NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2006; **13**:149–171
Published online 15 February 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nla.475

Multigrid multidimensional scaling

M. M. Bronstein*,†, A. M. Bronstein, R. Kimmel and I. Yavneh

Department of Computer Science, Technion, Israel Institute of Technology, Haifa 32000, Israel

SUMMARY

Multidimensional scaling (MDS) is a generic name for a family of algorithms that construct a configuration of points in a target metric space from information about inter-point distances measured in some other metric space. Large-scale MDS problems often occur in data analysis, representation and visualization. Solving such problems efficiently is of key importance in many applications.

In this paper we present a multigrid framework for MDS problems. We demonstrate the performance of our algorithm on dimensionality reduction and isometric embedding problems, two classical problems requiring efficient large-scale MDS. Simulation results show that the proposed approach significantly outperforms conventional MDS algorithms. Copyright © 2006 John Wiley & Sons, Ltd.

KEY WORDS: multigrid; multiresolution; multidimensional scaling; isometric embedding; SMACOF; BFGS; face recognition; bending-invariant canonical form; dimensionality reduction

1. INTRODUCTION

Multidimensional scaling (MDS) is a generic name for a family of algorithms that construct a configuration of points in a target metric space from information about inter-point distances (*dissimilarities*), measured in some other metric space. The spectrum of MDS applications is very broad and ranges from stock market analysis [1] to computational chemistry [2] and breast cancer diagnosis [3]. MDS is widely used in dimensionality reduction, data analysis and visualization applications, when, for example, one wishes to understand complicated high-dimensional data structures and represent them by low-dimensional ones [4, 5].

From the point of view of the underlying optimization problem, MDS is known to be hard, as it involves a non-linear non-convex objective function requiring heavy computation, whose gradient and Hessian are also hard to compute, and moreover, the Hessian is dense. Current

*Correspondence to: M. M. Bronstein, Department of Computer Science, Israel Institute of Technology, Technion, Haifa 32000, Israel.
†E-mail: bronstein@ieee.org

Contract/grant sponsor: The Israel Ministry of Science Infrastrucutre; contract/grant number: 01-01-499
Contract/grant sponsor: The Israel Science Foundation; contract/grant numbers: 738/04, 48/02
Contract/grant sponsor: The European FP 6 NoE; contract/grant number: 507752 (MUSCLE)

Received 15 May 2005
Revised 26 October 2005
Accepted 7 November 2005

MDS algorithms are notoriously slow, and limited to small data sets. Efficiently solving large-scale MDS problems arising in numerous applications has been a challenge for a long time.

In many cases, the dissimilarities can be thought of as geodesic distances measured on a smooth Riemannian manifold, and the underlying geometry can be used to advantage. This particular setting of the MDS problem is known as the *isometric embedding* problem, and from the geometric point of view, it is the problem of representing the intrinsic structure of a Riemannian manifold in some other metric space. The isometric embedding approach was used by Schwartz *et al.* [6] to flatten convoluted 3D surface of the brain cortex in order to study its structure as a 2D image. A similar approach was used in References [7, 8] for texture mapping. Elad and Kimmel [9] proposed embedding surfaces into a higher-dimensional Euclidean space in order to compute their isometry-invariant signatures. This approach was applied in References [10, 11] to the problem of 3D face recognition.

In Reference [12] we presented a multigrid approach for efficient isometric embedding of surfaces arising in the 3D face-recognition application. Here, we describe a general multigrid framework for MDS algorithms. Our main focus are isometric embedding problems, though we discuss and exemplify general MDS problems as well.

The organization of the paper is as follows: In Section 2 we give a formal definition to the isometric embedding problem and its discrete setting. We show applications where such problems arise. Section 3 shows how MDS is used to perform the isometric embedding. First- and second-order MDS methods are discussed. In Section 4 we introduce the full approximation scheme (FAS) multigrid MDS (hereinafter, MG-MDS) algorithm for the isometric embedding problem. Section 5 discusses its extension to a generic MDS problem. Section 6 is dedicated to experimental results. Our approach is exemplified on two problems: isometric embedding of a complicated Riemannian manifold ('Swiss roll') and of a facial surface. In addition, we apply a general MDS algorithm to the problem of dimensionality reduction of high-dimensional binary data. Section 7 concludes the paper.

2. ISOMETRIC EMBEDDING

Let $\mathscr{S}$ and $\mathscr{Q}$ be two complete smooth Riemannian manifolds, with geodesic distances induced by the Riemannian geometry and denoted by $\delta(\xi_1, \xi_2)$ and $d(x_1, x_2)$ for all $\xi_1, \xi_2 \in \mathscr{S}$ and $x_1, x_2 \in \mathscr{Q}$, respectively. A mapping $\varphi : \mathscr{S} \to \mathscr{Q}$ is called an *isometry* if it preserves the geodesic distances, i.e.

$$\delta(\xi_1, \xi_2) = d(\varphi(\xi_1), \varphi(\xi_2)) \quad \forall \xi_1, \xi_2 \in \mathscr{S} \tag{1}$$

The manifolds $\mathscr{S}$ and $\mathscr{Q}$ are said to be *isometric* in this case and have the same *intrinsic geometry* (properties associated with the geodesic distances.)[‡]

Let us assume that we wish to study the intrinsic geometry of the manifold $\mathscr{S}$. Yet, the manifold can be immersed in a complicated way into some ambient space, and thus can have a very complicated extrinsic geometry. In order to get rid of the extrinsic geometry, we map

[‡]Note that here we understand the terms *isometry* and *intrinsic geometry* in the context of metric rather than Riemannian geometry, i.e. terms associated with the geodesic distances and not with the Riemannian metric tensor.

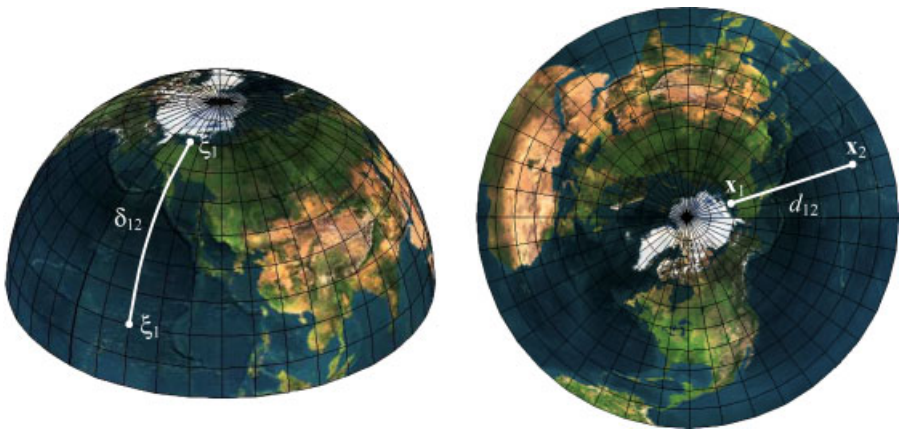

Figure 1. Illustration of the isometric embedding problem arising in cartography. Left: the Earth spherical surface (shown here is the upper hemisphere). Right: a planar map of the Earth created by embedding the spherical surface into $\mathbb{R}^2$, such that the geodesic distances are replaced with Euclidean ones.

$\mathscr{S}$ in an isometric manner into another manifold $\mathscr{Q}$, which has a simple intrinsic geometry. The problem of finding a representation of the intrinsic geometry of a Riemannian manifold using the intrinsic geometry of some other manifold is called the *isometric embedding problem*. The manifold $\mathscr{Q}$ into which the mapping is performed is usually referred to in this context as the *embedding space*, and φ as an *isometric embedding*. Usually, the embedding space can be chosen to our discretion. The most popular choice is a finite-dimensional Euclidean space $\mathbb{R}^m$ (though other choices are also possible [13–17]), which will be assumed hereinafter in this paper.

As an illustration, think of the problem of Earth mapping (Figure 1). We wish to map the spherical surface of the Earth (2-manifold) into a plane, preserving in the best possible way the distances between geographic objects. An optimal mapping would be an isometric embedding of the sphere into $\mathbb{R}^2$. However, it can be easily shown that a sphere is not isometrically embeddable into a Euclidean space of any finite dimension [5, 18]. Furthermore, it appears in most cases that a general non-trivial manifold is not isometrically embeddable into $\mathbb{R}^m$ [5]. We must therefore keep in mind this fact, which implies that since we are unable to find an exact isometric embedding, we have to resort to a near-isometric embedding, optimal in some sense. The measure of the geodesic distances distortion by such an embedding is referred to as the *embedding error*. The definition of the embedding error and ways of its minimization will be discussed in Section 3.

2.1. Discrete isometric embedding

In the discrete setting, we have the manifold $\mathscr{S}$ sampled at N points $\xi_1, \ldots, \xi_N$, and the geodesic distances between all the points are given and can be represented as a symmetric $N \times N$ dissimilarity matrix with zero-diagonal and non-negative elements

$$\delta_{ij} = \delta(\xi_i, \xi_j); \quad i, j = 1, \ldots, N \tag{2}$$

Isometric embedding in this case is defined as a mapping between two finite metric spaces,

$$\varphi : (\{\xi_1, \ldots, \xi_N\} \subset \mathscr{S}, \boldsymbol{\Delta}) \to (\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^m, \mathbf{D}) \tag{3}$$

trying to achieve $d_{ij} \approx \delta_{ij}$. Here $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$ denotes the distances in the Euclidean embedding space. The embedding error can be thought of as a discrepancy between all δ_{ij} and d_{ij}.

2.2. Applications of the isometric embedding

In Reference [9], Elad and Kimmel showed an application for the isometric embedding in the context of pattern recognition. By embedding surfaces into a low-dimensional Euclidean space, it is possible to get rid of their extrinsic geometry and obtain an intrinsic geometric representation. The authors coined the term *bending-invariant canonical form* referring to such a representation. The use of canonical forms allows recognizing objects independently of how they are bent, assuming that the deformation is isometric or near isometric (see Figure 2).

In References [10, 11], Bronstein *et al.* applied this approach to the problem of 3D face recognition. Under the assumption that facial expressions can be approximated as near-isometric deformations of the facial surface, isometric embedding is used to construct expression-invariant representations of facial surfaces (Figure 3). In the 3D face-recognition system developed at the Department of Computer Science, Technion [10], much effort was focused on fast computation of an expression-invariant representation of the face. Current implementation in C on an AMD Opteron machine allows the overall recognition process, including 3D surface acquisition, preprocessing and canonical form computation, to be

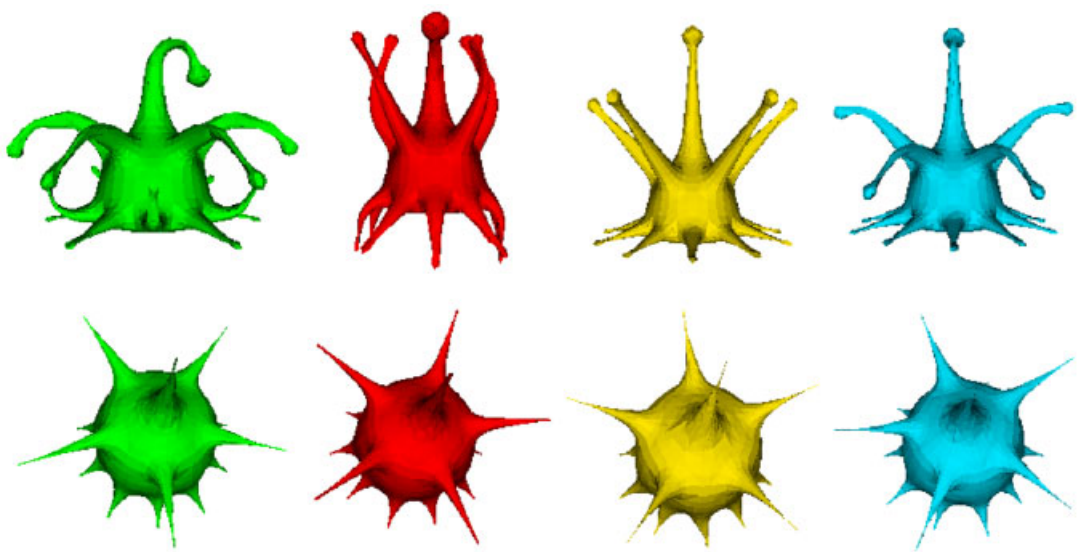

Figure 2. Near-isometric transformations of a deformable object (first row) and the corresponding canonical forms (second row). Image reproduced from Reference [9] with the permission of the authors.

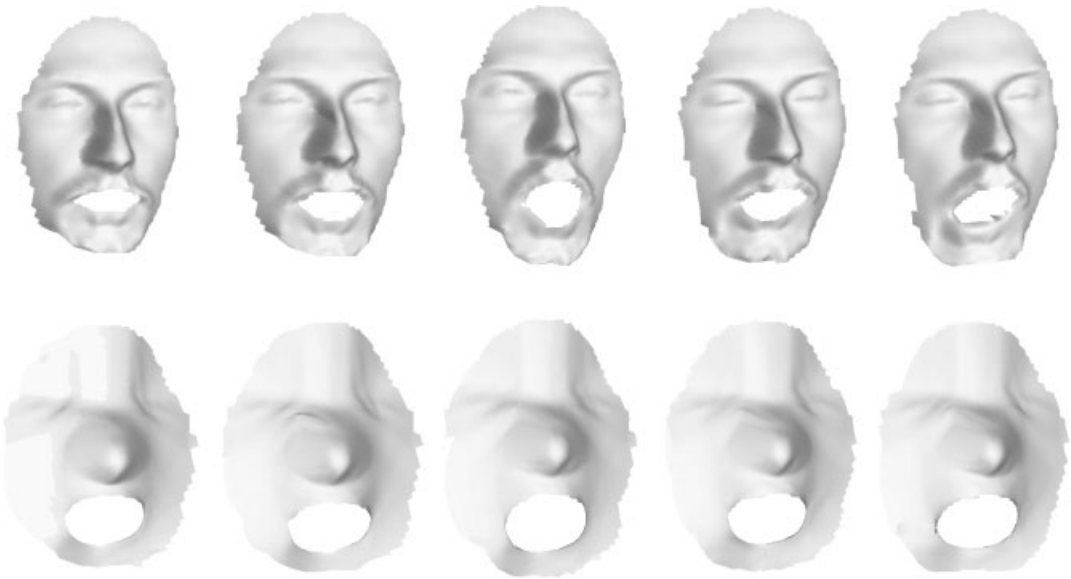

Figure 3. Example of canonical forms (second row) of facial surfaces (first row) and their insensitivity to facial expressions.

performed in less than 6 s. The MDS stage takes about 2 s,[§] which constitutes over 30% of the whole processing time. The MDS stage constitutes therefore a bottleneck of the 3D face-recognition algorithm. Being able to reduce the MDS computation time, one can benefit either from faster performance or the possibility to perform more accurate embedding using more surface points or allowing more iterations, which increases the recognition accuracy.

3. MULTIDIMENSIONAL SCALING

Let $\mathbf{\Delta}$ be a symmetric $N \times N$ matrix of geodesic distances δ_{ij} measured between N points on a Riemannian manifold $\mathscr{S}$. Our goal is to find a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ in $\mathbb{R}^m$, such that the embedding error, i.e. the discrepancy between δ_{ij} and d_{ij}, is minimal. An error criterion commonly used in MDS literature is the *stress* function, given by

$$s(\mathbf{X}; \mathbf{\Delta}, \mathbf{W}) = \sum_{i=1}^{N} \sum_{j=i+1}^{N} w_{ij}(d_{ij}(\mathbf{X}) - \delta_{ij})^2 \tag{4}$$

Here $\mathbf{X} = (x_{ij})$ is the $N \times m$ matrix of co-ordinates of the resulting points in $\mathbb{R}^m$. Following Elad and Kimmel, we refer to $\mathbf{X}^* = \arg\min_{\mathbf{X}} s(\mathbf{X}; \mathbf{\Delta}, \mathbf{W})$ as the *canonical form* of $\mathscr{S}$.[¶] The

[§]On a facial surface containing 2500 points, carried out using 40 iterations of the SMACOF algorithm (see Section 3.2). Note that the implementation of the algorithms in this paper (shown in Section 6) is in MATLAB and is about 100 times slower.

[¶]The term *configuration matrix* is used in the MDS community when referring to $\mathbf{X}$ [19].

symmetric $N \times N$ matrix of weights $\mathbf{W} = (w_{ij})$ determines the relative contribution of distances to the embedding error criterion. The use of weights allows to give more importance to some distances, or alternatively, exclude some distances from the stress computation. We will distinguish between the *weighted stress*, where values of w_{ij} are specified, and the *non-weighted stress*, where $w_{ij} = 1$.

3.1. General remarks

Note that the only input in the MDS problem is the matrix $\mathbf{\Delta}$ (and the matrix $\mathbf{W}$ in case of weighted stress). The solution of the MDS problem is *not unique*. Any isometric transformation in the embedding space applied to the canonical form $\mathbf{X}^*$ is a minimizer of the stress. In $\mathbb{R}^m$, such transformations are translations, reflections and rotations. The canonical form is computed by iterative minimization of $s(\mathbf{X}; \mathbf{\Delta}, \mathbf{W})$ w.r.t. $\mathbf{X}$. In many applications, reaching the exact minimum is not required or cannot be achieved due to real-time limitations (e.g. in 3D face recognition). Stopping criteria commonly used in MDS algorithms include reaching a predetermined stress value (if such is known *a priori*), or when the decrease of the stress function at two subsequent iterations is below some threshold [4].

MDS is in all respects a hard optimization problem. The stress is a non-linear non-convex function w.r.t. $\mathbf{X}$, and convex optimization algorithms do not guarantee convergence to a global minimizer of $s(\mathbf{X})$ and are liable to converge to local minima. Using a good initialization is therefore important. One possibility to construct such an initialization is using the Young–Torgerson *classical scaling* algorithm [20, 21]. Another possibility is the *multistart* approach, consisting of running an MDS algorithm using a small number of iterations from many different random starting configurations and choosing the one with the lowest stress. However, both these methods require extensive preprocessing and usually are computationally expensive [4].

In the isometric embedding problem, the extrinsic geometry of the manifold (i.e. the locations of the points $\xi_1, \ldots, \xi_N$ in the ambient space) can be sometimes used to construct a good initialization. In the particular case discussed here in the context of the 3D face-recognition application, $\mathbb{R}^3$ is assumed as the ambient and the embedding space. Therefore, we can use the Cartesian co-ordinates of $\xi_1, \ldots, \xi_N$ in $\mathbb{R}^3$ as the initialization of MDS [9]. In Section 6.2, we demonstrate empirically that such an initialization works particularly well for facial surfaces, since it appears to be sufficiently close to the optimal solution.

Another characteristic of the MDS problem is the high computational complexity of the stress function and its derivatives (see Table I); this stems from the fact that the matrix $\mathbf{\Delta}$ is dense. This property will result in some limitations on the choice of optimization algorithms for stress minimization.

3.2. Gradient-descent-type algorithms

The easiest choice of an optimization algorithm in the MDS problem are first-order, gradient-descent-type methods, in which the direction at the $(k+1)$st iteration is $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} - \alpha^{(k)} \nabla s(\mathbf{X}^{(k)})$. The gradient of $s(\mathbf{X})$ with respect to $\mathbf{X}$ can be written in matrix form [4] as

$$\nabla s(\mathbf{X}; \mathbf{\Delta}, \mathbf{W}) = 2\mathbf{V}\mathbf{X} - 2\mathbf{B}(\mathbf{X}; \mathbf{\Delta}, \mathbf{W})\mathbf{X} \tag{5}$$

Table I. Approximate computational complexity (in terms of multiplication operations) of the stress function $s(\mathbf{X})$ and its derivatives $\nabla s(\mathbf{X})$, $\nabla^2 s(\mathbf{X})$. N denotes the number of points, m denotes the embedding space dimension. C_s denotes the complexity of square-root computation in terms of multiplication operations; typically, $C_s \approx 25$.

	Weighted	Non-weighted
s	$\frac{1}{2}N(N-1)(3+C_s)$	$\frac{1}{2}N(N-1)(2+C_s)$
∇s	$\frac{1}{2}N(N-1)(3+C_s)+2Nm$	$\frac{1}{2}N(N-1)(2+C_s)+Nm$
$\nabla^2 s$	$\frac{1}{2}N(N-1)(8+C_s)m^2$	$\frac{1}{2}N(N-1)(7+C_s)m^2$

where $\mathbf{V}$ is a matrix with elements

$$v_{ij} = \begin{cases} -w_{ij} & \text{if } i \neq j \\ \sum_{k \neq i} w_{ik} & \text{if } i = j \end{cases} \tag{6}$$

and $\mathbf{B}$ is a matrix with elements

$$b_{ij} = \begin{cases} -w_{ij}\delta_{ij} d_{ij}^{-1}(\mathbf{X}) & \text{if } i \neq j \text{ and } d_{ij}(\mathbf{X}) \neq 0 \\ 0 & \text{if } i \neq j \text{ and } d_{ij}(\mathbf{X}) = 0 \\ -\sum_{k \neq i} b_{ik} & \text{if } i = j \end{cases} \tag{7}$$

The step size $\alpha^{(k)}$ is computed by means of *line search*. However, in our problem the computation complexity of the stress and its gradient is roughly the same ($\mathcal{O}(N^2)$; see Table I). Consequently, the complexity of exact line search becomes excessive in large-scale problems. An important conclusion is therefore that we would tend to favour optimization methods that use a constant step size, and use line search only as a safeguard, i.e. when the step increases the stress function value.

In Reference [22], a scaled gradient-descent algorithm has been proposed for the MDS problem, observing that the first-order optimality condition, $\nabla s(\mathbf{X}^*) = 0$, can be written as $\mathbf{V}\mathbf{X}^* = \mathbf{B}(\mathbf{X}^*)\mathbf{X}^*$, where $\mathbf{X}^*$ denotes a minimum of $s(\mathbf{X})$. It follows that the sequence

$$\mathbf{X}^{(k+1)} = \mathbf{V}^{\dagger}\mathbf{B}(\mathbf{X}^{(k)})\mathbf{X}^{(k)} \tag{8}$$

converges to a local minimum of $s(\mathbf{X})$ (here $\dagger$ denotes matrix pseudoinverse). The algorithm using this multiplicative update is called SMACOF [4, 23, 24]. It can be easily shown to be equivalent to weighted gradient descent with constant step size

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} - \tfrac{1}{2}\mathbf{V}^{\dagger}\nabla s(\mathbf{X}^{(k)}) \tag{9}$$

and if a non-weighted stress is used, it is essentially a gradient descent with constant step size

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} - \frac{1}{2N}\nabla s(\mathbf{X}^{(k)}) \tag{10}$$

SMACOF is widely used for large-scale MDS problems as it appears to be practically one of the most efficient MDS algorithms [4]. Particularly, it was adopted for computing the expression-invariant representations of faces in the Technion 3D face-recognition system

mentioned above. Its disadvantage is slow convergence in the proximity of the minimum, which is inherent to all first-order methods. In the Technion 3D face-recognition system, a preset number of 40 SMACOF iterations is used to compute the canonical forms of faces. Being certainly a compromise in accuracy (usually, more than 40 SMACOF iterations are required to achieve the minimum), this stopping criterion allows real-time performance.

3.3. Newton-type algorithms

Some recent papers (e.g. Reference [19]) propose using second-order (Newton-type) algorithms for stress minimization. A basic Newton iteration has the form $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} - \mathscr{H}^{-1}\nabla s(\mathbf{X}^{(k)})$, where $\mathscr{H}$ denotes the Hessian, which is a fourth-order tensor in this notation. If $\mathbf{X}$ is column stacked into an $Nm \times 1$ vector, the Hessian can be represented by an $Nm \times Nm$ matrix, consisting of m^2 blocks of size $N \times N$. Each block is given by

$$\mathbf{H}^{kl} = \begin{cases} \tilde{\mathbf{H}}^{kl} & \text{if } k \neq l \\ \tilde{\mathbf{H}}^{kl} + 2(\mathbf{V} - \mathbf{B}) & \text{if } k = l \end{cases}; \quad 1 \leqslant k, \ l \leqslant m \tag{11}$$

where

$$\tilde{h}_{ij}^{kl} = \begin{cases} -2w_{ij}(x_{ik} - x_{jk})(x_{il} - x_{jl})\delta_{ij}d_{ij}^{-3}(\mathbf{X}) & \text{if } i \neq j \text{ and } d_{ij}(\mathbf{X}) \neq 0 \\ 0 & \text{if } i \neq j \text{ and } d_{ij}(\mathbf{X}) = 0 \\ -\sum_{n \neq i} \tilde{h}_{in}^{kl} & \text{if } i = j \end{cases} \tag{12}$$

Note that when $m = 1$, the Hessian has the form $\mathbf{H}(\mathbf{X}) = 2\mathbf{B}(\mathbf{X}) + 2(\mathbf{V} - \mathbf{B}(\mathbf{X})) = 2\mathbf{V}$.

The Newton method has quadratic convergence, as, in the proximity of the minimum, a function is accurately approximated by a quadratic function. The disadvantage of the Newton method is the relatively high computational complexity required for the Hessian construction and inversion. The Newton algorithm with explicit Hessian inversion appears to be efficient only for small-scale problems (practically, hundreds of points), as the cost of Hessian construction and inversion (requiring $\mathcal{O}(N^3m^3)$ operations) becomes prohibitive with large N.

In order to reduce the cost of Newton iteration, *inexact-* or *truncated-Newton* methods [25–27] can be used. In these algorithms, the Newton direction computation $\mathscr{H}^{-1}\nabla s$ is considered as a linear system of equations and solved using some efficient method, e.g. *conjugate gradients*. Typically, the Hessian does not even have to be explicitly constructed, and only an efficient (exact or approximate) computation of Hessian-vector products of the form $\mathscr{H}\mathbf{X}$ is required. This computation is efficient if the Hessian $\mathscr{H}$ is sparse or has some other convenient structure, or alternatively, can be approximated by an operator with such a structure.

Unfortunately, in our case, though the Hessian is structured, it is not sparse. The diagonals of each block are N times larger than the rest of the elements, which leads to a structure with $2m - 1$ dominant diagonals. The Hessian can be approximated by such a multidiagonal matrix, yet, since the diagonals in each block consist of sums of all the block elements, the construction cost of such a sparse Hessian approximation (or alternatively, the approximate Hessian-vector product cost) is similar to that of the full Hessian. We are not aware of an efficient analytic Hessian approximation in the MDS problem. Yet, it is possible to approximate the Hessian-vector products using finite differences, thus practically computing only the gradients ∇s [26].

Another possible implementation of a truncated-Newton algorithm is using a *multigrid* solver for the Newton system at each iteration [26–30]. Such methods often show performance comparable to FAS multigrid methods, like the one developed in this paper.

3.4. Quasi-Newton algorithms

Another family of algorithms, widely used in problems where the Hessian computation is hard, is called *quasi-Newton* methods. The main idea of these algorithms is to construct an approximate inverse Hessian $\mathscr{L} \approx \mathscr{H}^{-1}$ at each iteration, using gradients from a few previous iterations [26]. Practical experience has shown the *Broyden–Fletcher–Goldfarb–Shanno* (BFGS) scheme to be the best quasi-Newton method in most circumstances. The inverse Hessian approximation update (in matrix representation) in this scheme is given by

$$\mathbf{L}^{(k+1)} = \left(\mathbf{I} - \frac{[\mathbf{Q}^{(k+1)}, \mathbf{Y}^{(k+1)}]}{\langle \mathbf{Q}^{(k+1)}, \mathbf{Y}^{(k+1)} \rangle}\right) \mathbf{L}^{(k)} \left(\mathbf{I} - \frac{[\mathbf{Y}^{(k+1)}, \mathbf{Q}^{(k+1)}]}{\langle \mathbf{Q}^{(k+1)}, \mathbf{Y}^{(k+1)} \rangle}\right) + \frac{[\mathbf{Q}^{(k+1)}, \mathbf{Q}^{(k+1)}]}{\langle \mathbf{Q}^{(k+1)}, \mathbf{Y}^{(k+1)} \rangle} \tag{13}$$

where $\langle \mathbf{Q}, \mathbf{Y} \rangle = \mathrm{trace}(\mathbf{Q}^{\mathrm{T}}\mathbf{Y})$ denotes matrix inner product; $[\mathbf{Q}, \mathbf{Y}] = (q_{ij} \cdot y_{kl})$ is the outer product arranged as an $Nm \times Nm$ matrix, and

$$\begin{aligned} \mathbf{Q}^{(k+1)} &= \mathbf{X}^{(k+1)} - \mathbf{X}^{(k)} \\ \mathbf{Y}^{(k+1)} &= \nabla s(\mathbf{X}^{(k+1)}) - \nabla s(\mathbf{X}^{(k)}) \end{aligned} \tag{14}$$

As an initialization, $\mathbf{L}^{(0)} = \mathbf{I}$ is usually chosen. The BFGS step is obtained directly as

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} - \mathscr{L}^{(k)} \nabla s(\mathbf{X}^{(k)}) \tag{15}$$

Note that here, as in truncated-Newton methods, Hessian-vector product rather than Hessian construction is required. The information necessary for Hessian-vector product computation at kth iteration is $\mathbf{Q}^{(i)}, \mathbf{Y}^{(i)}$, $i = 0, 1, \ldots, k-1$ and their inner products. This information can be efficiently stored. A version of BFGS using this idea is usually referred to as *limited-memory* or L-BFGS [26].

4. MULTIGRID MDS

Our previous results [18] demonstrated that substantial performance improvement can be achieved by using a multiresolution initialization to the embedding problem. Here, we extend these results into a genuine multigrid approach.

Originally, multigrid methods were introduced in the context of differential equations [31–34]. More recently, this framework was adapted to non-linear discrete optimization problems (see, for example, Reference [35]). The optimization problem $\min_{\mathbf{X}} s(\mathbf{X})$ is equivalent to the solution of the non-linear equation $\nabla s(\mathbf{X}) = 0$, arising from the first-order optimality conditions. The spirit of multigrid is to solve the non-linear problem $\nabla s(\mathbf{X}) = 0$ using a sequence of approximate solutions to non-linear problems of the form $\nabla s(\mathbf{X}) = \mathbf{T}$, solved on coarse grids. The term $\mathbf{T}$ arises from the residual transferred from previous levels. In the

optimization problem formulation, we need to minimize functions of the form

$$s(\mathbf{X}) - \langle \mathbf{X}, \mathbf{T} \rangle = s(\mathbf{X}) - \mathrm{trace}(\mathbf{X}^{\mathrm{T}}\mathbf{T}) \tag{16}$$

whose gradient equals $\nabla s(\mathbf{X}) - \mathbf{T}$.

4.1. Modified stress

The second (linear) term makes the function $s(\mathbf{X}) - \mathrm{trace}(\mathbf{X}^{\mathrm{T}}\mathbf{T})$ unbounded. In order to overcome this problem, we introduce the *modified stress*

$$\hat{s}(\mathbf{X}; \boldsymbol{\Delta}, \mathbf{W}) = \sum_{i>j} w_{ij}(d_{ij}(\mathbf{X}) - \delta_{ij})^2 + \lambda \sum_{j=1}^{m} \left(\sum_{i=1}^{N} x_{ij} \right)^2; \quad \lambda > 0 \tag{17}$$

and use it instead of $s(\mathbf{X})$ hereinafter. Having in mind that the solution to the MDS problem is defined up to an isometry in the embedding space, note that such a modification resolves the translation ambiguity by restricting the centre of mass of the resulting canonical form to be at the origin. In the following, the value of $\lambda = 1$ is tacitly assumed. Since the second term is quadratic, the function $\hat{s}(\mathbf{X}) - \mathrm{trace}(\mathbf{X}^{\mathrm{T}}\mathbf{T})$ is bounded. This modification results in an increment by 2λ of every element of the Hessian diagonal blocks, $\mathbf{H}^{kk}$. The computational complexity of the modified stress has the same order as of the original stress function.

4.2. Hierarchy of grids

We use a hierarchy of grids $\Omega_1 \supset \cdots \supset \Omega_R$, constructed in a multiresolution manner, where R is the coarsest resolution level, and denote by N_r the number of grid points at the rth level (such that $N_1 = N$). For simplicity, let us assume the original manifold to be 2D (i.e. a surface). If the surface is given in a parametric form, the grids can be constructed by sampling the parameterization domain. In the frequent case where the parameterization is given on the unit square domain, Cartesian (e.g. dyadic) grids can be used.

If the parameterization domain is complicated (for example, non-convex), or alternatively, if the surface is given in a polyhedral (triangulated) representation, a hierarchy of grids can be constructed using the *farthest point sampling* algorithm [36]. The key idea is progressively refining the grid in the following manner: Start from an arbitrary single source point x_{i_1}, and add a point x_{i_2} located at the largest geodesic distance from the source point. The set of source points is then updated and becomes $\{x_{i_1}, x_{i_2}\}$. The whole process is repeated, such that at the kth iteration the point

$$x_{i_k} = \underset{x_i : i=1,\ldots,N}{\arg\max} \min_{l=1,\ldots,k-1} \delta(x_i, x_{i_l}) \tag{18}$$

is selected. The geodesic distances are computed using an approximate numerical method, e.g. fast marching on triangulated domains [37] or parametric fast marching [38]. Selecting the first $N_R < N_{R-1} < \cdots < N_1$ points out of the obtained set, a hierarchy of grids covering the whole surface is constructed (see example in Figure 4). Such a sampling was used in References [7, 9] for subsampling of the surfaces prior to the construction of canonical forms.

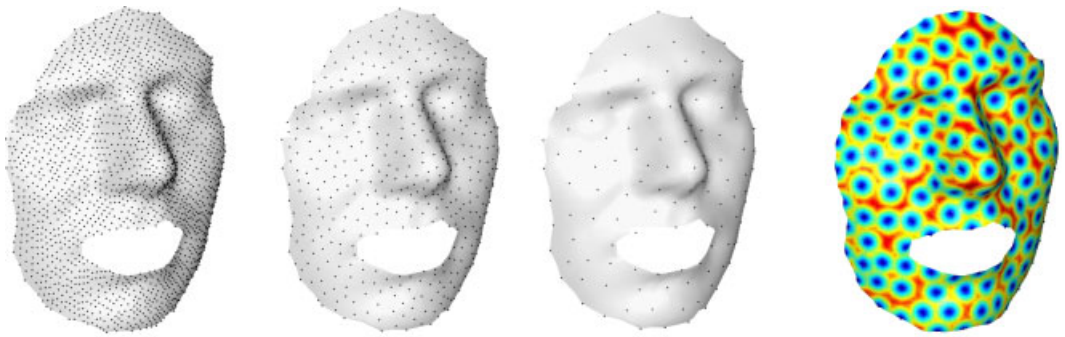

Figure 4. Example of a grid hierarchy construction in the facial surface embedding problem. Left: three resolution levels constructed using the farthest point sampling algorithm. Right: geodesic distances from the coarsest grid points.

4.3. *Coarsening strategy*

The transfer from resolution level r to the coarser level $r+1$ or the finer level $r-1$ is performed using an $N_{r+1} \times N_r$ matrix $\mathbf{P}_r^{r+1}$ (referred to as *restriction* operator in the multigrid literature) and an $N_{r-1} \times N_r$ matrix $\mathbf{P}_r^{r-1}$ (*interpolation* operator), respectively. These matrices are sparse (typically, every row contains from 1 up to 8 non-zero elements) and are often chosen to satisfy

$$\mathbf{P}_r^{r+1} = (\mathbf{P}_{r+1}^r)^{\mathrm{T}} \tag{19}$$

If the surface is given parametrically, the relationships between the points in the parameterization domain are used to construct the interpolation and restriction operators based, for example, on spline. Similarly, in case of a polyhedral representation, the triangulation is employed to determine the neighbours of each grid point and construct $\mathbf{P}_r^{r+1}$, $\mathbf{P}_{r+1}^r$. Attention should be taken on the boundary, ensuring that grid points at finer resolution levels can be interpolated from the coarse ones (this can be achieved, for example, by requiring that the points on finer resolution levels belong to the convex hull of the points on the coarsest level).

The optimization problem is transferred to the next coarser level by applying a restriction operator $\tilde{\mathbf{P}}_r^{r+1}$ (not necessarily equal to $\mathbf{P}_r^{r+1}$) to the matrices $\boldsymbol{\Delta}$ and $\mathbf{W}$ in the following manner:

$$\begin{aligned} \boldsymbol{\Delta}_{r+1} &= \tilde{\mathbf{P}}_r^{r+1} \boldsymbol{\Delta}_r (\tilde{\mathbf{P}}_r^{r+1})^{\mathrm{T}} \\ \mathbf{W}_{r+1} &= \tilde{\mathbf{P}}_r^{r+1} \mathbf{W}_r (\tilde{\mathbf{P}}_r^{r+1})^{\mathrm{T}} \end{aligned} \tag{20}$$

Consequently, we have a hierarchy of problems of the form

$$s_r(\mathbf{X}_r, \mathbf{T}_r) \equiv \hat{s}(\mathbf{X}_r; \boldsymbol{\Delta}_r, \mathbf{W}_r) - \mathrm{trace}(\mathbf{X}_r^{\mathrm{T}} \mathbf{T}_r) \tag{21}$$

that need to be solved at each level.

4.4. Relaxation

We use SMACOF-type iterations for the relaxation stage of the MG-MDS algorithm. The gradient (as opposed to the conventional stress) has two extra terms, attributed to the quadratic penalty in the modified stress function and the linear residual term. In matrix notation, the gradient has the form

$$\nabla \hat{s}(\mathbf{X}; \boldsymbol{\Delta}, \mathbf{W}) = 2\mathbf{V}\mathbf{X} - 2\mathbf{B}(\mathbf{X}; \boldsymbol{\Delta}, \mathbf{W})\mathbf{X} - \mathbf{T} + \lambda \mathbf{Z}(\mathbf{X}) \tag{22}$$

where the matrix $\mathbf{Z}(\mathbf{X})$ is defined in the following way:

$$z_{ij} \equiv \sum_k x_{kj} \tag{23}$$

We use an additive update form like in (9). Figure 5 demonstrates that such a relaxation has the error-smoothing property, which is one of the necessary properties for the success of a multigrid method. Below (left) the error

$$\varepsilon_i(\mathbf{X}^{(0)}, \mathbf{X}^*) \equiv \sqrt{\sum_{j=1}^{m} (x_{ij}^{(0)} - x_{ij}^*)^2} \tag{24}$$

in an isometric embedding problem of a surface given on a 33×33 grid is shown (see Section 6.1). The initialization $\mathbf{X}^{(0)}$ is random. The plot on the right demonstrates the error after a few SMACOF iterations, i.e. $\varepsilon(\mathbf{X}^{(k)}, \mathbf{X}^*)$. Note that the smoothing is slightly deteriorated on the boundary.

From the point of view of computational complexity it may be advantageous to use SMACOF iterations as the relaxation procedure at higher-resolution levels and Newton or quasi-Newton iterations at coarser resolution levels. Yet, in this paper we show only the aforementioned SMACOF-type relaxation.

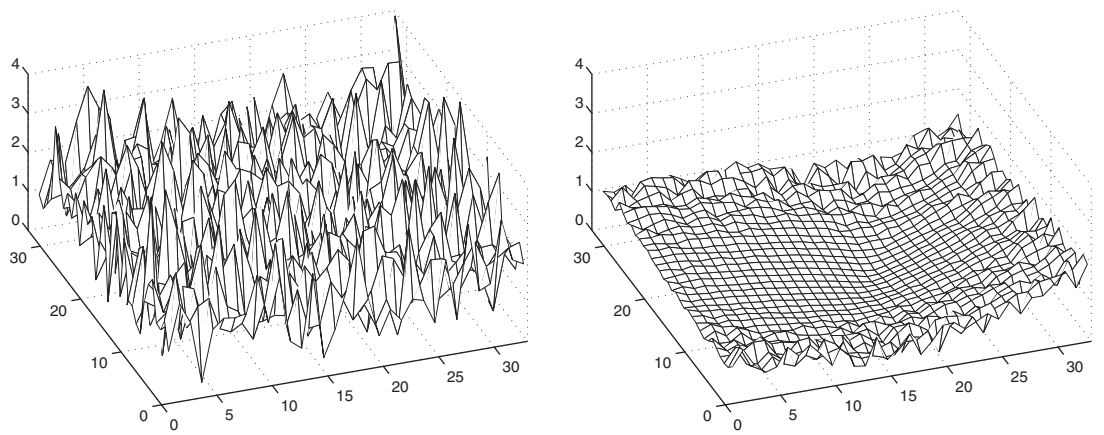

Figure 5. Error-smoothing property of the SMACOF algorithm: error magnitude in the Swiss roll problem before and after applying a few SMACOF iterations.

4.5. *V-cycle*

The simplest multigrid algorithm is the V-cycle. The complete non-linear multigrid optimization algorithm can be defined recursively in the following manner:

$$\texttt{MG\_Vcycle}(\mathbf{X}_r, \mathbf{T}_r, \mathbf{\Delta}_r, \mathbf{W}_r, K, K')$$

- If $r = R$ (coarsest level) solve

$$\min_{\mathbf{X}_R} s_R(\mathbf{X}_R, \mathbf{T}_R)$$

  and return.
- Otherwise:
  - Relaxation: apply K iterations of an unconstrained optimization algorithm to $s_r(\mathbf{X}, \mathbf{T}_r)$ initialized with $\mathbf{X}_r$ and obtain $\mathbf{X}'_r$.
  - Compute

$$\mathbf{G}'_r = \nabla s_r(\mathbf{X}'_r)$$

$$\mathbf{X}'_{r+1} = \mathbf{P}_r^{r+1}\mathbf{X}'_r$$

$$\mathbf{G}'_{r+1} = \nabla s_{r+1}(\mathbf{X}'_{r+1})$$

$$\mathbf{T}_{r+1} = \mathbf{G}'_{r+1} - \mathbf{P}_r^{r+1}\mathbf{G}'_r$$

  - Apply the multigrid method on the coarser level,

$$\mathbf{X}''_{r+1} \leftarrow \texttt{MG\_Vcycle}(\mathbf{X}'_{r+1}, \mathbf{T}_{r+1}, \mathbf{\Delta}_{r+1}, \mathbf{W}_{r+1}, K, K')$$

  - Correction

$$\mathbf{E}_r = \mathbf{P}_{r+1}^{r}(\mathbf{X}''_{r+1} - \mathbf{X}'_{r+1})$$

$$\mathbf{X}''_r \leftarrow \mathbf{X}'_r + \alpha\mathbf{E}_r$$

  - Relaxation: apply K' iterations of an unconstrained optimization algorithm to $s_r(\mathbf{X}, \mathbf{T})$ initialized with $\mathbf{X}''_r$ and obtain $\mathbf{X}'''_r$.

The procedure $\texttt{MG\_Vcycle}(\mathbf{X}, \mathbf{T}, \mathbf{\Delta}, \mathbf{W}, K, K')$ is initialized with some $\mathbf{X}$ and $\mathbf{T} = 0$ on the finest grid. The procedure is executed for a few iterations. Following standard notation, we denote as $V(K, K')$-cycle a V-cycle with K and K' relaxation iterations, respectively. When multigrid works well, a small number of V-cycles is required to obtain an accurate solution.

Note that theoretically the step size α at the fine grid correction stage should be computed using line search [35]. To reduce the computational cost, we select $\alpha = 1$ and use line search only as a safeguard, when such a step increases the stress value. In practice, it appears that the line search is almost never used.

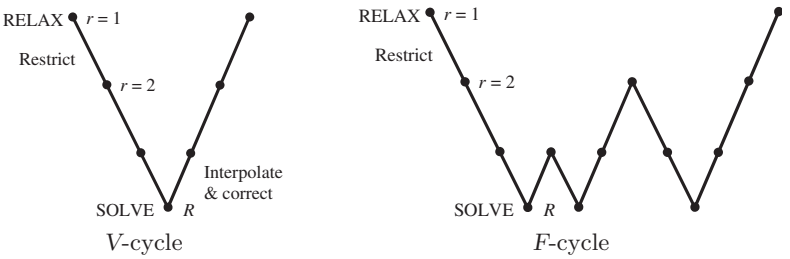

Figure 6. Schematic representation of different multigrid cycles.

4.6. F-cycle

Another popular MG cycle is the F-cycle [39]. In the recursive definition, the F-cycle procedure $\texttt{MG\_Fcycle}(\mathbf{X}_r, \mathbf{T}_r, \mathbf{\Delta}_r, \mathbf{W}_r, K, K')$ is very similar to V-cycle, except the recursive call, which has the form

$$\mathbf{X}''_{r+1} \leftarrow \texttt{MG\_Fcycle}(\mathbf{X}'_{r+1}, \mathbf{T}_{r+1}, \mathbf{\Delta}_{r+1}, \mathbf{W}_{r+1}, K, K')$$

$$\mathbf{X}''_{r+1} \leftarrow \texttt{MG\_Vcycle}(\mathbf{X}''_{r+1}, \mathbf{T}_{r+1}, \mathbf{\Delta}_{r+1}, \mathbf{W}_{r+1}, K, K')$$

The difference between V-cycle and F-cycle is shown in Figure 6.

5. GENERAL MDS PROBLEMS

Up to this point, we have discussed MDS in the context of the isometric embedding problem, where the matrix of distances $\mathbf{\Delta}$ arises from a smooth Riemannian manifold and the underlying geometry is explicitly available. This is, however only a particular case of the MDS problem. In general, an MDS problem can be applied to a case in which only the matrix $\mathbf{\Delta}$ is given and the geometry of the metric space of the original points is not available. Such a situation can occur in dimensionality reduction problems, where one has information about the distances (or more generally, *dissimilarities*) between the points, but has no access to the points themselves.

5.1. Generalization of the MG-MDS algorithm

The implication of the absence of explicit geometry in the general MDS problem is the inability to use the geometric multigrid framework described in Section 4. Simply put, there is no 'grid' in the problem. This particularly means that the construction of the interpolation and restriction operators is a much more complicated problem compared to the isometric embedding case. From a theoretical point of view, this problem is related to algebraic multigrid, e.g. the classical Ruge–Stüben scheme [40]. Defining the coarsening strategy in non-linear problems is still an open research question.

Here, we propose an *ad hoc* solution, which may be applicable in certain cases. We wish to construct a hierarchy of grids with decimation factor q (for simplicity, we assume that N is an integer power of q). The process starts by picking up a point at the finest resolution

level, whose index we denote by j_0^1, and finding its $q-1$ nearest neighbours (in sense of δ_{ij}), whose indices we denote by $j_1^1,\ldots,j_{q-1}^1$. The corresponding coarse level point is obtained by linear combination, e.g.

$$\mathbf{x}_1' = \frac{\sum_{k=1}^{q-1} \delta_{j_0^1 j_k^1}^1 \mathbf{x}_{j_k^1}}{\sum_{k=1}^{q-1} \delta_{j_0^1 j_k^1}^{-1}} \tag{25}$$

Other choices of the weighting coefficients are also possible. When $q=2$, we produce the coarsening by simple averaging of points j_0^1 and j_1^1, i.e. $\mathbf{x}_1' = \frac{1}{2}(\mathbf{x}_{j_0^1} + \mathbf{x}_{j_1^1})$. The process is repeated for all the rest of the points and at other resolution levels in a similar manner. Alternatively, the farthest point strategy can be employed for the construction of the hierarchy of grids.

6. RESULTS

In order to assess the performance of MG-MDS algorithm, three experiments were performed: two isometric embedding problems and a dimensionality reduction problem. In all the experiments, we used non-weighted MDS. The algorithms' performance was evaluated by the stress value versus the execution time (measured with the average error of about 10%) and the computational complexity in terms of multiplication operations (FLOPs). The square-root operation was estimated as $C_s = 25$ FLOPs. All tests were performed on a PC with a 2.0 GHz Mobile Intel Pentium 4 processor and 1 GB RAM. The algorithms were implemented in MATLAB under Windows XP.

The MG-MDS was terminated when the stress function at subsequent iterations decreased by less than 1% (this stopping criterion is common in the MDS literature, see, e.g. Reference [4]). For comparison, SMACOF and BFGS quasi-Newton[‖] algorithms were executed with the same initialization and were terminated when reaching the same stress achieved by the MG-MDS.

6.1. Unrolling the Swiss roll

In the first experiment, we performed isometric embedding of a surface called the *Swiss roll* into $\mathbb{R}^3$. The Swiss roll is known as a complicated object due to its highly non-linear and non-Euclidean structure [41]. Yet, this surface can be thought of as a 'rolled' planar patch, and is thus isometric to a planar surface. Consequently, MDS should 'unroll' it in $\mathbb{R}^3$.

In our experiment, the Swiss roll surface was given parametrically by the following formula:

$$\begin{aligned} x &= \theta \\ y &= 0.51\left(\frac{1}{2.75\pi} + 0.75\phi\right)\cos(2.5\pi\phi) \\ z &= 0.51\left(\frac{1}{2.75\pi} + 0.75\phi\right)\sin(2.5\pi\phi) \end{aligned} \tag{26}$$

where $(\theta,\phi) \in [0,1]\times[0,1]$ (see Figure 7, first column).

[‖] We used a MATLAB implementation of the BFGS algorithm from Kelley [26]. The code is available online: http://www4.ncsu.edu/~ctk/matlab_darts.html

Figure 7. Unrolling the Swiss roll: a few iterations of the MG-MDS. The resulting points in $\mathbb{R}^3$ are visualized as a surface using Phong shading. Shown here is a problem of size 2145.

Table II. Experiment Ia: Comparison of MG-MDS, SMACOF and BFGS algorithms on the Swiss roll isometric embedding problems of different sizes, initialization with the original points in $\mathbb{R}^3$. Overall execution time shown in seconds ($\pm 10\%$), overall complexity shown in MFLOPs.

	SMACOF			BFGS			MG-MDS $V(3,3)$-cycle, $R=3$		
N	Iter.	Time	Complexity	Iter.	Time	Complexity	Iter.	Time	Complexity
289	293	30.83	760.88	94	37.22	943.63	7	5.42	125.69
561	350	130.14	3.42×10^3	115	201.41	5.09×10^3	6	15.84	403.74
1089	340	452.58	1.25×10^4	118	804.37	2.04×10^4	6	56.20	1.51×10^3
2145	341	2.13×10^3	4.88×10^4	134	3.76×10^3	9.03×10^4	6	211.76	5.86×10^3

We compared our MG-MDS to conventional SMACOF and BFGS algorithms. In the first part of the experiment (Experiment Ia), we studied how the algorithms perform on problems of different size, namely: $N = 289$, 561, 1089 and 2145 (corresponding to grids of sizes 17×17, 33×17, 33×33 and 65×33, respectively). Approximated geodesic distances in these problems were computed on a finer 65×65 grid using the parametric fast marching [38]. The solution restriction and interpolation operators $\mathbf{P}_r^{r+1}$ and $\mathbf{P}_{r+1}^r$ were based on cubic and zero-order interpolation, respectively (generated using MATLAB function `griddata`). The data restriction operators $\tilde{\mathbf{P}}_r^{r+1}$ were based on zero-order interpolation.

The algorithms were initialized using the original points' co-ordinates in $\mathbb{R}^3$. In Experiment Ia, a $V(3,3)$ cycle with $R=3$ was used in MG-MDS. In the second part of the experiment (Experiment Ib), we compared different MG cycles on the problem of size 1089. All the algorithms were terminated when reaching the same stress value.

The experimental results are summarized in Tables II and III, which show a comparison between SMACOF, BFGS and the MG-MDS algorithms in terms of CPU time and computational complexity. The convergence ratio (i.e. the gradient norm ratio, $\|\nabla s(\mathbf{X}^{(k)})\|/\|\nabla s(\mathbf{X}^{(k-1)})\|^{-1}$, computed at the last iterations) of the MG-MDS algorithm in Experiment Ia ranged from 0.81

Table III. Experiment Ib: Comparison of different algorithms on the Swiss roll isometric embedding problem of size 1089, initialization with the original points in $\mathbb{R}^3$. Overall execution time shown in seconds ($\pm 10\%$), overall complexity shown in MFLOPs.

	Algorithm	Iter.	Time	Complexity
	SMACOF	340	452.58	1.25×10^4
	BFGS	118	804.37	2.04×10^4
MG	$V(3,3)$ $R=2$	21	191.40	5.18×10^3
	$V(2,2)$ $R=3$	11	73.75	1.99×10^3
	$V(3,3)$ $R=3$	6	56.20	1.51×10^3

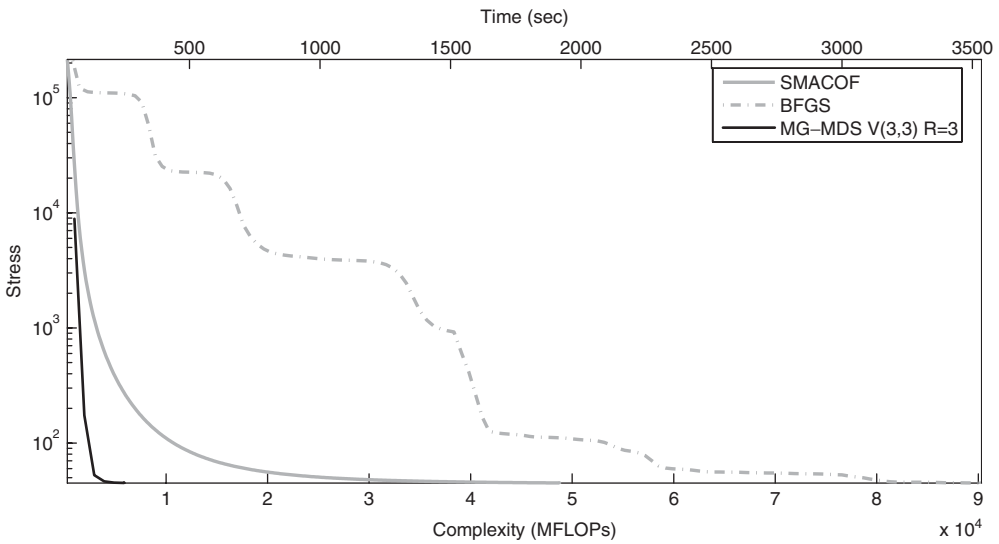

Figure 8. Experiment Ia: Convergence of SMACOF, BFGS and the MG-MDS $V(3,3)$ cycle in the Swiss roll problem of size 2145, initialization with the original points in $\mathbb{R}^3$. Shown here is the stress starting from the first iteration, as a function of computational complexity (bottom scale). The corresponding approximate CPU time is given on the upper scale ($1\,\mathrm{s} \sim 25.55 \pm 2.26$ MFLOPs).

($N = 289$ points) to 0.67 ($N = 2145$ points). Figure 7 shows a few iterations of the MG-MDS algorithm, which unroll the Swiss roll very quickly into a surface which looks approximately like a planar patch (due to numerical inaccuracies in geodesic distance computation, the Swiss roll is not exactly isometric to a planar patch). Figure 8 shows the convergence of SMACOF, BFGS and MG-MDS $V(3,3)$ cycle on the problem of size 2145. Figure 9 shows the convergence of SMACOF, BFGS and different MG-MDS cycles on the problem of size 1089.

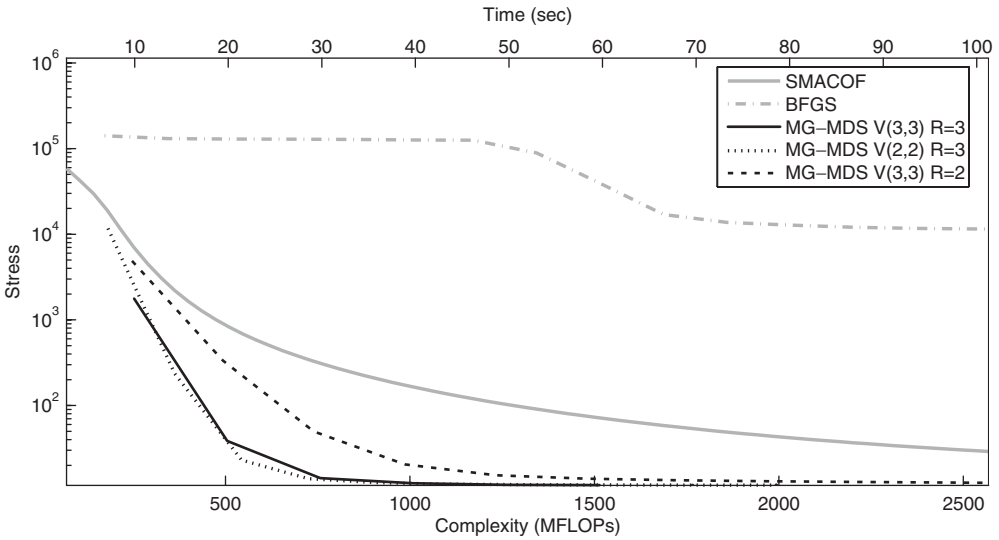

Figure 9. Experiment Ib: Convergence of different algorithms for the Swiss roll problem of size 1089, initialization with the original points in $\mathbb{R}^3$. Shown here (partially) is the stress starting from the first iteration, as a function of computational complexity (bottom scale). The corresponding approximate CPU time is given on the upper scale ($1\,\mathrm{s} \sim 25.4 \pm 0.3$ MFLOPs).

6.2. *Embedding a facial surface*

In the second experiment, we computed the expression-invariant representation of the face of one of the authors by isometrically embedding it into $\mathbb{R}^3$. The surface containing 5263 points was given parametrically; the parameterization domain had a non-convex shape. A hierarchy of grids with three resolution levels (containing 1977, 492 and 128 points, respectively) was constructed by the furthest point sampling algorithm. This is a problem of slightly smaller scale than those encountered in the 3D face-recognition applications (typically, $N \approx 2500$). Parametric fast marching [38] was used to compute the approximate geodesic distances on the surface in full resolution (5263 points).

The solution restriction and interpolation operators $\mathbf{P}_r^{r+1}$ and $\mathbf{P}_{r+1}^r$ were based on cubic and zero-order (nearest-neighbour) interpolation, respectively. The data restriction operators $\tilde{\mathbf{P}}_r^{r+1}$ were based on zero-order interpolation. The algorithms were initialized using the original points' co-ordinates in $\mathbb{R}^3$. We compared SMACOF, BFGS and different MG-MDS cycles. All the algorithms were terminated when reaching the same stress value.

The experimental results are summarized in Table IV that shows a comparison between the SMACOF, BFGS and the MG-MDS algorithms in terms of CPU time and computational complexity. Figure 10 depicts the results of a few iterations of the MG-MDS algorithm that efficiently 'flatten' the facial surface into the canonical form. Figure 11 shows the convergence of SMACOF, BFGS and different MG-MDS cycles. For additional results on facial surfaces embedding using the MG-MDS algorithm, see Reference [12].

Table IV. Experiment II: Comparison of different algorithms on the facial surface isometric embedding problem, initialization with the original points in $\mathbb{R}^3$. Overall execution time shown in seconds ($\pm 10\%$), overall complexity shown in MFLOPs.

	Algorithm	Iter.	Time	Complexity
	SMACOF	85	471.85	1.28×10^4
	BFGS	51	1.15×10^3	2.92×10^4
MG	$V(2,2)\ R=3$	4	86.52	2.36×10^3
	$V(3,3)\ R=3$	4	119.07	3.29×10^3
	$F(2,2)\ R=3$	3	66.28	1.81×10^3

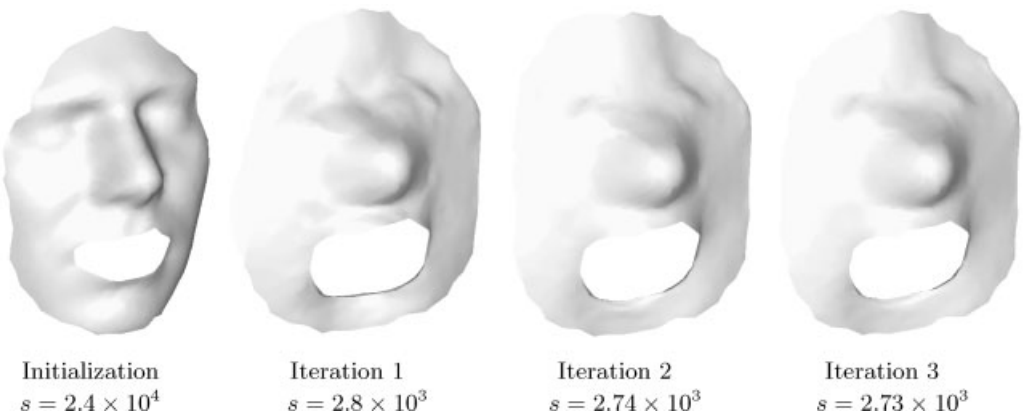

Figure 10. Embedding of a facial surface: as few as three MG-MDS iterations are sufficient in order to obtain a good expression-invariant representation. The resulting points in $\mathbb{R}^3$ are visualized as a surface using Phong shading.

6.3. *Dimensionality reduction*

In the third experiment, MDS was used to perform dimensionality reduction of a high-dimensional data set in a Euclidean space. The target embedding space was $\mathbb{R}^2$. Two sets of points were generated as random vectors (statistically independent in each co-ordinate) in $\mathbb{R}^{500}$ according to the following formula:

$$x_i = \text{sign}(\mathcal{N}(\pm 0.75, 1)); \quad i = 1, \ldots, 1024 \tag{27}$$

where $\mathcal{N}(\mu, \sigma)$ denotes Gaussian distribution with mean μ and variance σ^2, and

$$\text{sign}(x) \equiv \begin{cases} +1 & x > 0 \\ -1 & x \leqslant 0 \end{cases} \tag{28}$$

Data of such kind typically appear in biochemical applications [42]. Note that no geometric structure is given explicitly.

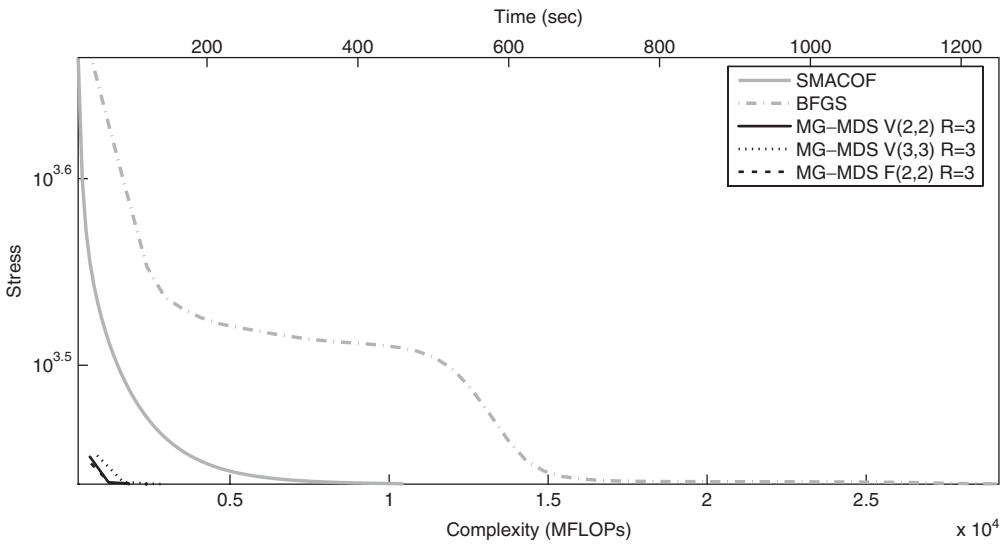

Figure 11. Experiment II: Convergence of different algorithms in the facial surface embedding problem, initialization with the original points in $\mathbb{R}^3$. Shown here is the stress starting from the first iteration, as a function of computational complexity (bottom scale). The corresponding approximate CPU time is given on the upper scale ($1\,\mathrm{s} \sim 23.33 \pm 1.7$ MFLOPs).

Table V. Experiment III: Comparison of MG-MDS and standard SMACOF algorithm on dimensionality reduction problems of different sizes, random initialization. Results averaged over 50 runs. Execution time shown in seconds, complexity shown in MFLOPs.

	SMACOF			MG-MDS (V-cycle, 3 levels)		
N	Iter.	Time	Complexity	Iter.	Time	Complexity
256	39.0 ± 9.5	3.5 ± 0.8	78.4 ± 18.7	5 ± 1.7	2.4 ± 0.8	49.2 ± 16.5
512	38.1 ± 7.8	12.5 ± 2.6	306.5 ± 61.0	4.8 ± 1.3	8.4 ± 2.3	189.5 ± 52.7
1024	40.7 ± 7.1	51.5 ± 9.1	$1.3 \times 10^3 \pm 223.2$	4.6 ± 0.9	31.3 ± 6.1	732.4 ± 142.8
2048	43.8 ± 6.4	223.5 ± 30.9	$5.6 \times 10^3 \pm 803.4$	4.4 ± 0.8	116.8 ± 22.1	$2.8 \times 10^3 \pm 526.4$

The difference between the two sets appears as one of them being slightly biased towards -1 and the other towards $+1$. If one observes each of the co-ordinates, the data will appear random. Therefore, this is a typical example of a dimensionality reduction problem where linear techniques like principal component analysis (PCA) would not produce plausible results, whereas MDS would.

In our experiment, we compared the conventional SMACOF algorithm and our MG-MDS ($V(2,2)$-cycle with 3 resolution levels). The restriction and interpolation operators $\mathbf{P}_r^{r+1}$, $\mathbf{P}_{r+1}^r$, $\tilde{\mathbf{P}}_r^{r+1}$ and $\tilde{\mathbf{P}}_{r+1}^r$ were computed using the *ad hoc* algorithm described in Section 5.1. Decimation factor $q = 2$ was used. Identical random initializations were used for MG-MDS and SMACOF. The results were averaged over 50 runs.

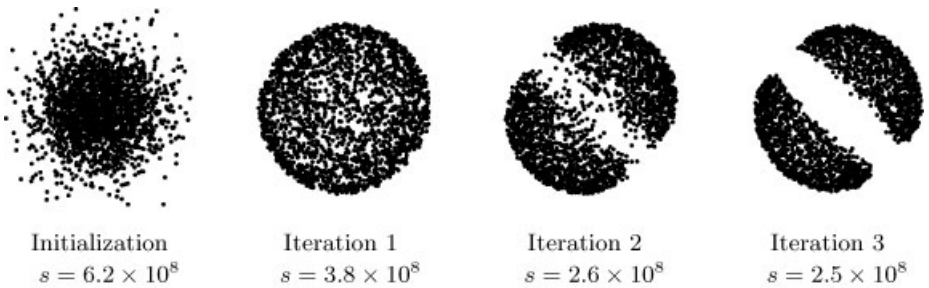

Figure 12. Dimensionality reduction: as few as three MG-MDS iterations are sufficient in order to obtain distinguishable clusters.

The experimental results are summarized in Table V that shows a comparison between the SMACOF and the MG-MDS algorithms in terms of average CPU time and computational complexity on problems of different sizes. Figure 12 shows the result of a few iterations of the MG-MDS algorithm. The separation into two distinguishable clusters becomes obvious after as few as two–three iterations.

6.4. Discussion

Several conclusions can be drawn from the above results. Firstly, MG-MDS demonstrates significant performance improvement in isometric embedding problems (over 8 times compared to SMACOF and over 15 times compared to BFGS in the Swiss roll experiment). Secondly, the improvements contributed by the multigrid algorithm, when compared to SMACOF or BFGS, become more pronounced as the size of the problem increases. That is, for large-scale problems the improvement is dramatic. Thirdly, the number of iterations of the MG-MDS in both isometric embedding experiments and also in the dimensionality reduction experiment seems to be independent of N, while the number of SMACOF and BFGS iterations tends to grow with N. Such a behaviour is typical for multigrid algorithms. Fourthly, the F-cycle generally appears to perform better than the corresponding V-cycle. For additional comparison results, see Reference [12].

We believe that the boundary effects observed in Figure 5 may result in slowing down the MG-MDS convergence. A simple remedy commonly used in multigrid literature would be performing extra relaxation on the boundary. It can either be done by applying additional iterations near the boundary points, or devising a smarter scheme of weighting the gradient-descent step to do stronger relaxation on the boundary.

The general MDS results presented here (the dimensionality reduction problem) are preliminary, as the coarsening strategy used was very naïve. Yet, a performance improvement of about 2 times was obtained over the SMACOF algorithm.

7. CONCLUSIONS

We proposed a multigrid framework for efficient solution of multidimensional scaling problems. The advantage of our MG-MDS algorithm in MDS problems with explicitly given

geometry (isometric embedding problems) is up to an order of magnitude in performance compared to standard algorithms for problems of moderate size. The gain is expected to increase appreciably with problem size. In general, MDS problems, where the geometry of the underlying metric space is not given explicitly, we showed that performance improvement can be obtained using our multigrid framework using even a very naïve coarsening approach. The problem of a good coarsening strategy for this case is still an open research question.

The range of applications in which our approach can be used is very wide, and includes, to mention a few, problems in data visualization, machine learning, computational chemistry, etc.

ACKNOWLEDGEMENT

We are grateful to Asi Elad for the permission to use Figure 2.

REFERENCES

1. Groenen P, Franses PH. Visualizing time-varying correlations across stock markets. *Journal of Empirical Finance* 2003; **7**:155–172.
2. Agrafiotis DK, Rassokhin DN, Lobanov VS. Multidimensional scaling and visualization of large molecular similarity tables. *Journal of Computational Chemistry* 2001; **22**(5):488–500.
3. Lähdesmäki H, Yli-Harja O, Shmulevich I, Zhang W. Distinguishing key biological pathways by knowledge-based multidimensional scaling analysis: application to discriminate between primary breast cancers and their lymph node metastases. *Proceedings of Bioinformatics*, 2003.
4. Borg I, Groenen P. *Modern Multidimensional Scaling—Theory and Applications*. Springer: New York, 1997.
5. Linial N, London E, Rabinovich Y. The geometry of graphs and some its algorithmic applications. *Combinatorica* 1995; **15**:333–344.
6. Schwartz EL, Shaw A, Wolfson E. A numerical solution to the generalized Mapmaker's problem: flattening nonconvex polyhedral surfaces. *IEEE Transactions on PAMI* 1989; **11**:1005–1008.
7. Zigelman G, Kimmel R, Kiryati N. Texture mapping using surface flattening via multi-dimensional scaling. *IEEE Transactions on Visualization and Computer Graphics* 2002; **9**(2):198–207.
8. Grossman R, Kiryati N, Kimmel R. Computational surface flattening: a voxel-based approach. *IEEE Transactions on PAMI* 2002; **24**(4):433–441.
9. Elad A, Kimmel R. On bending invariant signatures for surfaces. *IEEE Transactions on PAMI* 2003; **25**(10):1285–1295.
10. Bronstein AM, Bronstein MM, Kimmel R. Expression-invariant 3D face recognition. *Proceedings of Audio and Video-based Biometric Person Authentication*. Lecture Notes in Computer Science, vol. 2688. Springer: New York, 2003; 62–69.
11. Bronstein AM, Bronstein MM, Kimmel R. Three-dimensional face recognition. *International Journal of Computer Vision* 2005; **64**(1):5–30.
12. Bronstein MM, Bronstein AM, Kimmel R, Yavneh R. A multigrid approach for multidimensional scaling. *Proceedings of the Copper Mountain Conference on Multigrid Methods*, 2005.
13. Elad A, Kimmel R. Spherical flattening of the cortex surface. *Geometric Methods in Bio-medical Image Processing*, vol. 2191. Springer: New York, 2002; 77–89.
14. Bronstein AM, Bronstein MM, Kimmel R. On isometric embedding of facial surfaces into $\mathbf{S}^3$. *Proceedings of the International Conference on Scale Space and PDE Methods in Computer Vision*. Lecture Notes in Computer Science, vol. 3459. Springer: New York, 2005; 622–631.
15. Walter J, Ritter H. On interactive visualization of high-dimensional data using the hyperbolic plane. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002; 123–131.
16. Bronstein AM, Bronstein MM, Kimmel R. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Science* (PNAS), 2006, in press.
17. Bronstein AM, Bronstein MM, Kimmel R. Efficient computation of the Gromov-Hausdorff distance for smooth surfaces. *SIAM Journal on Scientific Computing*, under review.
18. Bronstein MM. Three-dimensional face recognition. *Master's Thesis*, Department of Computer Science, Technion, Israel Institute of Technology, 2004.
19. Kearsley A, Tapia R, Trosset M. The solution of the metric STRESS and SSTRESS problems in multidimensional scaling using Newton's method. *Computational Statistics* 1998; **13**(3):369–396.

20. Young G, Householder AS. Discussion of a set of point in terms of their mutual distances. *Psychometrika* 1938; **3**:19–22.
21. Torgerson WS. Multidimensional scaling I—theory and methods. *Psychometrika* 1952; **17**:401–419.
22. Guttman L. A general nonmetric technique for finding the smallest coordinate space for a configuration of points. *Psychometrika* 1968; 469–506.
23. de Leeuw J. Applications of convex analysis to multidimensional scaling. *Recent Developments in Statistics*. North-Holland: Amsterdam, 1977; 133–145.
24. de Leeuw J, Stoop I. Upper bounds on Kruskal's stress. *Psychometrika* 1984; **49**:391–402.
25. Nash S. A survey of truncated-Newton methods. *Journal of Computational and Applied Mathematics* 2000; **124**:45–59.
26. Kelley CT. *Iterative Methods for Optimization*. Frontiers in Applied Mathematics. SIAM: Philadelphia, 1999.
27. Nocedal J, Wright S. *Numerical Optimization*. Springer: New York, 1999.
28. Knoll DA, Rider WJ. A multigrid preconditioned Newton–Krylov method. *SIAM Journal on Scientific Computing* 1999; **21**(2):691–710.
29. Jones JE, Woodward CS. Newton–Krylov-multigrid solvers for large-scale, highly heterogeneous, variably saturated flow problems. *Advances in Water Resources* 2001; **24**:763–774.
30. Trottenberg U, Oosterlee C, Schuller A. *Multigrid*. Academic Press: New York, 2001.
31. Fedorenko RP. On the speed of convergence of an iterative process. *USSR Computational Mathematics and Mathematical Physics* 1964; **4**:559–564.
32. Bakhvalov NS. On the convergence of a relaxation method under natural constraints on an elliptic operator. *USSR Computational Mathematics and Mathematical Physics* 1966; **6**:861–883.
33. Brandt A. Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems. *Proceedings of the 3rd International Conference on Numerical Methods in Fluid Mechanics*, 1973; 82–89.
34. Brandt A. Multi-level adaptive solutions to boundary-value problem. *Mathematics of Computation* 1977; **37**:333–390.
35. Nash S. A multigrid approach to discretized optimization problems. *Journal of Optimization Methods and Software* 2000; **14**:99–116.
36. Eldar Y, Lindenbaum M, Porat M, Zeevi YY. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing* 1997; **6**(9):1305–1315.
37. Kimmel R, Sethian JA. Computing geodesic on manifolds. *Proceedings of the National Academy of Science* (PNAS), vol. 95, 1998; 8431–8435.
38. Spira A, Kimmel R. An efficient solution to the eikonal equation on parametric manifolds. *Interfaces and Free Boundaries* 2004; **6**(3):315–327.
39. Brandt A. Multigrid techniques: 1984 guide with applications to fluid dynamics. GMD-Studie, 1985.
40. Ruge J, Stuben K. Algebraic multigrid. *Multigrid Methods*, Frontiers in Applied Mathematics, vol. 3. SIAM: Philadelphia, 1987; 73–130.
41. Roweis ST, Saul LK. Nonlinear dimensionality reduction by locally linear embedding. *Science* 2000; **290**(5500):2323–2326.
42. KDD Cup 2001. DuPont Pharmaceuticals Research Laboratories thrombin dataset. Online. http://www.cs.wisc.edu/~dpage/kddcup2001/

