to the latter the $\mathbf{0}$ vector as the $(k+1)$st row and then the vector $[r_{0k}, r_{1k}, \ldots, r_{kk}]^{\mathrm{T}}$ as the $(k+1)$st 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 the memory. The vectors $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+k+1}$ need not be saved.

Input: k and n and the vectors $\mathbf{x}_n, \mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+k+1}$.

1 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 (2.17) and (2.18).

2 Determination of the γ_i:

`// For MPE`

3 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`

4 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 $\gamma_i = \lambda d_i$, $i = 0, 1, \ldots, k$.

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

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

6 Compute

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

7 Then compute

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

Table 2.4: An efficient implementation of MPE and RRE

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 shown in Table 2.4.

2.5 Error 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.

For linear sequences: When the iteration vectors $\mathbf{x}_i$ are generated linearly as in (2.20), we have

$$\mathbf{r}(\mathbf{x}) = \mathbf{b} - (\mathbf{I} - \mathbf{A})\mathbf{x} = (\mathbf{A}\mathbf{x} + \mathbf{b}) - \mathbf{x}.$$

Thus,

$$\mathbf{r}(\mathbf{x}_n) = \mathbf{x}_{n+1} - \mathbf{x}_n = \mathbf{u}_n.$$

Invoking also $\sum_{i=0}^{k} \gamma_i = 1$, where γ_i are as obtained when applying MPE or

RRE, we therefore have

$$\mathbf{r}(\mathbf{s}_{n,k}) = \sum_{i=0}^{k} \gamma_i \mathbf{u}_{n+i} = \mathbf{U}_k^{(n)} \boldsymbol{\gamma}. \tag{2.19}$$

We actually look at the l_2-norm of this vector, namely,

$$\|\mathbf{r}(\mathbf{s}_{n,k})\| = \|\mathbf{U}_k^{(n)} \boldsymbol{\gamma}\|.$$

For nonlinear sequences: When the $\mathbf{x}_i$ are generated nonlinearly as in (2.15), with $\mathbf{s}_{n,k}$ close to $\mathbf{s}$, we have

$$\mathbf{r}(\mathbf{s}_{n,k}) = \mathbf{F}(\mathbf{s}_{n,k}) - \mathbf{s}_{n,k} \approx \mathbf{U}_k^{(n)} \boldsymbol{\gamma}.$$

Thus,

$$\|\mathbf{r}(\mathbf{s}_{n,k})\| \approx \|\mathbf{U}_k^{(n)} \boldsymbol{\gamma}\|.$$

Whether the vectors $\mathbf{x}_i$ are generated linearly or nonlinearly, $\|\mathbf{U}_k^{(n)} \boldsymbol{\gamma}\|$ can be computed and in terms of already computed quantities and at no cost, even without having to compute $\mathbf{s}_{n,k}$. Indeed, we have

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

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 section [94].

2.6 Error Analysis for MPE and RRE

The analysis of MPE and RRE, within the context of linear systems, has been carried out in the works [92, 96, 95, 99, 100]. This analysis sheds considerable light on what vector extrapolation methods can achieve when they are applied in conjunction with iterative methods arising from nonlinear systems as well as linear ones.

The following result was given by [92].

Theorem 1. *Assume that the vectors* $\mathbf{x}_n$ *satisfy*

$$\mathbf{x}_n = \mathbf{s} + \sum_{i=1}^{p} \mathbf{v}_i \lambda_i^n,$$

where the vectors $\mathbf{v}_i$ *are linearly independent, and the nonzero scalars* $\lambda_i \neq 1$ *are distinct, and ordered as in*

$$|\lambda_1| \geq |\lambda_2| \geq \cdots .$$

If also

$$|\lambda_k| > |\lambda_{k+1}|,$$

then, both for MPE and RRE, there holds

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

and

$$\lim_{n\to\infty} \sum_{i=0}^{k} \gamma_i^{(n,k)} z^i = \prod_{i=1}^{k} \frac{\lambda - \lambda_i}{1 - \lambda_i}.$$

Here $\gamma_i^{(n,k)}$ *stands for* γ_i.

Note that when the $\mathbf{x}_n$ are generated by the iterative procedure in (2.20), they are precisely as described in this theorem, with λ_i being some or all of the distinct nonzero eigenvalues of the iteration matrix $\mathbf{A}$ and $\mathbf{v}_i$ being eigenvectors corresponding to these eigenvalues, when $\mathbf{A}$ is diagonalizable.

When the condition $|\lambda_k| > |\lambda_{k+1}|$ does not hold Theorem 1 above needs to be modified somewhat. This has been done previously [95], and we state a special case of it next.

Theorem 2. *Assume that the vectors* $\mathbf{x}_n$ *have been generated by the iterative procedure* $\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n + \mathbf{b}$, *the matrix* $\mathbf{I} - \mathbf{A}$ *being nonsingular. Denote the solution of* $(\mathbf{I} - \mathbf{A})\mathbf{x} = \mathbf{b}$ *by* $\mathbf{s}$. *Let us order the nonzero distinct eigenvalues* λ_i *of* $\mathbf{A}$ *as in*

$$|\lambda_1| \geq |\lambda_2| \geq \cdots .$$

Then, whether $|\lambda_k| > |\lambda_{k+1}|$ *or not, there holds*

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

(i) for RRE unconditionally and (ii) for MPE provided the eigenvalues of $\mathbf{I} - \mathbf{A}$ *lie strictly on one side of a straight line through the origin in the complex plane, which happens when* $\mathbf{A} + \mathbf{A}^*$ *is positive definite, for example.*

In connection with Theorems 1 and 2, we note that we have not assumed that $\lim_{n\to\infty} \mathbf{x}_n$ exists. Clearly, $\lim_{n\to\infty} \mathbf{x}_n$ exists and is equal to $\mathbf{s}$ provided $|\lambda_1| < 1$. In case $\lim_{n\to\infty} \mathbf{x}_n$ does not exist, we have that $|\lambda_1| \geq 1$ necessarily.

Now, if we use the $\mathbf{x}_i$, as they are, to approximate $\mathbf{s}$, we have the error

$$\boldsymbol{\epsilon}_n = \mathbf{x}_n - \mathbf{s} = O(\lambda_1^n) \quad \text{as } n \to \infty.$$

Thus, provided $|\lambda_{k+1}| < 1$, we have that $\lim_{n\to\infty} \mathbf{s}_{n,k} = \mathbf{s}$, whether $\lim_{n\to\infty} \mathbf{x}_n$ exists or not. Furthermore, when the sequence $\mathbf{x}_n$ converges (to $\mathbf{s}$), $\mathbf{s}_{n,k}$ converges (to $\mathbf{s}$) faster, provided $|\lambda_{k+1}| < |\lambda_1|$. In words, MPE and RRE accelerate the convergence of the sequence $\{\mathbf{x}_n\}$.

2.7 Cycling with MPE and RRE

The results of Theorems 1 and 2 pertain to the convergence and acceleration of convergence of MPE and RRE. In these theorems, we keep k fixed and let n go to infinity. Obviously, there is no way of letting n go to infinity in practice. It also follows from Theorems 1 and 2 that increasing k generally makes MPE and RRE converge faster. However, we have no way of increasing k indefinitely, 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

1	Choose integers n, k, and an initial vector $\mathbf{x}_0$.
2	Compute the vectors $\mathbf{x}_i$, $1 \le i \le n+k+1$, [via $\mathbf{x}_{n+1} = \mathbf{F}(\mathbf{x}_n)$], and save $\mathbf{x}_{n+i}$, $0 \le i \le k+1$.
3	Apply MPE or RRE to the vectors $\mathbf{x}_{n+i}$, $0 \le i \le k+1$, precisely as described in section (2.4), with end result $\mathbf{s}_{n,k}$.
4	If $\mathbf{s}_{n,k}$ satisfies accuracy test, stop.
5	Otherwise, set $\mathbf{x}_0 = \mathbf{s}_{n,k}$, and go to Step 1.

Table 2.5: Using vector extrapolation techniques in cycling mode

of application called *cycling* or *restarting* in which n and k are held fixed. Here are the steps of this mode. We will call each application of Steps 1–3 a *cycle*. We will also denote the $\mathbf{s}_{n,k}$ that is computed in the ith cycle $\mathbf{s}_{n,k}^{(i)}$.

A discussion of the error in this mode of usage—in case of linear $\mathbf{F}(x)$, i.e., when $\mathbf{F}(x) = \mathbf{A}\mathbf{x}+\mathbf{b}$—was given previously [99, 100]. 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, $\mathbf{s}_{n,k}^{(i)}$ may be very good approximations to the solution $\mathbf{s}$ with small k, hence small storage 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 GMRES stagnates. (See the numerical examples in [99, 100]). 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 [99, 100] 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 works by [102, 103]. 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 $\{\mathbf{s}_{n,k_i}^{(i)}\}_{i=0}^{\infty}$ 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 $\mathbf{s}$.

2.8 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 [84], two well known Krylov subspace methods. The following result was stated by [93].

Theorem 3. *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*

1	Choose integers n, k, and an initial vector $\mathbf{x}_0$.
2	Perform n iterations of the SMACOF algorithm
3	Perform additional k iterations of the SMACOF algorithm, saving the iterants as $\mathbf{x}_{n+i}$, $0 \leq i \leq k$
4	Apply MPE or RRE to the vectors $\mathbf{x}_{n+i}$, $0 \leq i \leq k+1$, precisely as described in section (2.4), with end result $\mathbf{s}_{n,k}$.
5	If $stress\left(\mathbf{x}_{n+k}\right) \leq stress\left(\mathbf{s}_{n,k}\right)$, take $\mathbf{s}_{n,k}$ to be $\mathbf{x}_{n+k}$
6	If $\mathbf{s}_{n,k}$ satisfies accuracy test based on relative stress change, stop.
7	Otherwise, set $\mathbf{x}_0 = \mathbf{s}_{n,k}$, and go to Step 1.

Table 2.6: Using vector extrapolation techniques for accelerating the SMACOF algorithm

$(\mathbf{I} - \mathbf{A})\mathbf{x} = \mathbf{b}$, *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.

2.9 Cycling with a safeguard

In order to accelerate the convergence of the SMACOF algorithm and the Beltrami flow, we use cycling mode. Since the problems at hand are nonlinear, there is no guarantee that the approximate limit vector given by extrapolation will result in an improvement of our cost function, or an approximation of a future iteration vector from the iterative sequence. In the case of the SMACOF algorithm, we have to use a safeguard. One possibility is to check the stress value at the extrapolated limit, and if the resulting stress is higher, use the last iteration vector instead, as the initial solution to the next cycle. This is simply done by changing step 5 in Table 2.5. An algorithmic description is shown in Table 2.6. Another possibility, instead of a simple safeguard, is to use a diminishing step size, halving the step size several times before resorting to the last iteration vector.

In the case of the Beltrami flow and related processes, in practice there was no need to use this kind of precaution.

2.10 An Alternative Motivation for the MPE and RRE methods

We now present a slightly different derivation of the MPE and RRE method, more intuitive in the case of nonlinear iterations.

As before, we still assume that the iteration process can be approximated as a linear process of the form

$$\mathbf{x}_{i+1} = \mathbf{A}\mathbf{x}_i + \mathbf{b} \tag{2.20}$$

We assume that the matrix $\mathbf{A}$ has a spectral radius of less than 1 and thus the iteration (2.20) has a fixed point $\mathbf{s}$, which satisfies

$$\mathbf{s} = \mathbf{A}\mathbf{s} + \mathbf{b}$$

Again, we do not assume knowledge of the matrix $\mathbf{A}$, or of the vector $\mathbf{b}$. We are interested in a good approximation of $\mathbf{s}$, using a relatively small number of vectors $\{\mathbf{x}_i\}_{i=0}^{k}, k \ll N$ and without inverting an $N \times N$ matrix. For this purpose, let us define the error of a vector sequence element. It is defined as:

$$\epsilon_i = \mathbf{s} - \mathbf{x}_i. \tag{2.21}$$

If the vector sequence is generated through a linear iterative process, one can relate the error in step i to the initial error

$$\epsilon_i = \mathbf{s} - \mathbf{x}_i = \mathbf{A}\mathbf{s} + \mathbf{b} - \mathbf{A}\mathbf{x}_{i-1} - \mathbf{b} = \mathbf{A}^i\epsilon_0.$$

Thus

$$\mathbf{x}_i = \mathbf{s} - \mathbf{A}^i\epsilon_0. \tag{2.22}$$

We further restrict our approximated solution to be a linear combination of the sequence elements:

$$\mathbf{s} = \sum_{i=0}^{k} \gamma_i \mathbf{x}_i \tag{2.23}$$

Substituting Equation (2.22) into Equation (2.23), we obtain

$$\mathbf{s} = \sum_{i=0}^{k} \gamma_i \mathbf{x}_i = \sum_{i=0}^{k} \gamma_i(\mathbf{s} - \mathbf{A}^i\epsilon_0) = \mathbf{s}\sum_{i=0}^{k} \gamma_i - \sum_{i=0}^{k} \gamma_i \mathbf{A}^i\boldsymbol{\epsilon}_0 \tag{2.24}$$

From this expression one easily sees two requirements made of the set of coefficients: Since the equality in Equation (2.24) can be multiplied by any arbitrary scalar, including 0, we would like to remove this excess degree of freedom. We therefore force the sum of the coefficients $\{\gamma_i\}$ to be equal to 1. Also, using Equation (2.24), we expect our coefficients to nullify the weighted sum of the error vectors. The polynomial in $\sum_{i=0}^{k} \gamma_i \mathbf{A}^i\boldsymbol{\epsilon}_0$ is known as the minimal polynomial with respect to $\boldsymbol{\epsilon}_0$, hence the name of MPE method. That is, our approximation of the limit to the sequence becomes

$$\mathbf{s}_{n,k} = \sum_{i=0}^{k} \gamma_i \mathbf{x}_i \tag{2.25}$$

and the coefficients γ_i are found as a solution to

$$\text{Find} \quad \{\gamma_i\}_{i=1}^{k} \quad \text{s.t.} \quad \sum_{i=0}^{k} \gamma_i = 1, \quad \text{and} \quad \sum_{i=0}^{k} \gamma_i \mathbf{A}^i\boldsymbol{\epsilon}_0 = 0. \tag{2.26}$$

However, there is a problem solving (2.25), (2.26). One does not know ϵ_0 and also we do not know the matrix $\mathbf{A}$. We now show that the minimal polynomial can be replaced by a term which is based on subsequent elements of the given vector sequence, i.e. the *residual*, defined in the linear case as the difference

$\text{res}(\mathbf{x}_i) = \mathbf{x}_{i+1} - \mathbf{x}_i = \mathbf{A}\mathbf{x}_i + \mathbf{b} - \mathbf{x}_i$. Since the minimal polynomial should cancel both $\boldsymbol{\epsilon}_0$ and $\boldsymbol{\epsilon}_1$, the difference between them should also cancel. Using the fact that $\sum_{j=0}^{k} \gamma_j = 1$, we then have

$$\begin{aligned}
& \sum_{j=0}^{k} \gamma_j A^j \epsilon_1 - \sum_{j=0}^{k} \gamma_j A^j \epsilon_0 \\
= & \sum_{j=0}^{k} \gamma_j A^j (\mathbf{x}_1 - \mathbf{s}) - \sum_{j=0}^{k} \gamma_j A^j (\mathbf{x}_0 - \mathbf{s}) \\
= & \sum_{j=0}^{k} \gamma_j A^j \mathbf{x}_1 - \sum_{j=0}^{k} \gamma_j A^j \mathbf{x}_0 \\
= & \sum_{j=0}^{k} \gamma_j A^j \mathbf{x}_1 + \sum_{j=0}^{k} \gamma_j \sum_{i=0}^{j-1} A^i \mathbf{b} - (\sum_{j=0}^{k} \gamma_j A^j \mathbf{x}_0 + \sum_{j=0}^{k} \gamma_j \sum_{i=0}^{j-1} A^i \mathbf{b}) \\
= & \sum_{j=0}^{k} \gamma_j \Delta \mathbf{x}_j
\end{aligned}$$

Now the term $\sum_{j=0}^{k} \gamma_j \Delta \mathbf{x}_j$ uses only known elements of the sequence. Minimizing the norm of this term can also be seen as seeking coefficients γ_j which minimize the residual norm (under some norm) $\min ||A\mathbf{s}_{n,k} + b - \mathbf{s}_{n,k}||$, as seen by

$$\begin{aligned}
& \|(\mathbf{A}\mathbf{s}_{n,k} + \mathbf{b} - \mathbf{s}_{n,k}\| \qquad (2.27) \\
= & \|(\mathbf{A} \sum_{j=0}^{k} \gamma_j \mathbf{x}_{n+j} + \mathbf{b} - \sum_{j=0}^{k} \gamma_j \mathbf{x}_{n+j}\| \\
= & \| \sum_{j=0}^{k} \gamma_j \left[\mathbf{A}\mathbf{x}_{n+j} + \mathbf{b} - \mathbf{x}_{n+j}\right] \| \\
= & \| \sum_{j=0}^{k} \gamma_j \left[\mathbf{x}_{n+j+1} - \mathbf{x}_{n+j}\right] \| \\
= & \|\mathbf{U}_k^{(n)} \boldsymbol{\gamma}\|
\end{aligned}$$

where $\boldsymbol{\gamma}$ is the coefficients vector with elements $(\boldsymbol{\gamma})_j = \gamma_j$, $j = 0, 1, ..., k$ and for a fixed integer k the matrix $\mathbf{U}_k^{(n)}$ denotes the matrix whose columns are the difference vectors $(\mathbf{U}_k^{(n)})_j = \Delta \mathbf{x}_{n+j}$, $j = 0, 1, ..., k$.

Since we assumed $\sum_{j=0}^{k} \gamma_j = 1$, the method must enforce it. Two different constraints on the coefficients $\{\gamma_j\}$, used in finding the optimal solution, result in the MPE and RRE algorithms.

Constraining the last coefficient to equal 1, and then normalizing the coefficients so that $\sum_{j=0}^{k} \gamma_j = 1$, Equation (2.27) still holds. This gives us the MPE

method. Imposing
$\sum_{j=0}^{k} \gamma_j = 1$ directly results in the RRE method. Both methods can be implemented by solving a system of linear equations, as shown hereafter.

2.11 Relation to Other Extrapolation Methods

We now set the two methods we have used, the MPE and RRE methods, in the more general context of vector extrapolation methods. To do so, one can characterize the extrapolation problem solved by these methods in a manner similar to the Shanks-Schmidt transformation [87, 91]. Shanks started with the scalar sequence $\{x_m\}$ of the form

$$x_m = s + \sum_{i=1}^{k} a_i \lambda_i^m$$

where s, a_i, λ_i are scalar constants. Shanks solved for the set of equations

$$x_m = s + \sum_{i=1}^{k} a_i \lambda_i^m, \quad n \leq m \leq n+k,$$

resulting in the following expression for the solution

$$s = \frac{D(x_n, x_{n+1}, \ldots, x_{n+k})}{D(1, 1, \ldots, 1)}$$

Where the determinant expressions $D(x_n, x_{n+1}, \ldots, x_{n+k}), D(1, 1, \ldots, 1)$ are the result of Cramer's rule

$$D(\sigma_0, \ldots, \sigma_k) = \begin{vmatrix} \sigma_0 & \sigma_1 & \ldots & \sigma_k \\ \Delta x_n & \Delta x_{n+1} & \ldots & \Delta x_{n+k} \\ \vdots & \vdots & \vdots & \vdots \\ \Delta x_{n+k-1} & \Delta x_{n+k} & \ldots & \Delta x_{n+2k-1} \end{vmatrix}$$

In general, however, the determinant expression can also be the solution of a set of vector equations. In the vector case, however, the problem may already be overconstrained, and furthermore, the usage of more vectors than the minimum required may be computationally expensive.

We therefore allow any set of k equations relating consecutive elements of the sequence, or projections thereof, unto linearly independent vectors. The resulting determinants will have the form

$$D(\sigma_0, \ldots, \sigma_k) = \begin{vmatrix} \sigma_0 & \sigma_1 & \ldots & \sigma_k \\ \langle \Delta\tilde{\mathbf{x}}_{0,0}, \mathbf{y}_1 \rangle & \langle \Delta\tilde{\mathbf{x}}_{0,1}, \mathbf{y}_1 \rangle & \ldots & \langle \Delta\tilde{\mathbf{x}}_{0,k}, \mathbf{y}_1 \rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle \Delta\tilde{\mathbf{x}}_{k-1,0}, \mathbf{y}_{k-1} \rangle & \langle \Delta\tilde{\mathbf{x}}_{k-1,1}, \mathbf{y}_{k-1} \rangle & \ldots & \langle \Delta\tilde{\mathbf{x}}_{k-1,k}, \mathbf{y}_{k-1} \rangle \end{vmatrix}$$

Choosing $\Delta\tilde{\mathbf{x}}_{i,j}$ to be $\Delta\mathbf{x}_{n+j}$ and $\mathbf{y}_i$ to be $\Delta\mathbf{x}_i, \Delta^2\mathbf{x}_i$ or $\mathbf{e}_i$ (the ith unit vector) would yield the MPE and RRE, and another method called Modified MPE (MMPE), respectively [98, 97].

Another choice for $\Delta\tilde{\mathbf{x}}_{i,j}$ and $\mathbf{y}_i$ could be a set of consecutive equations: $\Delta\tilde{\mathbf{x}}_{i,j} = \Delta\mathbf{x}_{i+j}$, and a constant vector $\mathbf{y}$. These determinants are known as the generalized Hankel determinants [16].

$$\tilde{\mathbf{H}}^n_{k+1}(\mathbf{x}_n) = \begin{vmatrix} \mathbf{x}_n & \mathbf{x}_{n+1} & \dots & \mathbf{x}_{n+k} \\ \langle \Delta\mathbf{x}_n, \mathbf{y} \rangle & \langle \Delta\mathbf{x}_{n+1}, \mathbf{y} \rangle & \dots & \langle \Delta\mathbf{x}_{n+k}, \mathbf{y} \rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle \Delta\mathbf{x}_{n+k-1}, \mathbf{y} \rangle & \langle \Delta\mathbf{x}_{n+k+1}, \mathbf{y} \rangle & \dots & \langle \Delta\mathbf{x}_{n+2k-1}, \mathbf{y} \rangle \end{vmatrix}$$

Solving the set of consecutive equations, and rearranging the determinant expressions, one arrives at the following form

$$\mathbf{s} = \tilde{\mathbf{H}}^{(n)}_{k+1}(\mathbf{x}_n) / \mathbf{H}^{(n)}_{k+1}(\langle \mathbf{y}, \Delta^2\mathbf{x}_n \rangle)$$

where $\mathbf{H}^{(n)}_{k+1}(x_n)$ is the ordinary Hankel determinant

$$\mathbf{H}^n_{k+1}(x_n) = \begin{vmatrix} x_n & x_{n+1} & \dots & x_{n+k} \\ x_{n+1} & x_{n+2} & \dots & x_{n+k+1} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n+k} & x_{n+k+1} & \dots & x_{n+2k} \end{vmatrix}$$

This would result in the Topological Epsilon Algorithm [16, 86].

Simply looking at each component of the vectors separately, one obtains the Scalar Epsilon Algorithm [118]. Finally, one can obtain the Vector Epsilon Algorithm [118], by generalizing the determinants as designants [85].

The algorithms of the epsilon type have the property that calculation of the limit can be done gradually, in a recursive manner. They require, however, a larger set of iteration vectors and do not result in general in a higher rate of convergence. We therefore chose to focus in this work on polynomial extrapolation methods.

Chapter 3

Multidimensional Scaling

Multidimensional scaling (MDS) [13] is a generic name for a family of algorithms that, given a matrix representing the pairwise distances between a set of points in some abstract metric space, attempts to find a representation of these points in a low-dimensional (typically Euclidean) space. The distances in the target space should be as close to the original ones as possible. MDS algorithms are of great importance in the field of multidimensional data analysis. Originally introduced in the field of psychology [112, 67], MDS algorithms have since then been applied to various applications. The most common applications include dimensionality reduction [89, 109], visualization and analysis of data (for example, financial data ([28, 75, 71]), information retrieval [5], graph visualization [48], texture mapping in computer graphics [121], bioinformatics [46, 60, 31], etc [80, 114, 25]. More recently, MDS methods have been brought into the computer vision community as efficient methods for non-rigid shape analysis and recognition [42, 17].

The data sets encountered in the above applications are often sizable, with the number of elements in the thousands and even millions. At the same time, nonlinear and non-convex nature of MDS problems tends to make their solution computationally demanding. As a result, MDS algorithms tend to be slow, which makes their practical application in large-scale problems challenging or even prohibitive. A number of low-complexity algorithms that find an *approximate* solution to an MDS problem have been recently proposed for large-scale settings [45, 23, 115, 35, 74, 119, 15]. Yet, some of the applications (for example, the representation of intrinsic geometry of non-rigid shapes in computer vision) require a (numerically) *exact* solution, which makes approximate MDS algorithms inappropriate. Recently, Bronstein *et al.* proposed an efficient multigrid solver for the exact MDS problem used for non-rigid shapes recognition [20]. The method showed significant improvement over traditional exact MDS algorithms. It made, however, heavy use of the underlying problem geometry. The generalization of this method to generic MDS problems, where the underlying geometry is not given explicitly, is not straightforward.

We propose in this work a new efficient approach for solving the exact MDS problem, based on vector extrapolation techniques. The vectors of the sequence in our case are the iteration vectors resulting from an iterative solution technique for the MDS problem.

Specifically, in Section 3.1, we formulate the MDS problems addressed in

the following sections and review some algorithms used for their solutions. We demonstrate the usage of vector extrapolation for accelerating the solution of the MDS problem, and present a few numerical examples in Section 3.2, which demonstrate the efficiency of the proposed method.

3.1 Multidimensional Scaling

Multidimensional scaling is a very broad term, encompassing both a wide class of problems and algorithms used for their solution. Generally speaking, an MDS problem consists of finding a representation for a set of N data points, given a set of dissimilarities between them. MDS problems (and as a result, algorithms thereof) differ in the definition of the dissimilarities and the manner in which the desired representation should approximate the dissimilarities (we refer the reader to [13] for a detailed survey).

When all that is expected of the representation is a correct rank ordering of the distances between various point pairs, the problem is usually referred to as *ordinal MDS*. Here, we focus on another subset of MDS problems called *metric MDS*, in which the data is assumed to arise from an unknown metric space, and the dissimilarities are *distances* satisfying metric axioms. Metric MDS problems attempt to make the resulting distances as close as possible to the given ones.

3.1.1 Least-squares MDS

Let us denote the coordinates of the new representation as $\{\mathbf{x}_i\}_{i=1}^N$, assuming for simplicity that the points $\mathbf{x}_i$ belong to an m-dimensional Euclidean space. We write the coordinates in the form of an $N \times m$ matrix $\mathbf{X} = (x_{ij})$, where x_{ij} is the jth coordinate of point i. Let $d_{ij}(\mathbf{X})$ denote the Euclidean distance between the points $\mathbf{x}_i$ and $\mathbf{x}_j$, and let δ_{ij} be the input distance, which we would like to preserve in the new representation.

A generic metric MDS problem can be formulated as the following optimization problem,

$$\min_{\mathbf{X}} \sum_{i<j} w_{ij} f_{ERR}(d_{ij}(\mathbf{X}), \delta_{ij}),$$

where w_{ij} are weights capturing the relative importance of each pairwise distance, and f_{ERR} is an error cost function used for comparing the input and approximated distances. Choosing the pairwise distance error function to be a quadratic term of the difference between the input and the resulting distances,

$$f_{STRESS}(d, \delta) = (d - \delta)^2,$$

we obtain the *stress* function. This is one of the most common objective functions, on which we will focus in the following discussion. Defining the error function as a quadratic term of the difference between squared distances,

$$f_{SSTRESS}(d, \delta) = (d^2 - \delta^2)^2,$$

results in a function commonly known as the *sstress*. This cost function tends to prefer large distances over the local distances [13].

3.1.2 SMACOF algorithm

Trying to minimize the stress function,

$$s(\mathbf{X}) = \sum_{i<j} w_{ij}(d_{ij}(\mathbf{X}) - \delta_{ij})^2,$$

we develop the gradient of $s(\mathbf{X})$ with respect to $\mathbf{X}$, which can be written [13] as

$$\nabla s(\mathbf{X}) = 2\mathbf{V}\mathbf{X} - 2\mathbf{B}(\mathbf{X})\mathbf{X},$$

where $\mathbf{V}$ and $\mathbf{B}$ are matrices with elements given by

$$(\mathbf{V})_{ij} = \begin{cases} -w_{ij} & \text{if } i \neq j \\ \sum_{k \neq i} w_{ik} & \text{if } i = j \end{cases} \tag{3.1}$$

and

$$(\mathbf{B})_{ij} = \begin{cases} -w_{ij}\delta_{ij}d_{ij}^{-1}(\mathbf{X}) & \text{if } i \neq j \text{ and } d_{ij}(\mathbf{X}) \neq 0 \\ 0 & \text{if } i \neq j \text{ and } d_{ij}(\mathbf{X}) = 0 \\ -\sum_{k \neq i}(\mathbf{B})_{ik} & \text{if } i = j \end{cases} \tag{3.2}$$

respectively. Using the first-order optimality condition, one obtains

$$2\mathbf{V}\mathbf{X} = 2\mathbf{B}(\mathbf{X})\mathbf{X},$$

or alternatively,

$$\mathbf{X} = \mathbf{V}^{\dagger}\mathbf{B}(\mathbf{X})\mathbf{X}, \tag{3.3}$$

where † denotes the Moore–Penrose pseudoinverse. This equation gave rise to the following iterative scheme proposed by [56, 33, 69, 34],

$$\mathbf{X}^{(k+1)} = \mathbf{V}^{\dagger}\mathbf{B}(\mathbf{X}^{(k)})\mathbf{X}^{(k)}, \tag{3.4}$$

of which equation (3.3) can be regarded as the fixed point. Iteratively performing the transformation (3.3) converges to a local minimum of the stress cost function. This process is known as the *SMACOF algorithm*, standing from *Scaling by Majorizing a Complicated Function* [13]. It can be shown to be equivalent to a weighted gradient descent with constant step size (see, e.g., [20]).

A remarkable property of the stress function is that such an algorithm guarantees a monotonously decreasing sequence of stress values, which is uncommon in general optimization problems. This property is shown by developing the iteration formula (3.4) using a technique known as *iterative majorization* [13]. At the same time, the convergence of the SMACOF algorithm is slow, and a large number of iterations may be required if a high accuracy is needed, depending on the size of the data set and the composition of the distance matrix. Although scalar acceleration techniques have been suggested for SMACOF [70], these methods usually work well only in the vicinity of the fixed point.

3.1.3 Classical Scaling

Another widely used cost function is the *strain*, giving rise to an algebraic MDS algorithm referred to as *classical scaling*. Let $\mathbf{\Delta}^{(2)}$ and $\mathbf{D}^{(2)}(\mathbf{X})$ denote the

matrices of the squared input and target Euclidean distances, respectively. Let $\mathbf{J}$ be the *centering matrix* defined by $\mathbf{J} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$, whose application to a vector removes its constant component. Let us assume the coordinates of the solution have zero mean. Expressing the squared Euclidean distances, we obtain

$$d^2(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i - \mathbf{x}_j, \mathbf{x}_i - \mathbf{x}_j \rangle = \langle \mathbf{x}_i, \mathbf{x}_i \rangle + \langle \mathbf{x}_j, \mathbf{x}_j \rangle - 2 \langle \mathbf{x}_i, \mathbf{x}_j \rangle , \tag{3.5}$$

or, in matrix notation,

$$\mathbf{D}^2(\mathbf{X}) = \mathbf{c}\mathbf{1}^T + \mathbf{1}\mathbf{c}^T - 2\mathbf{X}\mathbf{X}^T.$$

(here $c_i = \langle \mathbf{x}_i, \mathbf{x}_i \rangle$).

Assuming the given distances to be Euclidean, and applying the centering matrices to the input squared distances matrix, would give us the matrix of inner products of the points (the *Gram matrix*),

$$\mathbf{B}_\Delta = \frac{1}{2}\mathbf{J}\boldsymbol{\Delta}^{(2)}\mathbf{J},$$

where in the case of Euclidean input distances, one would get

$$(\mathbf{B}_\Delta)_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle .$$

In practice, however, the input distances are not necessarily Euclidean, and $\mathbf{B}_\Delta$ is therefore not necessarily equal to the dot products. The discrepancy between the two is a measure of the approximation quality, referred to as *strain*,

$$\begin{aligned} &\left\| -\frac{1}{2}\mathbf{J}(\mathbf{D}^{(2)}(\mathbf{X}) - \boldsymbol{\Delta}^{(2)})\mathbf{J} \right\|_{\mathrm{F}}^2 \\ &= \left\| \mathbf{X}\mathbf{X}^T + \frac{1}{2}\mathbf{J}(\boldsymbol{\Delta}^{(2)})\mathbf{J} \right\|_{\mathrm{F}}^2 \\ &= \left\| \mathbf{X}\mathbf{X}^T - \mathbf{B}_\Delta \right\|_{\mathrm{F}}^2 \end{aligned}$$

(the norm here is the Frobenius norm).

The global optimum of the strain function can be found by means of eigendecomposition of the matrix $\mathbf{B}_\Delta$. Classical scaling, however, is limited both in its versatility and adaptivity (specifically, arbitrary weights for distances, or variants on the cost function usually cannot be cast into an eigenvalue problem). Furthermore, its computational cost in large problems is significant, despite several approximation methods proposed for this problem [35, 119, 15].

These drawbacks make the least-squares MDS problem preferred, and thus in the following discussion, we limit ourselves to the stress function and the SMACOF algorithm. Yet, the fact that a global optimum is guaranteed makes classical scaling attractive for initializing the SMACOF algorithm [63].

3.2 Results

We now show the effect of using vector extrapolation methods for accelerating SMACOF. We note that in general, when a higher degree of accuracy is needed, extrapolation methods tend to work better, since the iterations tend to better approximate a linear iteration scheme, as discussed in Section 2.3.

3.2.1 Positioning of Random Points

As an example we first try to use MDS to reconstruct a random configuration of points set in $\mathbb{R}^3$, based on their Euclidean distances. Gaussian noise with different variance levels is added to the distance matrix. The points were positioned in a Gaussian distribution, $x_i \sim N(0,1)$, their distance matrix is computed, and the points configuration is then reconstructed using SMACOF. 50 point configurations were attempted for each parameters set. The average speed-up in CPU time comparing SMACOF iterations with and without RRE acceleration is shown in Table 3.1.

Size (N)	$\sigma = 0$	$\sigma = 10^{-3}$	$\sigma = 10^{-2}$	$\sigma = 10^{-1}$
$N = 400$	2.2642 ± 1.7209	1.6573 ± 0.9526	1.4870 ± 0.5303	1.5284 ± 0.2864
$N = 800$	1.9447 ± 1.4139	1.9020 ± 0.9072	1.2429 ± 0.1552	1.5742 ± 0.3211
$N = 1200$	1.9040 ± 1.2879	1.8419 ± 0.9321	1.4568 ± 0.3374	1.4823 ± 0.4370
$N = 1600$	1.6292 ± 0.8268	1.6577 ± 0.6187	1.5950 ± 0.4348	1.3912 ± 0.2109

Table 3.1: The resulting speedup, in terms of the rate of decrease in residual norm, obtained using RRE with $n = 5$, $k = 5$

3.2.2 A Euclidean Dimensionality Reduction Example

In this example, previously shown by [20], the degree of the embedding space is much smaller than that of the original space used to generate the distances between points. For example, the points originally reside in $\mathbb{R}^{500}$. The points are comprised of two sets, distributed according to

$$(\mathbf{x}_i)^{(j)} = \text{sign}(n_{ij}), \quad n_{ij} \sim N(\pm 0.75, 1),$$

where

$$\text{sign}(x) = \begin{cases} +1 & x > 0 \\ -1 & x \leq 0 \end{cases}$$

The speedup obtained compared to SMACOF, in terms of CPU time, is shown in Table 3.2

Size (N)	Speedup, CPU Time
$N = 512$	1.6129 ± 0.1041
$N = 768$	1.6587 ± 0.0973
$N = 1024$	1.5416 ± 0.1418
$N = 1536$	1.4781 ± 0.0782
$N = 2048$	1.4939 ± 0.0915

Table 3.2: The resulting speedup, in terms of CPU time, obtained using RRE with $n = 8$, $k = 10$, for the 2-classes Euclidean dimensionality reduction problem

3.2.3 Graph Visualization Using Accelerated MDS

We demonstrate the use of RRE-accelerated MDS on the 1138Bus graph distances matrix, taken from the Matrix Market repository [12], and shown as an example for graph visualization based on MDS [48]. The resulting speed-up values are shown, for various n and k parameters in Table 3.3.

	$k = 6$	$k = 8$	$k = 10$	$k = 12$
$n = 1$	1.148 ± 0.147	1.212 ± 0.1746	1.234 ± 0.123	1.218 ± 0.126
$n = 3$	1.303 ± 0.186	1.267 ± 0.149	1.250 ± 0.135	1.212 ± 0.119
$n = 5$	1.392 ± 0.167	1.320 ± 0.146	1.238 ± 0.154	1.211 ± 0.097

Table 3.3: The resulting speed-up, in terms of CPU time, obtained using RRE for the 1138Bus graph, using various n and k parameters

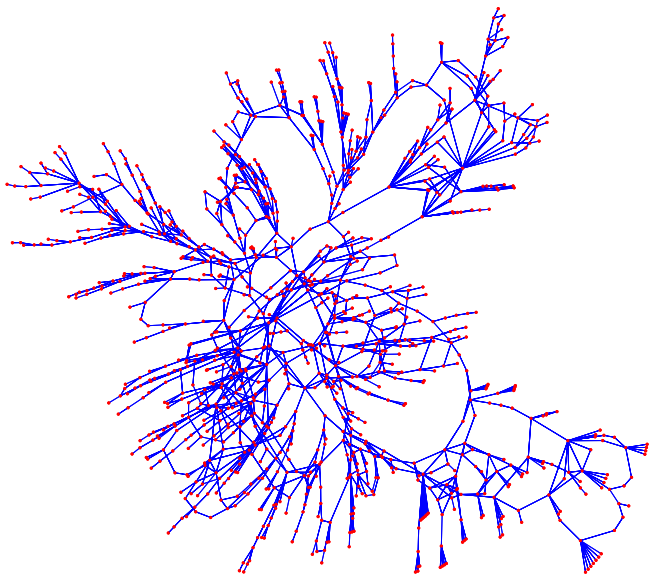

Figure 3.1: The visual results using SMACOF and RRE-accelerated SMACOF, obtained for the 1138Bus graph

3.2.4 Isometry Invariant Canonical Forms

Another application of MDS we exemplify here is the representation of non-rigid shapes invariant to deformations. This method, referred to as *canonical forms* was proposed in [42]. In this application, MDS is used to map the intrinsic geometry of the shape (captured by the matrix of pairwise geodesic distances) into a Euclidean space. This produces a representation invariant to isometric deformations of the surface, and which can be treated as a rigid shape.

The canonical forms method was used in the problem of face recognition by [18]. An example of a facial surface and its canonical form is shown in Figure 3.2. In order to compare our work to existing acceleration techniques for SMACOF, we show in Figure 3.3 the resulting stress values, as a function of CPU time, when using RRE and multigrid [20] to accelerate the computation of canonical forms.

Both the RRE and the multigrid methods seem to accelerate MDS significantly for the canonical forms problem, by a factor of $4 - 5$ for the specific example at hand. In practice, the speedup gained may vary significantly, based on the problem.

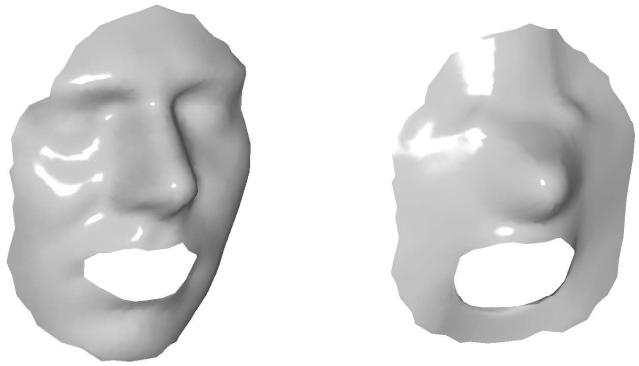

Figure 3.2: Left: A facial surface used as input to the canonical forms algorithm. Right: The resulting canonical form

3.2.5 TCIE using Accelerated MDS

We now demonstrate the use of accelerated MDS in a *nonlinear dimensionality reduction* (NLDR) application, as part of the topologically constrained isometric embedding (TCIE) algorithm [81]. Nonlinear dimensionality reduction techniques [89, 109, 83, 6, 117, 26] attempt to describe a given high-dimensional data set of points as a low dimensional manifold, by a nonlinear map preserving certain properties of the data.

These techniques have applications in several fields, such as color perception, pathology tissue analysis [26], motion understanding [78], enhancement of MRI images [37], face recognition [18], and biochemistry [64, 31].

In the TCIE algorithm, a weighted MDS problem is solved, using distances obtained by using local Euclidean distances, and their extension by means of dynamic programming. This approximates the geodesics connecting points in the manifold the data is sampled from. The weights allow the algorithm to

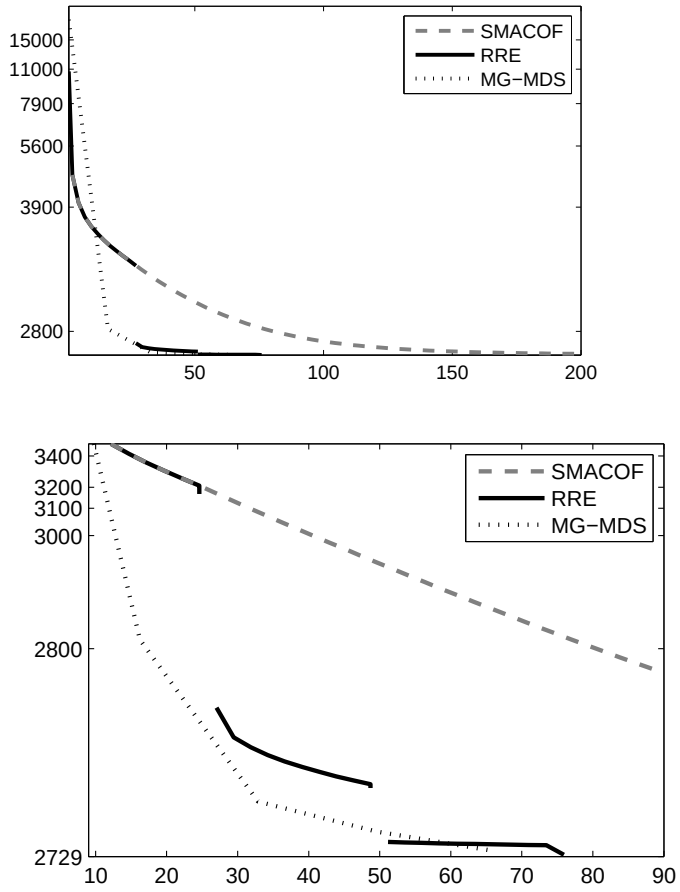

Figure 3.3: Stress, as a function of CPU time for the canonical forms problem. CPU time is approximated. The second sub-figure is a close-up of the first sub-figure.

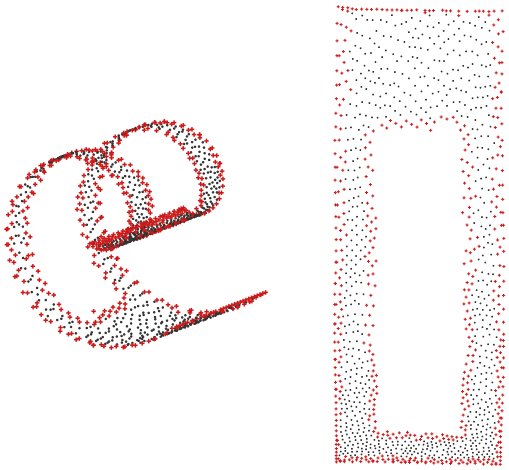

Figure 3.4: Left: Swiss roll surface with a hole in it. Detected boundary points are shown in red. Right: The mapping obtained using the TCIE algorithm

avoid approximating geodesics which should not map to a straight line in the new representation [81]. The algorithm is further described in Chapter 4.

We have used RRE to accelerate the convergence of the MDS algorithm used in the TCIE algorithm [81]. We used a Swiss roll with a hole in it as an example. The original data, as well as the data after reduction to $\mathbb{R}^2$ are shown in Figure 3.4. The RRE accelerated MDS, combined with a multiscale optimization framework, to compute the coordinates which minimize the stress function. The resulting speedups, were at least 2-3 times compared to the ordinary SMACOF algorithm computed in a multiscale manner. A graph comparing the progress of the TCIE algorithm with and without using reduced rank extrapolation, and multiscale optimization is shown in Figure 3.5

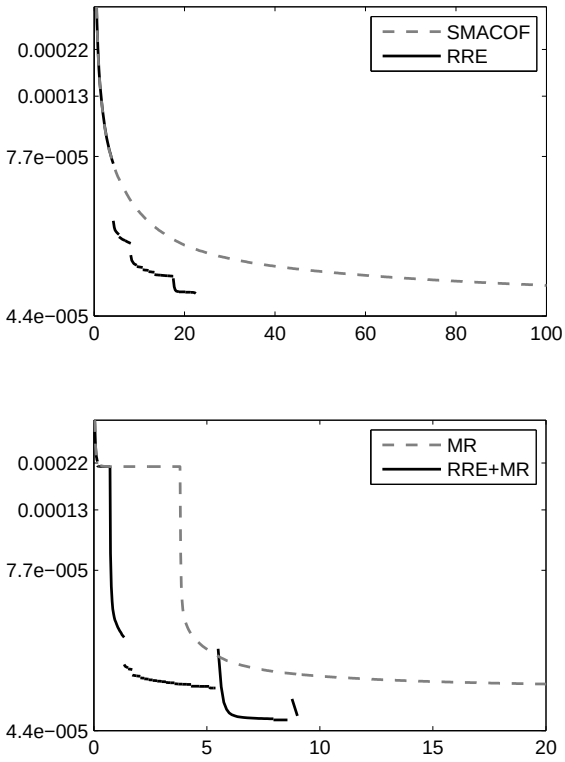

Figure 3.5: Convergence (in terms of stress value) of basic SMACOF (top, dashed gray), SMACOF with RRE (top, black), SMACOF with multiresolution acceleration (bottom, dashed gray), and SMACOF with both RRE and multiscale (bottom, black), in terms of CPU time, as part of the TCIE algorithm. CPU time is approximated. Convergence at each scale was stopped at the same relative change of stress value.

Chapter 4

Topologically Constrained Isometric Embedding

Analysis of high-dimensional data is encountered in numerous pattern recognition applications. In many cases, it appears that often just a small number of dimensions is needed to explain the high-dimensional data. Dimensionality reduction methods such as principal components analysis (PCA) [76, 40] and multidimensional scaling (MDS) [13] are often used to obtain a low dimensional representation of the data. This is a commonly used pre-processing stage in pattern recognition.

The principal components analysis algorithm linearly projects the points to a low dimensional space by minimizing the least square fitting error. Multidimensional scaling algorithms minimize the error in the pairwise distances between data points, and are highly linked in some formulations to PCA [13].

While methods such as PCA and *independent components analysis* (ICA) [59, 27, 40] assume the existence of a linear map between the data points and the parametrization space, such a map often does not exist. Applying linear dimensionality reduction methods to nonlinear data may result in a distorted representation, as shown in Figure 4.1. Nonlinear dimensionality reduction (NLDR) methods attempt to describe a given high-dimensional data set of points as a low dimensional manifold, by a nonlinear map preserving certain properties of the data. This kind of analysis has applications in numerous fields, such as color perception, pathology tissue analysis [26], enhancement of MRI images [37], face recognition [42, 18], motion understanding [78, 62] and biochemistry [64], to mention a few.

As the input data, we assume to be given N points in the M-dimensional Euclidean space, $\{\mathbf{z}_i\}_{i=1}^N \subset \mathbb{R}^M$. The points constitute vertices of a proximity graph with the set of edges E; the points $\mathbf{z}_i, \mathbf{z}_j$ are neighbors if $(i,j) \in E$. The data points are further assumed to be samples of an m-dimensional manifold $\mathcal{M} \subset \mathbb{R}^M$, (typically $m \ll M$). This manifold and the metric $d_{\mathcal{M}}$ defined on $\mathcal{M}$ form a metric space. The manifold is represented by a *parameterization domain* $\mathcal{C}$ using the smooth bijective map $\varphi : \mathcal{C} \subset \mathbb{R}^m \to \mathcal{M}$.

The goal of NLDR algorithms is to uncover the parameterization of $\mathcal{M}$. More precisely, we are looking of a set of points $\{\mathbf{x}_i \approx \varphi^{-1}(\mathbf{z}_i)\}_{i=1}^N \subset \mathcal{C} \subset \mathbb{R}^m$ representing the parametrization. We will try to compute the $N \times m$ matrix $\mathbf{X}$

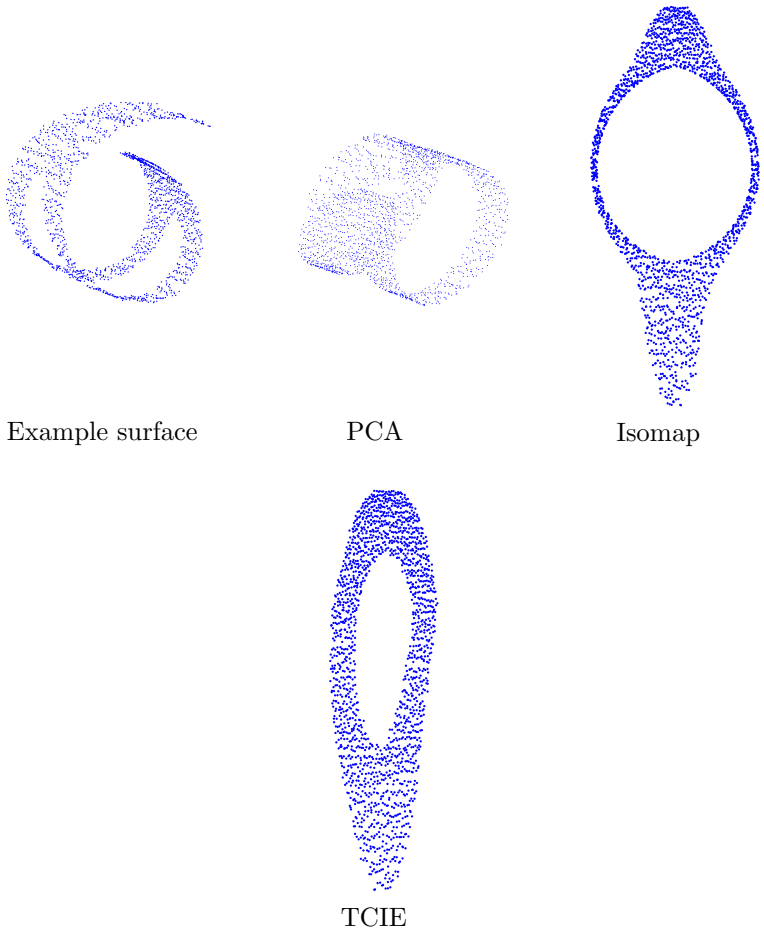

Figure 4.1: Left to right: A sampled surface (1815 samples) and the results of its dimensionality reduction into $\mathbb{R}^2$ using PCA, Isomap and the proposed TCIE algorithm.

representing the coordinates of the points in the parametrization domain.

Many NLDR techniques attempt to find an m-dimensional representation for the data, while preserving *local* properties. For example, the *locally linear embedding* (LLE) algorithm [83] tries to preserve the representation of each data point as a linear combination of its neighbors. The *Laplacian eigenmaps* algorithm [6] uses the Laplacian operator for selecting low dimensionality coordinate functions based on its eigenfunctions. The resulting coordinate system maps neighboring points in $\mathcal{M}$ to neighboring points in $\mathbb{R}^m$. The *diffusion maps* framework [26] generalizes this framework in the context of analysis of diffusion processes, making it more robust to non-uniform sampling density. The *Hessian locally linear embedding* (HLLE, [53]), tries to use the proximity graph for finding coordinate functions that have a minimal response to the Hessian operator of the surface.

Another class of algorithms preserves *global* properties, like the *geodesic distances* d_{ij}, approximated as shortest paths on the proximity graph. The *Semidefinite embedding* [117] algorithm maximizes the variance in the data set while keeping the local distances unchanged, thereby approximately preserving geodesic distances in the manifold. The problem is formulated and solved as a semidefinite programming (SDP) [9] problem, under constraints reflecting invariance to translation and local isometry to Euclidean space. Yet, the computational cost for solving an SDP problem is $O(N^6)$ [9], which is prohibitive even in medium-scale problems. Attempts to overcome it by using *landmarks* [116] still incur high computational complexity.

The *Isomap* algorithm [89, 109] considers both local and *global* invariants – the lengths of geodesics between points on the manifold. Short geodesic distances are assumed to be equal to Euclidean distances, and longer ones are approximated as shortest path lengths on the proximity graph, using standard graph search methods like the Dijkstra's algorithm [38, 29]. The resulting distance measure, $\delta_{ij} = \delta(\mathbf{z}_i, \mathbf{z}_j)$, approximates $d_{\mathcal{M}}(\mathbf{z}_i, \mathbf{z}_j)$ under certain assumptions, as shown by Bernstein et al. [10]. Isomap then uses multidimensional scaling [13] attempting to find an m-dimensional Euclidean representation of the data, such that the Euclidean distances between points are as close as possible to the corresponding geodesic ones. For example, using the L_2 criterion (referred to as *stress*),

$$\mathbf{X}^* \quad = \quad \underset{\mathbf{X}\in\mathbb{R}^{N\times m}}{\operatorname{argmin}} \sum_{i<j} \left(d_{ij}(\mathbf{X}) - \delta_{ij}\right)^2,$$

where $d_{ij}(\mathbf{X}) = \|\mathbf{x}_i - \mathbf{x}_j\|_2$ is the Euclidean distance between points $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathbb{R}^m$. Instead of Dijkstra's algorithm, higher accuracy algorithms can be used when dealing with surfaces, resulting in a more accurate mapping [121, 42].

The main advantage of Isomap is that it uses global geometric invariants, which are less sensitive to noise compared to local ones. Yet, its underlying assumption is that $\mathcal{M}$ is *isometric* to $\mathcal{C} \subset \mathbb{R}^m$ with the *geodesic metric* $d_{\mathcal{C}}$, induced by the Riemannian structure of $\mathcal{C}$. This metric is different in general from the metric of $\mathbb{R}^m$ restricted to $\mathcal{C}$, referred to as the *restricted* metric and denoted by $d_{\mathbb{R}^m}|_{\mathcal{C}}$. That is, Isomap assumes $\delta(\mathbf{z}_i, \mathbf{z}_j) = d_{\mathbb{R}^m}(\mathbf{x}_i, \mathbf{x}_j)$ for all $i, j = 1, ..., N$. If $\mathcal{C}$ is convex, the restricted metric $d_{\mathbb{R}^m}|_{\mathcal{C}}$ coincides with the geodesic metric $d_{\mathcal{C}}$ and Isomap succeeds recovering the parametrization of $\mathcal{M}$. Otherwise, $\mathcal{C}$ has no longer Euclidean geometry and no isometric map of the

dataset to $\mathbb{R}^m$ can be found. The convexity assumption of $\mathcal{C}$ appears to be too restrictive, as many data manifolds have a complicated topology. Indeed, Grimes and Donoho [52] showed examples of data for which $\mathcal{C}$ is not convex, and pointed out that Isomap fails in such cases.

We claim that even when violating the convexity assumption, one could still use non-local distances in order to stabilize the flattening procedure. We do that by detecting geodesics which may be inconsistent with the underlying convexity assumption.

Specifically, we propose a solution based on ignoring distances between pairs of points inconsistent with the convexity assumption. Our approach, hereinafter referred to as the topologically constrained isometric embedding (TCIE), allows handling flat data manifolds of arbitrary boundaries and "holes" that may often occur with sampling natural phenomena.

The rest of the chapter is organized as follows. In Section 4.1, we introduce the algorithm and prove that it rejects inconsistent geodesics. Section 4.2 discusses the numerical implementation of the algorithm and suggests ways to speed up its convergence. In Section 4.3, we demonstrate our approach on synthetic data. Proofs of supporting propositions are presented in the Appendix.

4.1 Topologically Constrained Isometric Embedding

The Isomap algorithm assumes that $\mathcal{C}$ is a convex subset of $\mathbb{R}^m$, and relies on the assumption of an isometry between $(\mathcal{C}, d_{\mathcal{C}})$ and $\mathcal{M}$ in order to find the map from $\mathcal{M}$ to the metric space $(\mathcal{C}, d_{\mathcal{C}})$ by means of MDS (the stress in the solution will be zero). If $\mathcal{C}$ is convex, this assumption is valid because $d_{\mathcal{C}} = d_{\mathbb{R}^m}|_{\mathcal{C}}$. In case $\mathcal{C}$ is non-convex, this is not necessarily true, as there may exist pairs of points for which $d_{\mathcal{C}} \neq d_{\mathbb{R}^m}|_{\mathcal{C}}$. We call such pairs *inconsistent*. An example of an inconsistent pair is shown in Figure 4.2. We denote the set of all consistent pairs by

$$P = \{(i,j) : d_{\mathcal{C}}(\mathbf{x}_i, \mathbf{x}_j) = d_{\mathbb{R}^m}|_{\mathcal{C}}(\mathbf{x}_i, \mathbf{x}_j)\} \subseteq I_N \times I_N.$$

where $I_N = \{1 \dots N\}$. In the TCIE algorithm, we find a subset $\bar{P} \subseteq P$ of pairs of points that can be consistently represented by an MDS problem. The algorithm goes as follows

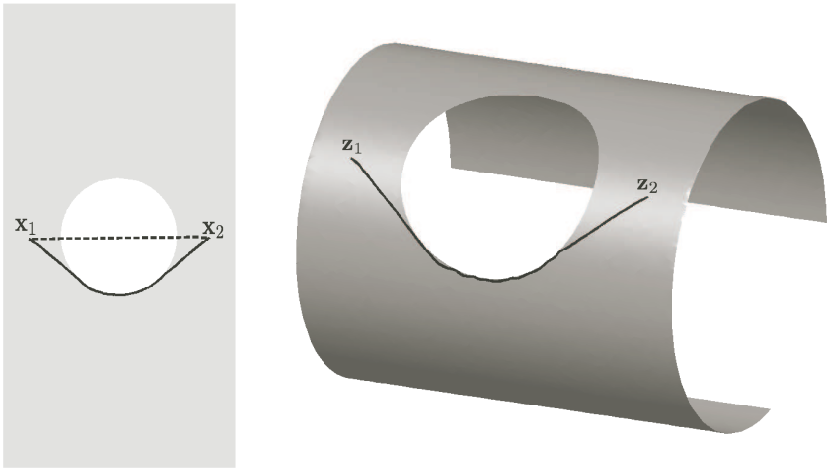

Figure 4.2: Example of two points $\mathbf{x}_1$ and $\mathbf{x}_2$, for which the line connecting the points $\mathbb{R}^m$ (dashed black line) is shorter than the geodesic $g_{\mathcal{C}}(\mathbf{x}_1, \mathbf{x}_2)$ (solid black curve), even after flattening due to non-convexity.

1 Compute the $N \times N$ matrix of *geodesic distances* $\mathbf{\Delta} = (\delta_{ij})$.
2 Detect the boundary points $\partial\mathcal{M}$ of the data manifold.
3 Detect a subset of *consistent distances* according to the following criterion (criterion 1),

$$\bar{P}_1 = \{(i,j) : c(\mathbf{z}_i, \mathbf{z}_j) \cap \partial\mathcal{M} = \emptyset\}, \quad (1)$$

or (criterion 2)

$$\bar{P}_2 = \{(i,j) : d_{\mathcal{M}}(\mathbf{z}_i, \mathbf{z}_j) \leq d_{\mathcal{M}}(\mathbf{z}_j, \partial\mathcal{M}) + d_{\mathcal{M}}(\mathbf{z}_i, \partial\mathcal{M})\}, \quad (2)$$

where $d_{\mathcal{M}}(\mathbf{z}, \partial\mathcal{M}) = \inf_{\mathbf{z}' \in \partial\mathcal{M}} d_{\mathcal{M}}(\mathbf{z}, \mathbf{z}')$ denotes the distance from $\mathbf{z}$ to the boundary.
4 Solve the MDS problem for consistent pairs only,

$$\mathbf{X}^* = \underset{\mathbf{X} \in \mathbb{R}^{N \times m}}{\operatorname{argmin}} \sum_{(i,j) \in \bar{P}} \left(d_{ij}(\mathbf{X}) - \delta_{ij}\right)^2.$$

Algorithm 1: Pseudo-code for the TCIE algorithm

The three main steps of the algorithm are (i) detection of boundary points, (ii) detection of a set of consistent geodesics, (iii) solution of a weighted MDS problem.

The choice of ignoring inconsistent pairs identified by either $\bar{P}_1$ or $\bar{P}_2$ can be generalized, using a weighted MDS with weights for each point pair that are monotonously decreasing as one approaches failure of the appropriate criterion.

4.1.1 Detection of Boundary Points

Step 2 in our algorithm involves detection of boundary points in multidimensional data. There exist many algorithms that detect boundaries in point clouds. Most of the related papers focus on practical applications for surface processing and modeling [14, 51, 47, 7], though some algorithms emerged from the field of numerical solution for PDEs (see [58], for example), or metric geometry [24], and some algorithms for boundary detection and local dimensionality estimation were motivated by perceptual research [57, 111]. Most of these methods are limited by design to specific intrinsic and extrinsic dimensions, although some methods are more easily adaptable to higher dimensional data [73, 24]. We show here a few methods for solving this generic problem and refer the reader to existing literature for a broader overview.

One approach for boundary detection on high dimensional manifolds is based on the observation that each boundary point in an m-dimensional Riemannian manifold is locally homeomorphic to a half-space of $\mathbb{R}^m$, where the boundary point is mapped to the hyperplane bordering the half-space in $\mathbb{R}^m$. An example of two such neighborhoods is shown for a two-dimensional manifold (surface) in Figure 4.3.

We therefore expect the boundary point to have its neighbors on one side of a single hyperplane in an m-dimensional mapping of its neighborhood. Multidimensional scaling of the neighborhood distances matrix can be used to obtain such a mapping.

Looking at the normal direction to this hyperplane, the mean of the neighboring points should be far from the boundary point. The first boundary detection method we present, (Algorithm 2), follows this line of thought.

for $i = 1, ..., N$ **do**
    Find the set $\mathcal{N}(i)$ of the K nearest neighbors of the point i.
    Apply MDS to the $K \times K$ matrix $\mathbf{\Delta}_K = (\delta_{k,l \in \mathcal{N}(i)})$ and obtain a set of local coordinates $\mathbf{x}'_1, ..., \mathbf{x}'_K \in \mathbb{R}^m$, where $\mathbf{x}'_1$ denotes the mapping of point i.
    Let $\boldsymbol{\mu}(i) = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{x}_i$ denote the mean of the neighbors of $\mathbf{x}_1$. $|\mathcal{N}(i)|$ denotes the cardinality of $\mathcal{N}(i)$.
    Denote by $\overline{d}_1(i)$ the distance between $\mathbf{x}'_1$ and $\boldsymbol{\mu}(i)$, normalized by the average distance between points in $\mathbf{x}'_1, ..., \mathbf{x}'_K$.
    if $\overline{d}_1(i)$ *is larger than some threshold* τ_a **then**
        Label point i as `boundary`.
    end
end

Algorithm 2: Boundary detection algorithm

The described method is similar to that of Belton and Lichti [7], where each coordinate is normalized separately, based on the standard deviation of points along the tangent space direction. For a high dimensional manifolds, this method can be extended as shown in Algorithm 3.

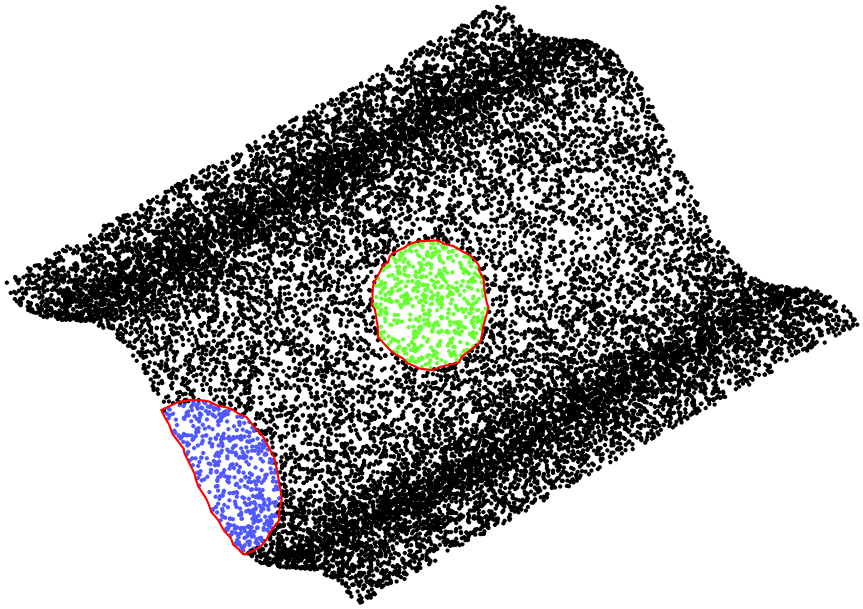

Figure 4.3: An example of two neighborhoods of points in $\mathcal{M}$, one of which is close to the boundary $\partial\mathcal{M}$. In the example, $N = 20000$, and each neighborhood was chosen to include 500 nearest neighbors of a point on the manifold.

for $i = 1, ..., N$ **do**
    Find the set $\mathcal{N}(i)$ of the K nearest neighbors of the point i.
    Apply MDS to the $K \times K$ matrix $\mathbf{\Delta}_K = (\delta_{kl \in \mathcal{N}(i)})$ and obtain a set of local coordinates $\mathbf{x}'_1, ..., \mathbf{x}'_K \in \mathbb{R}^m$, where $\mathbf{x}'_1$ denotes the mapping of point i.
    Let $\boldsymbol{\mu}(i) = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{x}_i$ denote the mean of the neighbors of $\mathbf{x}_1$.
    Denote by $\overline{d}_2(i)$ the distance between $\mathbf{x}'_1$ and the $\boldsymbol{\mu}(i)$, where each coordinate (according to some basis) is scaled in order to normalize its standard deviation.
    if $\overline{d}_2(i)$ *is larger than some threshold* τ_b **then**
        Label point i as `boundary`.
    end
end

Algorithm 3: Boundary detection algorithm

Algorithm 4 is similarly motivated, yet more heuristic. Directions are selected according to neighboring points, avoiding the need to artificially determine the normal direction, or using the mean of the points, which may be sensitive to sampling. Assuming that for an interior point, for all directions, the distribution of projected points is homogeneous, the algorithm tries to detect directions for which the projection of the sampled neighboring points has a single sided distribution. This scheme uses neighboring points in order to determine directions of projection, voting among possible directions in order to obtain a more robust classification.

for $i = 1, ..., N$ **do**
    Find the set $\mathcal{N}(i)$ of the K nearest neighbors of the point i.
    Apply MDS to the $K \times K$ matrix $\mathbf{\Delta}_K = (\delta_{kl \in \mathcal{N}(i)})$ and obtain a set of local coordinates $\mathbf{x}'_1, ..., \mathbf{x}'_K \in \mathbb{R}^m$.
    for $j = 1, ..., K$ **do**
        if $\frac{\left|\{\mathbf{x} \in \mathcal{N}(i) : \langle \mathbf{x}'_i - \mathbf{x}'_j, \mathbf{x} - \mathbf{x}'_i \rangle > 0\}\right|}{\left|\{\mathbf{x} \in \mathcal{N}(i) : \langle \mathbf{x}'_i - \mathbf{x}'_j, \mathbf{x} - \mathbf{x}'_i \rangle \leq 0\}\right|} \leq \tau_c$ **then**
            mark j as `candidate`.
        end
    end
end
if *the number of* `candidate` *points is larger than* τ_d **then**
    Label point i as `boundary`.
end

Algorithm 4: Boundary detection algorithm

Another property, which may be useful in boundary detection is the fact that a small neighborhood around an interior point should be isotropic in shape and sampling, whereas the neighborhood should be anisotropic for a boundary point, as seen in the example in Figure 4.3. Belton and Lichti [7] claim, however, that direct usage of this property in an algorithm would be too sensitive to non-uniform sampling.

In our experiments, the proposed algorithms performed similarly on noisy data. Other boundary detection algorithms can be used as well. We expect a voting mechanism (see for example, [57]) to be quite beneficial for robust detection of the boundaries. Multiscale information should also prove quite beneficial, although using it in an optimal manner is still an open question. For manifolds with a large intrinsic dimensionality, dense sampling is usually required for reliable boundary detection.

4.1.2 Detection of Inconsistent Geodesics

An important part of the proposed algorithm is the detection of inconsistent pairs. We find all point pairs adhering to consistency criteria (1) or (2), including all inconsistent pairs.

The first consistency criterion requires to check whether geodesics touch the boundary. Once we have detected the boundary points, we use a modification of the Dijkstra algorithm [38, 29], as summarized in Algorithm 5, using a notation similar to the one used by Cormen et al. [29].

```
1  foreach z_u ∈ M \ {z_s} do
2      d(z_s, z_u) ← ∞
3  end
4  d(z_s, z_s) ← 0
5  Let Q be a queue of remaining vertices, sorted according to current
   distance to z_s. Let S be the set of all vertices whose distance to z_s has
   already been fixed by the algorithm. Set w_ij ← 1 for all point pairs i, j.
6  while Q ≠ ∅ do
7      Let z_u be the minimum distance vertex stored in Q
8      Add z_u to S
9      foreach z_v ∈ N_u do
10         if d(z_s, z_v) > d(z_s, z_u) + d_{R^M}(z_u, z_v) then
11             d(z_s, z_v) ← d(z_s, z_u) + d_{R^M}(z_u, z_v)
12             if w_su = 0 or (z_u ∈ ∂M and d(z_s, z_u) > 0) then
13                 w_sv = 0
14             else
15                 w_sv = 1
16             end
17         end
18     end
19 end
```

Algorithm 5: A modification to Dijkstra's shortest path algorithm . We set w_{ij} to 0 if the geodesic between i and j touches the boundary, and to 1 otherwise. $d(\cdot,\cdot)$ denotes here the current approximation to the geodesic distance, as computed by the Dijkstra algorithm

Note that the second condition in line 12 of the algorithm protects paths with only a boundary end point from being removed. This way we eliminate only geodesics for which the point of intersection with the boundary is a midpoint. Similar modifications can be made to the Bellman-Ford and Floyd algorithms, or other algorithms [66]. We note that for the criterion defining $\bar{P}_2$, detection of inconsistent geodesics is done simply by comparing the relevant geodesic distances.

When proving the correctness of the algorithm, we assume a continuous case, in which the manifold is sampled with non-zero density. We make the same assumptions on the sampling density and uniformity made by Bernstein et al. [10], who proved the convergence of the graph distances approximation, used by the Isomap algorithm [89, 109], to the geodesic distances on the manifold. Also note that the requirement of a positive density function prevents problems that may occur in geodesics approximated by a graph when the surface is sampled in a regular pattern. In our case, there is also the question of whether or not we remove too many geodesics. The answer is related to the topology of the manifold.

In the continuous setting, our algorithm approximates an isometry φ^{-1} between $\mathcal{M}$ with the geodesic metric δ and $\mathcal{C} \subset \mathbb{R}^m$ with the geodesic metric $d_{\mathcal{C}}$. In the case where $\mathcal{C}$ is a convex region, geodesics connecting points in $\mathcal{C}$ are always straight lines, and the geodesic metric is identical to the restricted Euclidean metric. When $\mathcal{C}$ is no longer a convex region, $\bar{P}_1$ restricts our choice of point pairs, selecting only consistent distances, as shown the following proposition.

Proposition 1. *Let $\mathcal{M}$ be a manifold, isometric to $\mathcal{C} \subseteq \mathbb{R}^m$, and $c_{\mathcal{M}}(\cdot\cdot)$ denote geodesics in $\mathcal{M}$. Then $\bar{P}_1 = \{(i,j) : c_{\mathcal{M}}(\mathbf{z}_i, \mathbf{z}_j) \cap \partial\mathcal{M} = \emptyset\} \subseteq P$.*

Therefore, for every geodesic in $\mathcal{M}$ which was not detected as touching the boundary, the image under φ^{-1} is a line, which is approximated correctly by the MDS procedure.

In the case where $\mathcal{C}$ is no longer a subset of a Euclidean space, but rather part of a manifold $\mathcal{C}'$ endowed with another metric. we claim that selecting only point pairs from $\bar{P}_2$ still leaves us with only consistent pairs.

Proposition 2. $\bar{P}_2 = \{(i,j) : d_{\mathcal{M}}(\mathbf{z}_i, \mathbf{z}_j) \leq d_{\mathcal{M}}(\mathbf{z}_j, \partial\mathcal{M}) + d_{\mathcal{M}}(\mathbf{z}_i, \partial\mathcal{M})\} \subseteq P$.

The proofs of Propositions 1 and 2 can be found in the appendix.

Note that for a parametrization manifold $\mathcal{C}'$ with an arbitrary Riemannian metric, the MDS procedure would not be able to give us the correct mapping. This would require the use of a more general procedure, as done by Bronstein et al. [19]. Criterion 2 may still be useful in cases where the metric on $\mathcal{C}'$ is close to Euclidean, and yet we only want to use geodesics which do not leave $\mathcal{C}$.

4.1.3 Weighted LS-MDS

The final stage of our approach is solving the MDS problem for the subset $\bar{P}_i, i \in \{1,2\}$ of distances. One way to include only consistent point pairs in the optimization is to use *weighted stress*,

$$\mathbf{X}^* = \underset{\mathbf{X}}{\operatorname{argmin}} \sum_{i=0, i<j} w_{ij}(d_{ij}(\mathbf{X}) - \delta_{ij})^2,$$

where $w_{ij} = 1$ if $(i,j) \in \bar{P}$ and $w_{ij} = 0$ otherwise. This allows us, by choosing the right weights, to minimize the error only for consistent geodesics.

The geodesics that were not marked as inconsistent have their weight set to one. We also allow a positive weight for short geodesics, in order to keep the connectivity of the manifold, even at boundary points. All other geodesics have their weight set to zero. We then use the SMACOF algorithm [13] to minimize the weighted stress.

We note that the correctness of these conditions depends on the assumption that our manifold is isometric to a subregion of an Euclidean space, similarly to the underlying assumption of Isomap.

4.2 Implementation Considerations

For determining the shortest paths we used the Dijkstra algorithm implementation supplied by Tenenbaum et al. [109], for the Isomap code. We added the detection of geodesics touching boundary points. The rest of the algorithm was implemented in MATLAB. In practice, the Dijkstra algorithm takes less than 10% of the total running time for 2000 points. Solving the MDS optimization problem consumes most of the time ($O(N^2)$ per iteration). The boundary detection takes $O(N^2)$, but with much smaller constants, and the distance computation takes $O(N^2 \log N)$, using the Dijkstra algorithm. We note that while the number of SMACOF iterations is not invariant to the number of samples, in practice it rises slowly, depending on the topology of the manifold and the noise level.

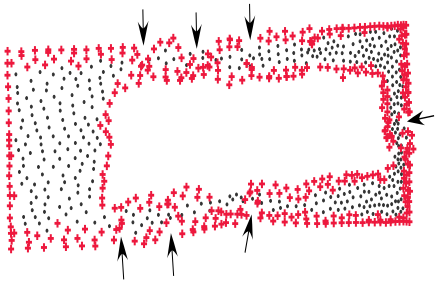

Figure 4.4: Example of a local minimizer of the weighted MDS problem. Flips along the boundaries of the rectangle are marked with arrows.

4.2.1 Numerical Properties and Convergence

The optimization problem solved is a non-convex one, and as such is liable to local convergence if convex optimization methods are employed [113]. In our experiments, we have seen that removing more distances from the stress function caused the problem to be more sensitive to local minima. An example of one such local minimum, encountered when flattening the Swiss roll with a hole, is shown in Figure 4.4. It appears as a fold over, or a "flip". In general, the number of remaining weights depends on the surface topology, as well as the number of sampled points in the surface [1].

We reduce the risk of convergence to a local minimum by starting from a classical scaling (as mentioned by Trosset et al. [63] and Aharon et al. [1]) or unweighted least squares scaling solution. This allows the algorithm to avoid some of the local minima. Although the solutions found by classical scaling and least square scaling may differ, under the assumption of correct distance approximation, the solutions are similar.

Using the unweighted LS-MDS problem to avoid local minima, and then gradually changing the problem into the weighted one is in the flavor of graduated non-convexity [11] algorithms, variants of which are well-known in the MDS literature [54].

To speed up the convergence of the SMACOF iterations, we employ *vector extrapolation*, as is described by Rosman et al. [82], and reviewed in Chapter 3.

4.2.2 Convergence Acceleration by Multiscale Optimization

Another way to accelerate the solution of the MDS problem is using *multiresolution* (MR) methods [23, 77, 101, 20, 15]. The main idea is subsequently approximating the solution by solving the MDS problem at different resolution levels. At each level, we work with a *grid* consisting of points with indices $\Omega_L \subset \Omega_{L-1} \subset ... \subset \Omega_0 = \{1, ..., N\}$, such that $|\Omega_l| = N_l$. At the lth level, the data is represented as an $N_l \times N_l$ matrix $\mathbf{\Delta}_l$, obtained by extracting the rows and columns of $\mathbf{\Delta}_0 = \mathbf{\Delta}$, corresponding to the indices Ω_l. The solution $\mathbf{X}_l^*$ of

[1] Typically, in our experiments $\mathbf{W}$ contained between 6% to 18% nonzero weights.

the MDS problem on the lth level is transferred to the next level $l-1$ using an *interpolation operator* P_l^{l-1}, which can be represented as an $N_{l-1} \times N_l$ matrix.

1 Construct the hierarchy of grids $\Omega_0, ..., \Omega_L$ and interpolation operators $P_1^0, ..., P_L^{L-1}$.
2 Start with some initial $\mathbf{X}_L^{(0)}$ at the coarsest grid, and $l = L$.
3 **while** $l \geq 0$ **do**
4 Solve the lth level MDS problem

$$\mathbf{X}_l^* = \underset{\mathbf{X}_l \in \mathbb{R}^{N_l \times m}}{\operatorname{argmin}} \sum_{i,j \in \Omega_l} w_{ij}(d_{ij}(\mathbf{X}_l) - \delta_{ij})^2$$

using SMACOF iterations initialized with $X_l^{(0)}$.
5 Interpolate the solution to the next resolution level, $\mathbf{X}_{l-1}^{(0)} = P_l^{l-1}(\mathbf{X}_l^*)$
6 $l \longleftarrow l - 1$
7 **end**

Algorithm 6: Multiscale implementation of SMACOF

We use a modification of the *farthest point sampling* (FPS) [50, 61, 44] strategy to construct the grids, in which we add more points from the boundaries, to allow correct interpolation of the fine grid using the coarse grid elements. We use linear interpolation with weights determined using a least squares fitting problem with regularization made to ensure all available nearest neighbors are used.

The multiresolution scheme can be combined with vector extrapolation by employing MPE or RRE methods at each resolution level. In our experiments we used the RRE method, although in practice, for the SMACOF algorithm, both the MPE and the RRE algorithms gave comparable results, giving us a three-fold speedup. This is further detailed in Chapter 3.

4.3 Results

In order to assess the performance of the proposed approach, a few experiments were performed. In the first experiment, we used the Swiss roll surface with a large rectangular hole, sampled at 1200 points. Flattening was performed for points sampled on the manifold with additive independently identically distributed Gaussian noise in each coordinate of each point. The various instances of the surface with noise are shown in Figure 4.5.

We compare the proposed algorithm to Isomap, LLE, Laplacian eigenmaps, diffusion maps and Hessian LLE, in Figures 4.7 – 4.9[2]. Our algorithm finds a

[2]We used the same number of neighboring points (12) in all algorithms, except for the diffusion maps algorithm, where we used value of the diffusion distance constant according to

$$\sigma^2 = \frac{2}{N} \sum_{i=0}^{N} \min_j \|x_i - x_j\|^2.$$

This is the rule used by Lafon [68] up to a constant. The larger diffusion distance gave us more robust results. More specifically, the values of ϵ we used in the noiseless example and with $\sigma = 0.015$, $\sigma = 0.05$ were 3.897×10^{-3}, 3.998×10^{-3} and 8.821×10^{-3} respectively.

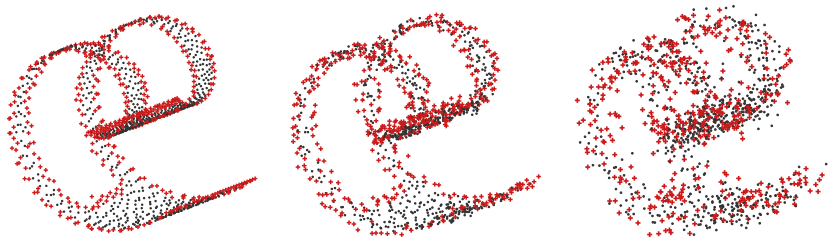

Figure 4.5: Left to right: Swiss roll surface without noise, contaminated by Gaussian noise with $\sigma = 0.015$ and $\sigma = 0.05$. Detected boundary points are shown as red crosses.

representation of the manifold, with relatively small distortion. Adding i.i.d. Gaussian noise to the coordinates of the sampled points, our method remains accurate compared to other popular algorithms that exhibit large distortions. This can be seen, for example, for 1200 points, with $\sigma = 0.05$, in Figure 4.9, where for comparison, $\mathrm{diam}(\mathcal{M}) \approx 6$. The algorithm was allowed to converge until the relative change in the weighted stress was below some threshold. Tests with higher noise levels were also performed and the performance boosting due to the RRE method is quite consistent. Using multiscale further reduces the computational cost of the solution by a factor of 2, for the problem shown in the example. We note that the speedup depends on both the manifold topology and the problem size. The reduction in computational effort is typical for all the problems we tested.

In the second experiment, the data were samples of a planar patch cut in the form of a spiral. Ideally, a correct solution can be achieved by linear methods such as PCA on the embedding space coordinates, and the large number of geodesics removed (only 6% of the geodesic distances remained) makes it a worst case scenario for our algorithm. In practice, as shown in Figure 4.10, the proposed algorithm introduces just a minor distortion whereas other methods fail to extract the structure of the manifold.

In the third experiment, we applied our algorithm on a set of images representing human eyes gaze directions. Although in practical cases the data manifold is not necessarily isometric to a subset of a low-dimensional Euclidean space, our algorithm is able to produce meaningful results in image analysis applications. Figure 4.11 demonstrates the recovery of gaze direction of a person from a sequence of gray-scale images. Assuming that facial pose and expressions do not change significantly, images of the area of the eyes form a manifold approximately parameterized by the direction of the gaze. Similar to previous image manifold experiments [109], we use Euclidean distances between the row-stacked images as the distance measure. In order to reduce the effect of head movement, simple block matching was used.

As for α, the parameter used by Coifman et al. [26] to specify various types of diffusion maps, we show here the results for $\alpha = 1$, though other values of α were also tested.

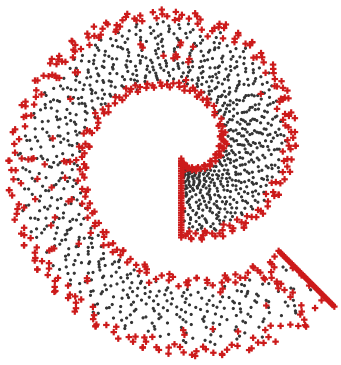

Figure 4.6: A planar surface cut as a spiral. Detected boundary points are shown as red crosses.

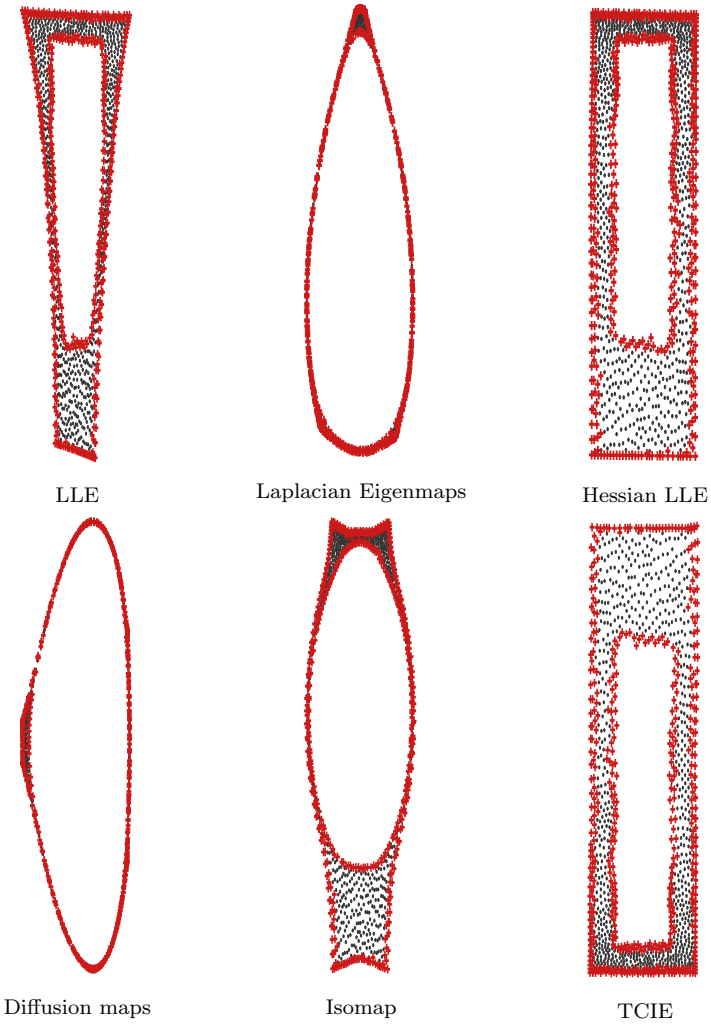

Figure 4.7: Embedding of the Swiss roll (without noise), as produced by LLE, Laplacian eigenmaps, Hessian LLE, diffusion maps, Isomap, and our algorithm. Detected boundary points are shown as red crosses.

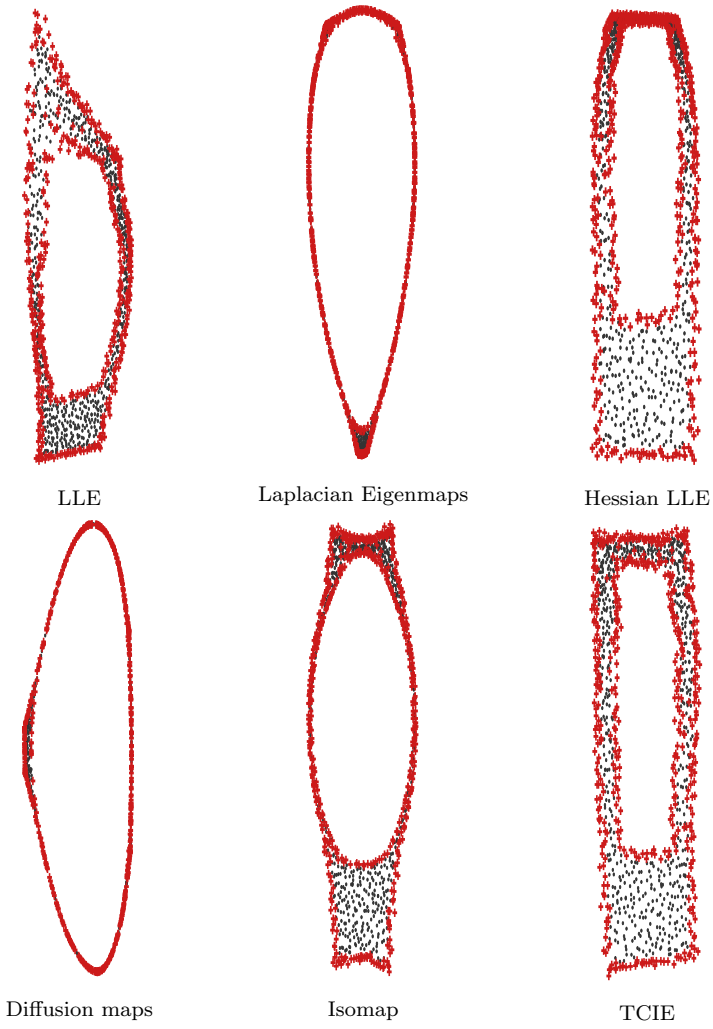

Figure 4.8: Embedding of the Swiss roll contaminated by Gaussian noise with $\sigma = 0.015$, as produced by LLE, Laplacian eigenmaps, Hessian LLE, diffusion maps, Isomap, and our algorithm. Detected boundary points are shown as red crosses.

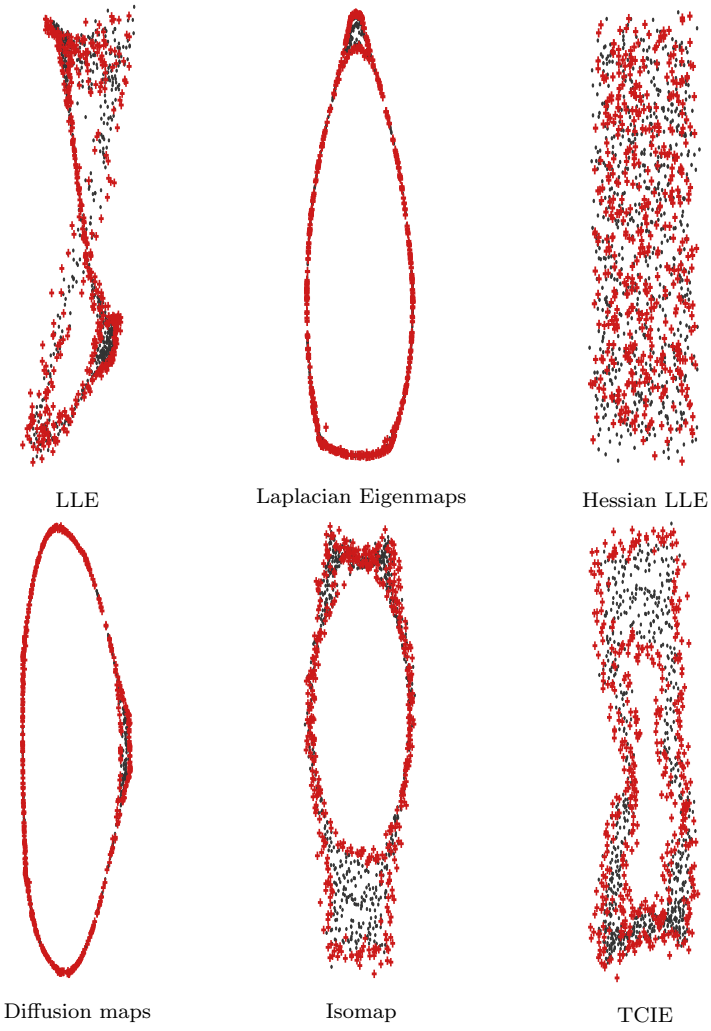

Figure 4.9: Embedding of a 2D manifold contaminated by Gaussian noise with $\sigma = 0.05$, as produced by LLE, Laplacian eigenmaps, Hessian LLE, diffusion maps, Isomap, and our algorithm. Detected boundary points are shown as red crosses.

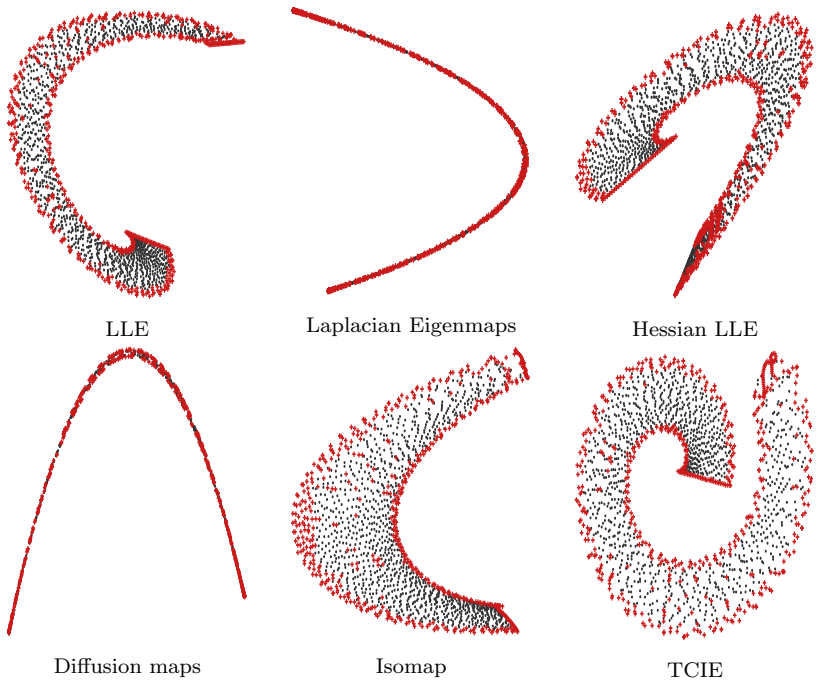

Figure 4.10: Embedding of the spiral manifold, as produced by LLE, Laplacian eigenmaps, Hessian LLE, diffusion maps, Isomap, and our algorithm. Detected boundary points are shown as red crosses.

Figure 4.11: Representation of the gaze direction manifold uncovered by our algorithm.

Chapter 5

Efficient Beltrami Filtering using Vector Extrapolation

The Beltrami framework was introduced in [106], [107], followed by [120]. This nonlinear flow was applied as an edge preserving denoising and deblurring algorithm for signals and especially multi-channel images, see for example [3]. 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 [108]. 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 [36] for an approximation of the Beltrami flow. This approximation is not, however, consistent with the Beltrami operator. Rather, it discretizes a slightly different PDE.

Strongly related to the Beltrami operator, the bilateral operator was studied in different contexts (see for example [104], [110], [105], [43], [4]), and can be shown to be an approximation of the Beltrami kernel.

In this chapter, study the application of the two vector extrapolation methods surveyed in Chapter 2 to acceleration of the convergence rate of standard explicit schemes for the Beltrami flow in color. 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 [32], and we elaborate upon it here. We note that although Krylov subspace methods and related extrapolation techniques have been used for computation of nonlinear diffusion based processing in image processing

[22], they have not been used, to the best of our knowledge, for accelerating the computation of the Beltrami flow.

We demonstrate the efficiency and accuracy of MPE and RRE in color image processing applications, such as scale-space analysis, denoising, and deblurring.

This chapter is organized as follows: Section 5.1 gives a brief summary of the Beltrami framework. In Section 5.2, we review approximations based on standard explicit finite-difference schemes. In Section 5.3, we apply RRE to our Beltrami color flow and demonstrate the resulting speed-up.

5.1 The Beltrami Framework

Let us briefly review the Beltrami framework for non-linear diffusion in computer vision [65, 106, 107, 120]. For a more complete introduction to the Riemannian geometry tools used, henceforth, we refer the reader to [39].

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 $X : \Sigma \to M$,

$$S[X, g_{ij}, h_{ab}] = \int d^m\sigma \sqrt{g} ||dX||^2_{g,h}, \tag{5.1}$$

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 [79] 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 with respect to the embedding (assuming Euclidean embedding space) are

$$0 \quad = \quad -\frac{1}{\sqrt{g}} h^{ab} \frac{\delta S}{\delta I^b} = \underbrace{\frac{1}{\sqrt{g}} \mathrm{div}\,(D \nabla I^a)}_{\Delta_g I^a}, \tag{5.2}$$

where the matrix $D = \sqrt{g} G^{-1}$. See [106] 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 [107] for details).

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

$$I^a_t = -\frac{1}{\sqrt{g}} \frac{\delta S}{\delta I^a} = \Delta_g I^a. \tag{5.3}$$

For Euclidean embedding, the functional in Eq. (5.1) reduces to

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

where

$$g = \det(g_{ij}) = 1 + \beta^2 \sum_{a=1}^{3} |\nabla I^a|^2 + \frac{1}{2}\beta^4 \sum_{a,b=1}^{3} |\nabla I^a \times \nabla I^b|^2. \tag{5.4}$$

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 [65]. 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

$$\Psi = \frac{\alpha}{2} \sum_{a=1}^{3} ||K I^a - I^a_0||^2 + S(X),$$

where K is a bounded linear operator. In the denoising case, K is the identity operator and in the deblurring case, K is a convolution operator of I^a with a given filter. 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:

$$I^a_t = -\frac{1}{\sqrt{g}} \frac{\delta \Psi}{\delta I^a} = -\frac{\alpha}{\sqrt{g}} (I^a - I^a_0) + \Delta_g I^a. \tag{5.5}$$

ii) for deblurring:

$$I_t^a = -\frac{1}{\sqrt{g}}\frac{\delta\Psi}{\delta I^a} == -\frac{\alpha}{\sqrt{g}}\bar{k} * (k * I^a - I_0^a) + \Delta_g I^a, \tag{5.6}$$

where $Ku = k * u$, k is the blurring kernel (often assumed to be Gaussian), and $\bar{k}(x, y) = k(-x, -y)$.

The Laplace-Beltrami operator in Eqs. (5.5) and (5.6) 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.

5.2 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 h in space. The spatial units are normalized such that $h = 1$. For each channel I^a, $a \in \{1, 2, 3\}$, we define the discrete approximation $(I^a)_{ij}^n$ by

$$(I^a)_{ij}^n \approx I^a(ih, jh, n\Delta t).$$

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

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

$$(I^a)_{ij}^{n+1} = (I^a)_{ij}^n + \Delta t O_{ij}^n(I^a), \tag{5.7}$$

where O_{ij}^n is the discretization of the right hand side of the relevant continuous equation (5.3), (5.5), or (5.6). Below, we give the exact form of the operator O_{ij}^n for each of the above cases.

- *Beltrami-based smoothing.*

  The explicit scheme (5.7) for discretizing Eq. (5.3) takes the form

  $$(I^a)_{ij}^{n+1} = (I^a)_{ij}^n + \Delta t L_{ij}^n(I^a), \tag{5.8}$$

  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 (5.7) is given in this case by

  $$(I^a)_{ij}^{n+1} = (I^a)_{ij}^n + \Delta t\big[L_{ij}^n(I^a) + \frac{\alpha}{\sqrt{g}}((I_0^a)_{ij}^n - (I^a)_{ij}^n)\big]. \tag{5.9}$$

- *Beltrami-based deblurring.*

Similarly, in the deblurring case, the explicit scheme (5.7) reads

$$(I^a)_{ij}^{n+1} = (I^a)_{ij}^n + \Delta t\big[L_{ij}^n(I^a) + \frac{\alpha}{\sqrt{g}}\bar{k}_{ij}^n * \big((I_0^a)_{ij}^n - k_{ij}^n * (I^a)_{ij}^n\big)\big], . \tag{5.10}$$

Due to stability requirements (see [30], [55]), 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.

5.3 Experimental Results

We have applied RRE to the problems described in Section 5.1 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 RRE-accelerated scheme, specifying the CPU runtime and the resulting speed-up. 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 5.1.

In our experiments, we apply RRE in the cycling mode (with $n = 10$ initial explicit iterations and, unless specified otherwise, $k = 10$). The RRE-accelerated scheme allows 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 RRE scheme remains stable as the number of iterations increases.

Figure 5.1 top row depicts the scale-space behavior of the Beltrami color flow, obtained using Eq. (5.8). 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 measure the approximation error using l^2-norm values. Figure 5.2 shows the l^2-norm values of the images generated by the explicit and the RRE schemes during the scale-space evolution. 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 value observed during scale-space evolution was 0.194%. 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.

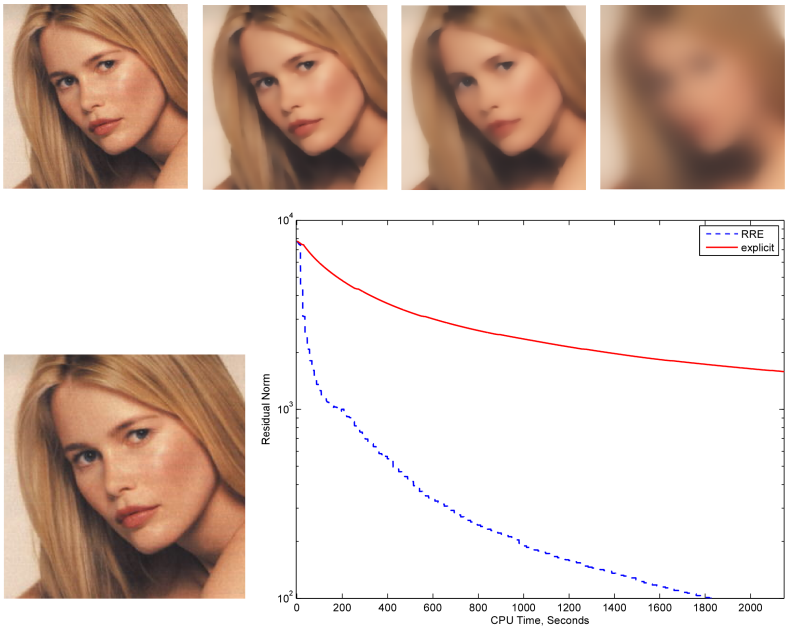

Figure 5.1: Top (left to right): RRE iterations: $50, 150, 300, 450$. Bottom: left: Original picture. right: Comparison of the residual norms versus CPU time. Parameters: $\beta = \sqrt{1/0.0005} \simeq 44$, $\Delta t = 0.0042/\beta^2$.

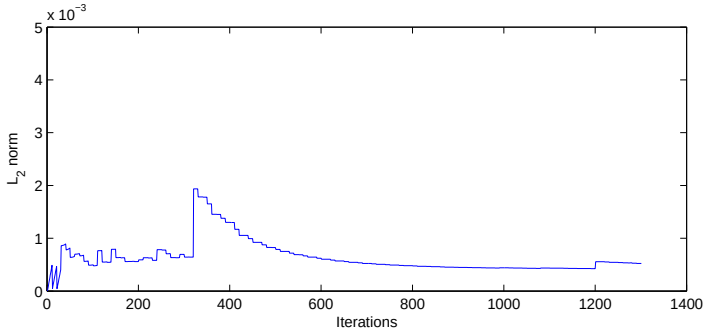

Figure 5.2: l^2-norm error between the RRE iterations sequence and the corresponding explicit iterations images, in the case of scale-space evolution, shown in Figure 5.1. A sequence using $k = 15$ is shown, which gave the worst l^2-norm values in practice. Parameters: $\beta = \sqrt{1/0.001} \simeq 31.62$, $\Delta t = 0.0042/\beta^2$.

5.3.1 Beltrami-based Denoising

Figure 5.3 displays the restoration of an image from its noisy version, corrupted by additive Gaussian noise, by applying Eq. (5.9). The speed-up factor in this case is about 10.

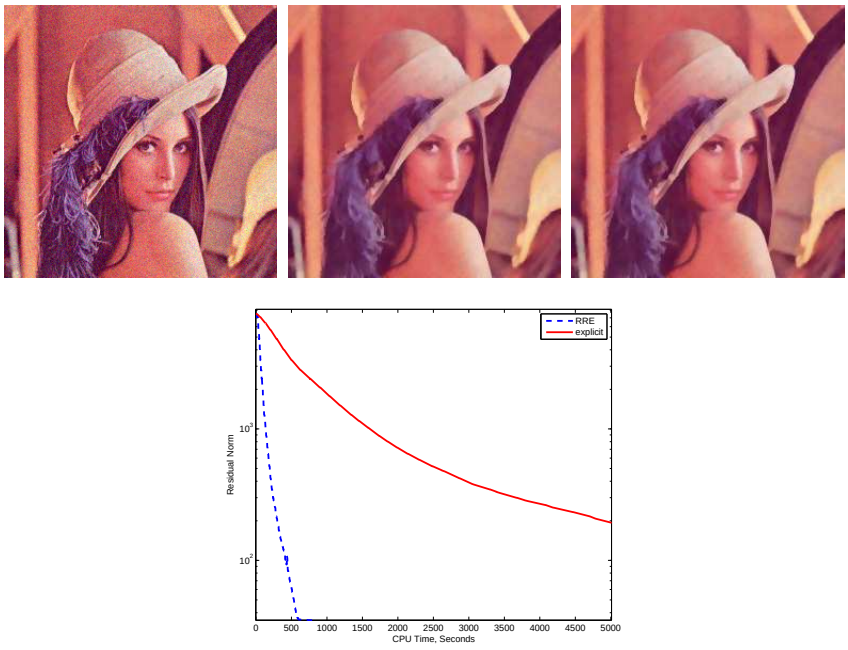

Figure 5.3: Beltrami based denoising. Top left: Noisy image. Middle: Denoised image obtained by the RRE (901 iterations). Right: Denoised image obtained by the explicit scheme (11541 iterations). Bottom: Comparison of the residual norms versus CPU time. Parameters: $\beta = \sqrt{1000}$, $\lambda = 0.02$,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

5.3.2 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 Eq. (5.10). A significant speed-up is obtained in this case, as seen in Figure 5.4 bottom.

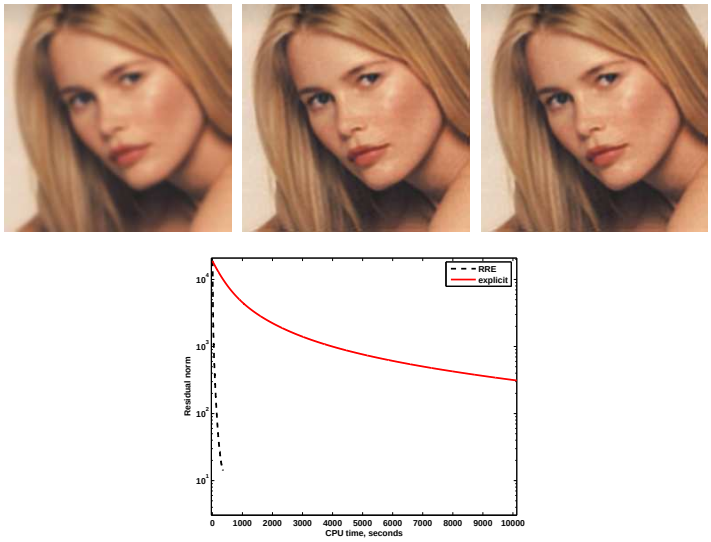

Figure 5.4: Beltrami-based deblurring. Top left: Blurred image. Middle: Deblurred image obtained by the RRE scheme (1301 iterations). Right: Deblurred image obtained by the explicit scheme (196608 iterations). Bottom: Comparison of the residual norms versus CPU time. Parameters: $\beta = \sqrt{1/0.0005} \simeq 44$, $\Delta t = 0.0021/\beta^2$, $\lambda = 0.03$.

Chapter 6

Conclusions

The multidimensional scaling problem is encountered in many applications, in a variety of areas of science. The computational complexity of its accurate solution often limits the scale of problems in which it can be accurately solved. This leaves room for improving the efficiency of the computation, in each of its steps. In this work we presented a way of accelerating the computation of a solution to the MDS problem using the SMACOF algorithm, using vector extrapolation techniques, and demonstrated its effectiveness in a variety of practical problems.

In the case of the Beltrami process, 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.

Although these methods were based on linear iterative schemes, they are shown in this work to be quite effective in the case of multidimensional scaling using the SMACOF algorithm, and in the case of nonlinear filtering using the Beltrami diffusion framework.

While dependent on the specific nonlinear problem at hand, and while there are no global convergence bounds for these nonlinear problems, we demonstrated these methods work well in the case of several typical applications of MDS, making them an attractive option for accelerating the solution of MDS and related problems. In the case of Beltrami filtering, there was no need in practice to safeguard these methods from divergence due to nonlinear behavior of the process involved. This stages extrapolation techniques as candidates for application in the solution of various problems in computer vision and image processing.

In the field of dimensionality reduction methods, we have presented a new nonlinear dimensionality reduction method. The method is applicable to flat data manifolds with non-trivial topology. We showed that by a careful selection of the geodesics we can robustly flatten non-convex manifolds. In a more general context, the method illustrates one of the main limitations of the Isomap algorithm and related methods, and proved that this limitation can be overcome. Since the proposed method uses global information it is less sensitive to noise than methods that use local distances between the data points, as confirmed

in our experiments. This sets the ground for new algorithms that will enjoy the relative robustness of the global geodesics-based methods of dimensionality reduction compared to locally acting ones, without being hindered by their limitations.

Appendix A

Proof of Proposition 1

Let $(i,j) \in \bar{P}_1$. To prove the proposition, it is sufficient to show that the pair of points (i,j) is consistent, i.e., $(i,j) \in P$. Let $c_{\mathcal{M}}(\mathbf{z}_i, \mathbf{z}_j)$ be the geodesic connecting $\mathbf{z}_i$ and $\mathbf{z}_j$ in $\mathcal{M}$, and let $c_{\mathcal{C}}(\mathbf{x}_i, \mathbf{x}_j)$ be its image under φ^{-1} in $\mathcal{C}$. Since $c_{\mathcal{M}}(\mathbf{z}_i, \mathbf{z}_j) \cap \partial\mathcal{M} = \emptyset$ and because of the isometry, $c_{\mathcal{C}}(\mathbf{x}_i, \mathbf{x}_j) \subset \text{int}(\mathcal{C})$, where $\text{int}(\mathcal{C})$ denotes the interior of $\mathcal{C}$.

Assume that (i,j) is inconsistent. This implies that $d_{\mathcal{C}}(\mathbf{x}_i, \mathbf{x}_j) \neq d_{\mathbb{R}^m}(\mathbf{x}_i, \mathbf{x}_j)$. Because of the way the geodesic metric is defined, $d_{\mathcal{C}}(\mathbf{x}_i, \mathbf{x}_j) > d_{\mathbb{R}^m}(\mathbf{x}_i, \mathbf{x}_j)$. The uniqueness of $c_{\mathbb{R}^m}(\mathbf{x}_i, \mathbf{x}_j)$ states that $c_{\mathcal{C}}(\mathbf{x}_i, \mathbf{x}_j)$ cannot be a straight line (there exists a single straight line connecting each two points in Euclidean geometry).

Therefore, there exits a point $\mathbf{x} \in c_{\mathcal{C}}(\mathbf{x}_i, \mathbf{x}_j)$, whose proximity $c_{\mathcal{C}}(\mathbf{x}_i, \mathbf{x}_j)$ is not a straight line. Since $c_{\mathcal{C}}(\mathbf{x}_i, \mathbf{x}_j) \subset \text{int}(\mathcal{C})$, and $\text{int}(\mathcal{C})$ is an open set, there exists a ball $B_\epsilon(\mathbf{x})$ with the Euclidean metric $d_{\mathbb{R}^m}$ around $\mathbf{x}$ of radius $\epsilon > 0$. Let us take two points on the segment of the geodesic within the ball, $\mathbf{x}', \mathbf{x}'' \in c_{\mathcal{C}}(\mathbf{x}_i, \mathbf{x}_j) \cap B_\epsilon(\mathbf{x})$. The geodesic $c_{\mathcal{C}}(\mathbf{x}', \mathbf{x}'')$ coincides with the segment of $c_{\mathcal{C}}(\mathbf{x}_i, \mathbf{x}_j)$ between $\mathbf{x}', \mathbf{x}''$. Yet, this segment is not a straight line, therefore we can shorten the geodesic by replacing this segment with $c_{\mathbb{R}^m}(\mathbf{x}', \mathbf{x}'')$, in contradiction to the fact that $c_{\mathcal{C}}(\mathbf{x}_1, \mathbf{x}_2)$ is a geodesic. Therefore, $(i,j) \in P$.

In the more general case, where $(\mathcal{M}, d_{\mathcal{M}})$ is not isometric to a subregion of a Euclidean space, the criterion defining $\bar{P}_2$ ensures that if the manifold is isometric to a subregion $\mathcal{C}$ of a space $\mathcal{C}'$ with Riemannian metric, we only select geodesics for which the geodesic metric and the metric of $\mathcal{C}'$ restricted to $\mathcal{C}$ identify. This is the case assumed by Proposition 2.

Proof of Proposition 2 Assume we have a pair of points for which the geodesic and the restricted metric on $\mathcal{C}$ are not the same, $(i,j) \notin P$. Clearly,

$$d_{\mathcal{C}}(\mathbf{x}_i, \mathbf{x}_j) > d_{\mathcal{C}'}(\mathbf{x}_i, \mathbf{x}_j), \tag{6.1}$$

because of the way the geodesic metric is defined. On the other hand observe that the geodesic connecting the two points in $\mathcal{C}$ is not equal to the geodesic connecting them in $\mathcal{C}'$. Specifically, the geodesic in $\mathcal{C}'$ must cross the boundary of $\mathcal{C}$ (otherwise the distances would be equal). This results in the inequality

$$d_{\mathcal{C}'}(\mathbf{x}_i, \mathbf{x}_j) > d_{\mathcal{C}}(\mathbf{x}_i, \partial\mathcal{C}) + d_{\mathcal{C}}(\mathbf{x}_j, \partial\mathcal{C}).$$

Where the segments connecting $\mathbf{x}_i$ and $\mathbf{x}_i$ to $\partial\mathcal{C}$ are inside $\text{int}(\mathcal{C})$. Here we assume there is only one excursion outside of $\mathcal{C}$. Combining Inequality 6.1, we obtain

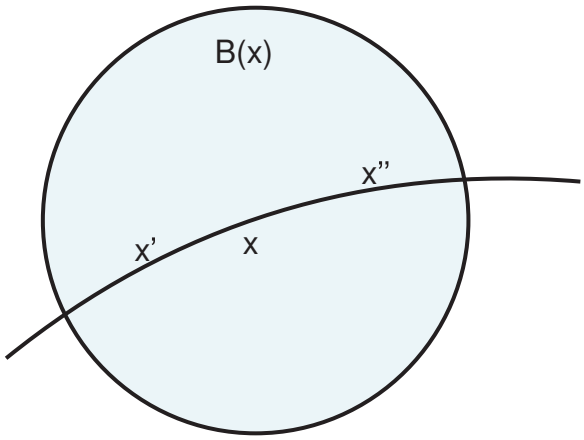

Figure 6.1: An illustrations of points x, x', x'' used in the proof of Proposition 1

$$d_{\mathcal{C}}(\mathbf{x}_i, \mathbf{x}_j) > d_{\mathcal{C}}(\mathbf{x}_i, \partial\mathcal{C}) + d_{\mathcal{C}}(\mathbf{x}_j, \partial\mathcal{C}),$$

and therefore $(i, j) \notin \bar{P}_2$.

Bibliography

[1] Michal Aharon and Ron Kimmel. Representation analysis and synthesis of lip images using dimensionality reduction. *International Journal of Computer Vision*, 67(3):297–312, 2006.

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

[3] Leah Bar, Alexander Brook, Nir A. Sochen, and Nahum Kiryati. Color image deblurring with impulsive noise. In Nikos Paragios, Olivier D. Faugeras, Tony Chan, and Christoph 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] Danny 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.

[5] Brian T. Bartell, Garrison W. Cottrell, and Richard K. Belew. Latent semantic indexing is an optimal special case of multidimensional scaling. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 161–167, New York, NY, USA, 1992. ACM Press.

[6] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Inf. Proc. Sys.*, volume 14, pages 585–591, Cambridge, MA, 2002. MIT Press.

[7] David Belton and Derek D. Lichti. Classification and segmentation of terrestrial laserscanner point clouds using local variance information. *ISPRS Image Engineering and Vision Metrology*, 61(5):307–324, 2007.

[8] Rami Ben-Ari and Nir Sochen. A general framework and new alignment criterion for dense optical flow. *Computer Vision and Pattern Recognition*, 1:529–536, 2006.

[9] Aharon Ben-Tal and Arkadi. S. Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[10] Mira Bernstein, Vin de Silva, John C. Langford, and Joshua B. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Technical report, Stanford University, January 2001.

[11] Andrew Blake and Andrew Zisserman. *Visual Reconstruction*. The MIT Press, London, England, 1987.

[12] Ronald F. Boisvert, Roldan Pozo, Karin Remington, Richard Barrett, and Jack J. Dongarra. The Matrix Market: A web resource for test matrix collections. In Ronald F. Boisvert, editor, *Quality of Numerical Software, Assessment and Enhancement*, pages 125–137, London, 1997. Chapman Hall.

[13] Ingwer Borg and Patrick Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Verlag, New York, 1997.

[14] Terrance E. Boult and John R. Kender. Visual surface reconstruction using sparse depth data. In *Computer Vision and Pattern Recognition*, volume 86, pages 68–76, 1986.

[15] Ulrik Brandes and Christian Pich. Eigensolver methods for progressive multidimensional scaling of large data. In Michael Kaufmann and Dorothea Wagner, editors, *Graph Drawing, Karlsruhe, Germany, September 18-20, 2006*, pages pp. 42–53. Springer, 2007.

[16] Claude Brezinski. *Accélération de la Convergence en Analyse Numérique*. Number 584 in Lecture Notes in Mathematics. Springer-Verlag, Berlin, 1977.

[17] Alex M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Expression-invariant face recognition via spherical embedding. In *Proc. IEEE International Conf. Image Processing (ICIP)*, 2005.

[18] Alex M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Three-dimensional face recognition. *International Journal of Computer Vision*, 64(1):5–30, August 2005.

[19] Alex M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proc. Natl. Acad. Sci. USA*, 103(5):1168–1172, January 2006.

[20] Michael M. Bronstein, Alex M. Bronstein, R. Kimmel, and I. Yavneh. Multigrid multidimensional scaling. *Numerical Linear Algebra with Applications, Special issue on multigrid methods*, 13(2-3):149–171, March-April 2006.

[21] S. Cabay and L.W. Jackson. A polynomial extrapolation method for finding limits and antilimits of vector sequences. *SIAM J. Numer. Anal.*, 13:734–752, 1976.

[22] D. Calvetti, B. Lewis, and L. Reichel. A hybrid GMRES and TV-norm based method for image restoration. *Proceedings of the Society of Photo-Optical Instrumentation Engineers*, 4791:192–200, June 2002.

[23] Matthew Chalmers. A linear iteration time layout algorithm for visualising high-dimensional data. In *IEEE Visualization*, pages 127–132, 1996.

[24] Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. Stability of boundary measures. Research Report 6219, INRIA Sophia Antipolis, June 2007.

[25] Ka W. Cheung and Hing C. So. A multidimensional scaling framework for mobile location using time-of-arrival measurements. *IEEE transactions on signal processing*, 53(2):460–470, 2005.

[26] Ronald R. Coifman, Stephane Lafon, Ann B. Lee, Mauro Maggioni, Boaz Nadler, Frederick Warner, and Steven W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data. *Proc. Natl. Acad. Sci. USA*, 102(21):7426–7431, May 2005.

[27] Pierre Comon. Separation of stochastic processes. In *Workshop on Higher-Order Spectral Analysis*, pages 174–179, June 1989.

[28] Lee G. Cooper. A review of multidimensional scaling in marketing research. *Applied Psychological Measurement*, 7(4):427–450, 1983.

[29] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1990.

[30] R. Courant, K. Friedrichs, and H. Lewy. Uber die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100(1):32–74, 1928.

[31] Payel Das, Mark Mol, Hernan Stamati, Lydia E. Kavraki, , and Cecilia Clementi. Low-dimensional, free-energy landscapes of protein-folding reactions by nonlineardimensionality reduction. *Proc. Natl. Acad. Sci. USA*, 103(26):9885–9890, June 2006.

[32] Lorina Dascal, Guy Rosman, and Ron 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.

[33] Jan de Leeuw. Applications of convex analysis to multidimensional scaling. In J.R Barra, F. Brodeau, G. Romier, and B. Van Custem, editors, *Recent Developments in Statistics*, pages 133–146, Amsterdam, 1977. North Holland Publishing Company.

[34] Jan de Leeuw. Convergence of the majorization method for multidimensional scaling. *Journal of Classification*, 5(2):163–180, September 1988.

[35] Vin de Silva and Joshua B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Inf. Proc. Sys.*, pages 705–712. MIT Press, 2002.

[36] Thomas Deschamps, Ravi Malladi, and Igor 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.

[37] Roberto Moreno Diaz and Alexis Quesada Arencibia, editors. *Coloring of DT-MRI Fiber Traces using Laplacian Eigenmaps*, Las Palmas de Gran Canaria, Spain, February 24–28 2003. Springer Verlag.

[38] Edsger W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[39] Manfredo Perdigao do Carmo. *Riemannian Geometry*. Birkhäuser Verlag, Boston, MA, 1992.

[40] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification and Scene Analysis*. Wiley-Interscience, 2nd edition, 2000.

[41] R.P. Eddy. Extrapolating to the limit of a vector sequence. In P.C.C. Wang, editor, *Information Linkage Between Applied Mathematics and Industry*, pages 387–396, New York, 1979. Academic Press.

[42] Asi Elad and Ron Kimmel. On bending invariant signatures for surfaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(10):1285–1295, 2003.

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

[44] Yuval Eldar, Micha Lindenbaum, Moshe Porat, and Yehoshua Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, September 1997.

[45] Christos Faloutsos and King-Ip Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In Michael J. Carey and Donovan A. Schneider, editors, *International Conference on Management of Data*, pages 163–174, San Jose, California, 22–25 May 1995.

[46] Ronald A. Fisher. The systematic location of genes by means of crossover observations. *The American Naturalist*, 56:406–411, 1922.

[47] Daniel Freedman. Efficient simplicial reconstructions of manifolds from their samples. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(10):1349–1357, October 2002.

[48] Emden R. Gansner, Yehuda Koren, and Stephen C. North. Graph drawing by stress majorization. In János Pach, editor, *Graph Drawing*, volume 3383 of *Lecture Notes in Computer Science*, pages 239–250. Springer, 2004.

[49] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, London, third edition, 1996.

[50] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

[51] Meenakshisundaram Gopi. On sampling and reconstructing surfaces with boundaries. In *CCCG*, pages 49–53, 2002.

[52] Carrie Grimes and David L. Donoho. When does Isomap recover the natural parameterization of families of articulates images? Technical Report 2002-27, Department of Statistics, Stanford University, Stanford, CA 94305-4065, 2002.

[53] Carrie Grimes and David L. Donoho. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. Natl. Acad. Sci. USA*, 100(10):5591–5596, May 2003.

[54] Patrick J. F. Groenen, Willem J. Heiser, and Jacqueline J. Meulman. Global optimization in least-squares multidimensional scaling by distance smoothing. *Journal of Classification*, 16(2):225–254, 1999.

[55] Bertil Gustafsson, Heinz Kreiss, and Joseph Oliger. *Time dependent problems and difference methods*. Wiley, New York, 1995.

[56] Louis Guttman. A general nonmetric technique for finding the smallest coordinate space for a configuration of points. *Psychometrika*, 33:469–506, 1968.

[57] Gideon Guy and Gérard Medioni. Inference of surfaces, 3D curves, and junctions from sparse, noisy, 3D data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(11):1265–1277, November 1997.

[58] Aamer Haque and Gary A. Dilts. Three-dimensional boundary detection for particle methods. *J. Comput. Phys.*, 226(2):1710–1730, 2007.

[59] Jeanny Herault and Bernard Ans Christian Jutten. Detection de grandeurs primitives dans un message composite par une architecture de calcul neuromimetique en apprentissage non supervise. In *GRETSI*, volume 10e, pages 1017–1022, May 1985.

[60] Yoshi hiro Taguchi and Yoshitsugu Oono. Relational patterns of gene expression via non-metric multidimensional scaling analysis. *Bioinformatics*, 21(6):730–740(11), march 2005.

[61] Dorit Hochbaum and David Shmoys. A best possible approximation for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[62] Nan Hu, Weimin Huang, and Surendra Ranganath. Robust attentive behavior detection by non-linear head pose embedding and estimation. In Ales Leonardis, Horst Bischof, and Axel Pinz, editors, *Proceedings of the 9th European Conference on Computer Vision (ECCV)*, volume 3953, pages 356–367, Graz, Austria, May 2006. Springer.

[63] Anthony Kearsley, Richard Tapia, and Michael W. Trosset. The solution of the metric stress and sstress problems in multidimensional scaling using Newton's method. *Computational Statistics*, 13(3):369–396, 1998.

[64] Yosi Keller, Stephane Lafon, and Michael Krauthammer. Protein cluster analysis via directed diffusion. In *The fifth Georgia Tech International Conference on Bioinformatics*, November 2005.

[65] Ron Kimmel, Ravi Malladi, and Nir 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.

[66] Ron Kimmel and James A. Sethian. Computing geodesic paths on manifolds. *Proc. Natl. Acad. Sci. USA*, 95(15):8431–8435, July 1998.

[67] Jospeh B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[68] Stehpane Lafon. *Diffusion Maps and Geometric Harmonics*. Ph.D. dissertation, Graduate School of Yale University, May 2004.

[69] Jan Leeuw. Differentiability of Kruskal's stress at a local minimum. *Psychometrika*, 49(1):111–113, March 1984.

[70] Jan De Leeuw. Accelerated least squares multidimensional scaling. Technical Report 493, University of California, Los Angeles, Department of Statistics, June 2006.

[71] Cecilio Mar-Molinero and Carlos Serrano-Cinca. Bank failure: a multidimensional scaling approach. *The European Journal of Finance*, 7(2):165–183, June 2001.

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

[73] Philippos Mordohai and Gérard Medioni. Unsupervised dimensionality estimation and manifold learning in high-dimensional spaces by tensor voting. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 798–803, 2005.

[74] Alistair Morrison, Greg Ross, and Matthew Chalmers. Fast multidimensional scaling through sampling, springs and interpolation. *Information Visualization*, 2(1):68–77, 2003.

[75] Philip H. Frances Patrick Groenen. Visualizing time-varying correlations across stock markets. *Journal of Empirical Finance*, 7:155–172, 2000.

[76] Karl Pearson. On lines and planes of closest fit to systems of points in space. *London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. Sixth Series.

[77] John C. Platt. Fastmap, metricmap, and landmark MDS are all nystrom algorithms. Technical Report MSR-TR-2004-26, Microsoft Research (MSR), September 2004.

[78] Robert Pless. Using Isomap to explore video sequences. In *IEEE International Conference on Computer Vision*, pages 1433–1440, Nice, France, October 2003.

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

[80] Keith T. Poole. Nonparametric unfolding of binary choice data. *Political Analysis*, 8(3):211–237, March 2000.

[81] Guy Rosman, Alex M. Bronstein, Michael M. Bronstein, and Ron Kimmel. Topologically constrained isometric embedding. In *Proc. Conf. on Machine Learning and Pattern Recognition (MLPR)*, 2006.

[82] Guy Rosman, Alex M. Bronstein, Michael M. Bronstein, Avram Sidi, and Ron Kimmel. Fast multidimensional scaling using vector extrapolation. *SIAM J. Sci. Comput.*, 2 2008. Submitted.

[83] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

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

[85] A. Salam. Non-commutative extrapolation algorithms. *Numerical Algorithms*, 7(2–4):225–251, September 1994.

[86] A. Salam and P.R. Graves-Morris. On the vector epsilon-algorithm for solving linear systems of equations. *Numerical Algorithms*, 29:229–247(19), March 2002.

[87] J. R. Schmidt. On the numerical solution of linear simultaneaous equations by an iterative method. *Phil. Mag.*, 7(32):369–383, 1941.

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

[89] Eric. L. Schwartz, Alan Shaw, and Estarose Wolfson. A numerical solution to the generalized mapmaker's problem: Flattening nonconvex polyhedral surfaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11:1005–1008, November 1989.

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

[91] Daniel Shanks. Non-linear transformations of divergent and slowly convergent sequences. *Journal of Mathematics and Physics*, 34:1–42, 1955.

[92] A. Sidi. Convergence and stability properties of minimal polynomial and reduced rank extrapolation algorithms. *SIAM J. Numer. Anal.*, 23:197–209, 1986. Originally appeared as NASA TM-83443 (1983).

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

[94] 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.

[95] A. Sidi. Convergence of intermediate rows of minimal polynomial and reduced rank extrapolation tables. *Numer. Algorithms*, 6:229–244, 1994.

[96] A. Sidi and J. Bridger. Convergence and stability analyses for some vector extrapolation methods in the presence of defective iteration matrices. *J. Comp. Appl. Math.*, 22:35–61, 1988.

[97] A. Sidi and M.L. Celestina. Convergence acceleration for vector sequences and applications to computational fluid dynamics. Technical Report NASA TM-101327, NASA Lewis Research Center, 1988.

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

[99] 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).

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

[101] Vin De Silva and Joshua B. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Stanford University, 2004.

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

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

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

[105] Nir Sochen, Ron Kimmel, and Alfred M. Bruckstein. Diffusions and confusions in signal and image processing. *Journal of Mathematical Imaging and Vision*, 14(3):195–209, 2001.

[106] Nir Sochen, Ron Kimmel, and Ravi 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.

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

[108] Alon Spira, Ron Kimmel, and Nir Sochen. Efficient Beltrami flow using a short time kernel. *IEEE Trans. Image Process.*, 16:1628–1636, 2007.

[109] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

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

[111] Wai-Shun Tong, Chi-Keung Tang, Phillipos Mordohai, and Gérard Medioni. First order augmentation to tensor voting for boundary inference and multiscale analysis in 3d. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):594–611, May 2004.

[112] Warren S. Torgerson. Multidimensional scaling I. theory and method. *PSym*, 17:401–419, 1952.

[113] Michael Trosset and Rudolf Mathar. On the existence of nonglobal minimizers of the stress criterion for metric multidimensional scaling. *American Statistical Association: Proceedings Statistical Computing Section*, pages 158–162, november 1997.

[114] Shusaku Tsumoto and Shoji Hirano. Visualization of rule's similarity using multidimensional scaling. In *ICDM*, pages 339–346, 2003.

[115] Jason Tsong-Li Wang, Xiong Wang, King-Ip Lin, Dennis Shasha, Bruce A. Shapiro, and Kaizhong Zhang. Evaluating a class of distance-mapping algorithms for data mining and clustering. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 307–311, New York, NY, USA, 1999. ACM Press.

[116] Kilian Q. Weinberger, Ben D. Packer, and Laurence K. Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, Barbados, January 2005.

[117] Kilian Q. Weinberger and Laurence K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 988–995, Washington D.C., 2004. IEEE Computer Society.

[118] P. Wynn. Acceleration techniques for iterated vector and matrix problems. *Mathematics of Computation*, 16(79):301–322, July 1962.

[119] Tynia Yang, Jinze Liu, Leonard McMillan, and Wei Wang. A fast approximation to multidimensionalscaling. In *IEEE workshop on Computation Intensive Methods for Computer Vision*, 2006.

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

[121] Gil Zigelman, Ron Kimmel, and Nahum Kiryati. Texture mapping using surface flattening via multidimensional scaling. *IEEE Transactions on Visualization and Computer Graphics*, 8(2):198–207, 2002.