7

Multidimensional Scaling

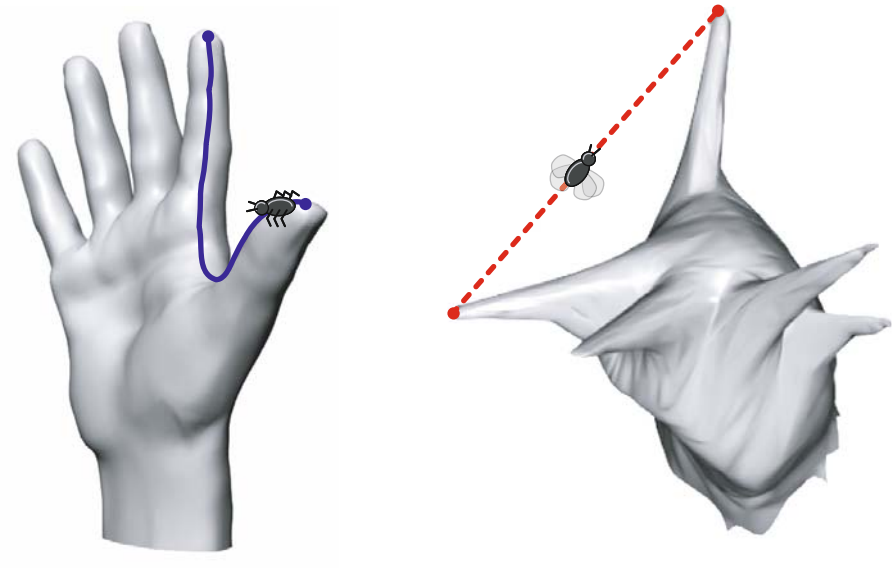
> The world is complex, dynamic, multidimensional; the paper is static, flat. How are we to represent the rich visual world of experience and measurement on mere flatland?
>
> E. R. Tufte, *Envisioning Information*

Thus far, in our fairy-tale example, the Prince could find Cinderella by comparing the extrinsic geometries of the glass slipper and the feet of all the ladies in his kingdom using rigid similarity methods. Now, assume that instead of dropping a glass slipper, Cinderella has lost a silk glove while escaping from the ball. Trying to naïvely approach the glove fitting problem with rigid similarity tools, the Prince soon finds that, because the glove is a non-rigid object, comparison based on the extrinsic geometry does not work anymore.

Let us leave the desperate Prince for a while and recall what we said in Chapter 2: in order to compare non-rigid shapes, we should look at their intrinsic geometries, which are invariant to isometric deformations. In other words, considering shapes as metric spaces, we need to compare the spaces X and Y with the geodesic metrics d_X and d_Y, respectively. Such a comparison appears to be by far a more complicated task than is the comparison of extrinsic geometry, for the following reason. The relative simplicity of the extrinsic similarity problem we had when discussing rigid shapes in Chapter 6 was due to the fact that the shapes were considered subsets of a common metric space ($\mathbb{R}^3$ with the Euclidean metric). Hence, we could measure their similarity using the Hausdorff distance, which led to the ICP algorithms. In the case of intrinsic similarity, the situation is more difficult: we now have two *different* metric spaces (X, d_X) and (Y, d_Y), which cannot be compared using the Hausdorff distance.

In this and the next few following chapters, we try to build a bridge between the two approaches. We will see how to represent the intrinsic geometries of the shapes in a common metric space where they can be compared using rigid similarity algorithms. This will lead us to a class of computationally tractable methods for measuring intrinsic similarity of non-rigid shapes.

A. Bronstein et al., *Numerical Geometry of Non-Rigid Shapes*,
Monographs in Computer Science, DOI 10.1007/978-0-387-73301-2_7



Surface Canonical form

Figure 7.1. Illustration of the isometric embedding problem. Left: original shape; right: canonical form.

7.1 Isometric embedding problem

Let us come back for a moment to the example we have already used in Chapter 2 to illustrate the concept of intrinsic geometry. Assume that our shape is inhabited by an insect, which always chooses the shortest path to crawl between any two points. Now, imagine that there is another shape embedded into $\mathbb{R}^m$, whose points correspond with those of the original shape, while the Euclidean distances between the points are equal to the original geodesic ones. On this new shape, there lives another winged insect, which flies along straight lines between the points.

The lengths of the paths are the isometry-invariant description of our object. Because the distances traveled by both insects are equal, the descriptions produced by them are the same. However, for our application, the second insect's point of view is preferable, as his world is Euclidean. The advantage stems from the smaller number of degrees of freedom that influence our description: whereas the first insect would not feel any isometric deformation of the shape, the only way we can fool the second one is by applying rigid transformations, which are limited to rotations, translations, or reflections.

Formalizing the above intuition, given a shape (X, d_X), we would like to find a map $f : (X, d_X) \to (\mathbb{R}^m, d_{\mathbb{R}^m})$, such that

$$d_X(x, x') = d_{\mathbb{R}^m}(f(x), f(x')),$$

for all $x, x' \in X$. Such an f is an isometric embedding, and the space $\mathbb{R}^m$ is referred to as the *embedding space* in this context. The image $f(X)$, which we call the *canonical form* of X, can be used as an extrinsic representation of the intrinsic geometry of X (see example in Figure 7.1). Note that we regard $f(X)$ as a metric space with the restricted Euclidean metric $d_{\mathbb{R}^m}|_{f(X)}$. Up to isometries in $\mathbb{R}^m$, it defines an equivalence class of all the shapes that are indistinguishable from the point of view of intrinsic geometry. In simple words, two isometric shapes have identical canonical forms, possibly differing by an isometry of $\mathbb{R}^m$.

In a sense, this embedding allows us to undo the non-uniqueness of the way the metric structure of X is realized in $\mathbb{R}^m$ (all the possible bendings), thus reducing its vast number of degrees of freedom. Consequently, considering the canonical forms instead of the shapes themselves, we translate the non-rigid shape similarity problem into a much simpler problem of rigid similarity, with which we already know how to deal. This simple idea, proposed by Asi Elad and R. K. [147, 149], allows us to define the similarity between two shapes as an extrinsic distance between their canonical forms, measured by means of ICP or the moments method as shown in Algorithm 7.1. We call this distance the *canonical form distance* and denote it by d_{CF}.

Though originally formulated with the particular choice of $\mathbb{R}^3$ (i.e., $m = 3$) as the embedding space, the canonical forms approach can be generalized to any embedding space, as we will see in Chapter 9. Here, we stick to the Euclidean embedding, but assume m to be arbitrary. Thus far, the canonical forms method seems an ideal recipe for our problem of non-rigid shape comparison. However, there is still a question whether a shape X is isometrically embeddable into $\mathbb{R}^m$.

Unfortunately, the answer is usually negative. As the simplest case, consider the problem of embedding a sphere into $\mathbb{R}^2$. This problem arose in cartography centuries ago. One of the fundamental problems in map-making is creating a planar map of the Earth, which reproduces, in the best way, the distances between geographic objects. That is, equipped with a simple ruler, we can measure distances on the map, which represent geodesic distances on the Earth (Figure 7.2). Every cartographer knows that it is impossible to create a map of the Earth that preserves all the geodesic distances.[1] This, as a matter of fact, is a consequence of the *theorema egregium*: because the Gaussian curvature of the sphere is positive, whereas the plane has zero curvature, these two surfaces cannot be isometric.

input : shapes (X, d_X) and (Y, d_Y).
output: canonical forms distance $d_{\mathrm{CF}}(X, Y)$.

1 Find the isometric embedding f and g of X and Y into $\mathbb{R}^m$.
2 Compute $d_{\mathrm{CF}}(X, Y)$ as $d_{\mathrm{MOM}}(f(X), g(Y))$ or $d_{\mathrm{ICP}}(f(X), g(Y))$.

Algorithm 7.1. Idealized canonical forms distance computation.

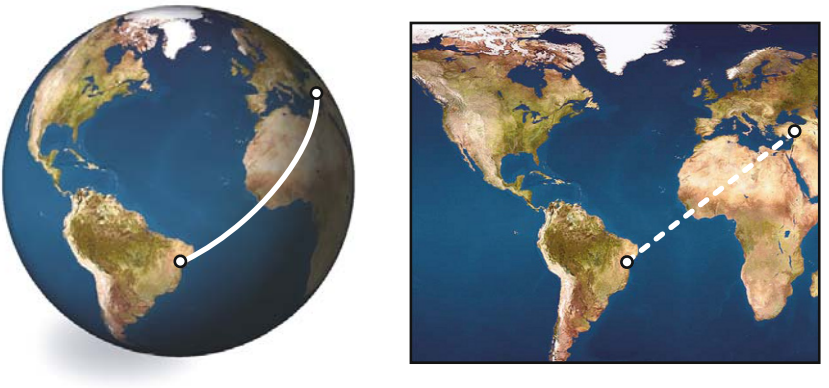


Figure 7.2. The problem of isometric embedding arising in cartography: the spherical surface of the Earth (shown is the upper hemisphere, left) is to be mapped into the plane so that it preserves the geodesic distances (right). A consequence of *theorema egregium* is that such a map does not exist, and a distortion of the distance is inevitable.

Yet, maybe by increasing the embedding space dimension, i.e., trying to embed the sphere into $\mathbb{R}^3, \mathbb{R}^4, \mathbb{R}^5$, and so on, we could succeed in finding an isometric embedding? Even this appears to be impossible. The following example, shown by Nathan Linial [253], demonstrates that even a very simple discrete metric space consisting of only four points cannot be isometrically embedded into a Euclidean space of any finite dimension.

Example 7.1 (Linial's example). Consider four points $x_1, \ldots, x_4$, sampled on the sphere of radius $R = \frac{2}{\pi}$ as shown in Figure 7.3 (one point at the north pole and three points along the equator). The distances between the points are given by the following matrix

$$D_X = \begin{pmatrix} 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 1 \\ 2 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

We denote the embedded points by $z_1, \ldots, z_4$ and assume that the embedding is distortionless, that is, $(D_X)_{ij} = d_{\mathbb{R}^m}(z_i, z_j)$ for $i, j = 1, \ldots, 4$. Let us consider first the triangle with vertices z_1, z_2, z_3, with edges of lengths $d_{\mathbb{R}^m}(z_1, z_2) = d_{\mathbb{R}^m}(z_2, z_3) = 1$ and $d_{\mathbb{R}^m}(z_1, z_3) = 2$. Because $d_{\mathbb{R}^m}(z_1, z_3) = d_{\mathbb{R}^m}(z_1, z_2) + d_{\mathbb{R}^m}(z_2, z_3)$, the triangle is flat, i.e., the points z_1, z_2, z_3 are collinear. Applying the same reasoning, we conclude that the points z_1, z_4, z_3 are collinear, which implies that $z_2 = z_4$ and consequently, $d_{\mathbb{R}^m}(z_2, z_4) = 0$, contradicting the assumption that $d_{\mathbb{R}^m}(z_2, z_4) = d_X(x_2, x_4) = 1$. Because we

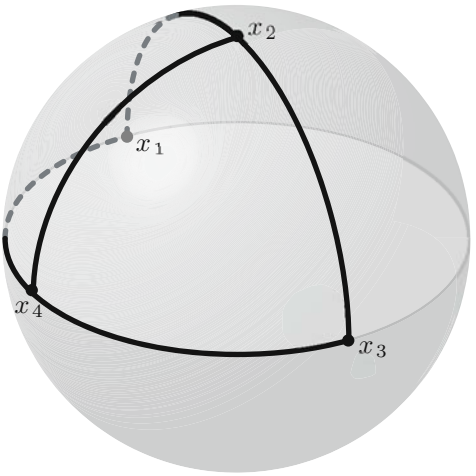


Figure 7.3. Linial's example of a metric space obtained by sampling the sphere at four points, which cannot be embedded into a Euclidean space of any finite dimension.

have not assumed any particular m, we conclude that the given structure cannot be isometrically embedded into a Euclidean space of any finite dimension. Moreover, if this is the case for such a simple object as a sphere, our conclusion is that a general shape cannot be isometrically embedded into $\mathbb{R}^m$.

It is important to emphasize that this result by no means contradicts the Nash embedding theorem. Nash guarantees that any Riemannian structure can be realized as a length metric induced by a Euclidean metric, whereas we are trying to realize it using the restricted Euclidean metric.

Although the embedding error makes it impossible to find a truly isometric embedding, we could try constructing an approximate representation of the shape X, looking for a *minimum-distortion embedding*, i.e., such f that distorts d_X the least, in the sense of some criterion. In Chapter 2, we defined the distortion, which reads in our problem as

$$\operatorname{dis} f = \sup_{x,x' \in X} |d_X(x,x') - d_{\mathbb{R}^m}(f(x), f(x'))|.$$

Adopting this criterion, we can measure how the distances on the original shape differ from those in the embedding space in the sense of the L_∞-norm. In practice, it is useful to replace the L_∞ criterion by an L_p analog,

$$\sigma_p = \int_{X \times X} |d_{\mathbb{R}^m}(f(x), f(x')) - d_X(x,x')|^p \, da \times da, \tag{7.1}$$

where da denotes the area element on X. The L_∞ criterion can be obtained as the limit of $(\sigma_p)^{1/p}$ when $p \to \infty$.

In the discrete setting, when the shape X is sampled at N points $\{x_1, \ldots, x_N\}$, the L_∞ criterion becomes

$$\sigma_\infty = \max_{i,j=1,\ldots,N} |d_{\mathbb{R}^m}(f(x_i), f(x_j)) - d_X(x_i, x_j)|,$$

and the discrete version of the L_p criterion can be expressed as

$$\sigma_p = \sum_{i>j} a_i a_j |d_{\mathbb{R}^m}(f(x_i), f(x_j)) - d_X(x_i, x_j)|^p, \tag{7.2}$$

where a_i and a_j are discrete area elements corresponding with the points x_i, x_j. If the shape is sampled uniformly, we can simplify σ_p by setting $a_i = 1/N$.

The canonical form obtained by means of a minimum-distortion embedding is only an approximate representation of the shape's intrinsic geometry. Nevertheless, we can still measure the similarity of shapes and the distance between their canonical forms, of course, having in mind that the distortion introduced by the embedding would influence the accuracy of such a similarity.

7.2 Multidimensional scaling

An important question is how to find the minimum-distortion embedding in practice. Assume that the shape X is uniformly sampled at points $\{x_1, \ldots, x_N\}$ and σ_2 is used as the distortion criterion. We are looking for the minimum-distortion embedding into $\mathbb{R}^m$,

$$f = \operatorname*{argmin}_{f:X\to\mathbb{R}^m} \sum_{i>j} |d_{\mathbb{R}^m}(f(x_i), f(x_j)) - d_X(x_i, x_j)|^2.$$

Denoting by $z_i = f(x_i)$ the m-dimensional Euclidean coordinates of the image of the shape sample x_i under f and arranging them into an $N \times m$ matrix $Z = (z_i^j)$, we can rewrite our distortion criterion as

$$\sigma_2(Z; D_X) = \sum_{i>j} |d_{ij}(Z) - d_X(x_i, x_j)|^2.$$

Here $D_X = (d_X(x_i, x_j))$ is an $N \times N$ matrix of geodesic distances and $d_{ij}(Z)$ is a shorthand notation for the Euclidean distance between the i-th and the j-th points on the canonical form, $d_{\mathbb{R}^m}(z_i, z_j) = \|z_i - z_j\|_2$. In this formulation, we find the coordinates of the discrete canonical form directly as the solution of a nonlinear least-squares problem,

$$Z^* = \operatorname*{argmin}_{Z\in\mathbb{R}^{N\times m}} \sigma_2(Z). \tag{7.3}$$

Problem (7.3) is a non-convex optimization problem [380] in Nm variables, in which the objective function $\sigma_2 : \mathbb{R}^{N\times m} \to \mathbb{R}$ is defined over the space of

matrices $\mathbb{R}^{N\times m}$. The data in the problem are the geodesic distances, represented as an $N \times N$ matrix $D_X = (d_X(x_i, x_j))$. The solution is not unique, because, as already mentioned, applying any Euclidean isometry to Z^*, we do not change the value of σ_2.

Historically, problem (7.3) is called *multidimensional scaling* (MDS), a term coined by Torgerson [379] in 1952. Such problems have been researched in depth in the psychology and statistics communities in the relation to multidimensional data analysis and visualization. The MDS problem in a formulation similar to the one we give here is attributed to Shepard [355] and Kruskal [232, 233]; the latter also proposed a numerical algorithm for its solution. Efforts to consolidate these methods were invested during the 1960s and the 1970s. To the field of computer vision and pattern recognition, MDS methods arrived relatively late. One of the first applications to analysis of surfaces was shown by Schwartz *et al.* in 1989, in relation to the problem of brain surface analysis [345]. The paper showed how to use MDS in order to create a planar map of the convoluted brain cortex surface, in a manner similar to that cartographers employ to map the Earth. Later, Schweitzer applied MDS for classification of databases of images [347]. These studies have been an inspiration for the embedding-based approaches to non-rigid shape comparison we discuss here.

7.3 SMACOF algorithm

In the MDS literature, the function $\sigma_2(Z)$ we use as the distortion criterion is commonly referred to as the (*Kruskal*) *stress* [44]. We call $\sigma_2(Z)$ the L_2-*stress* in order to distinguish it from other distortion criteria. We can write the L_2-stress in a more convenient matrix form,

$$\begin{aligned}\sigma_2(Z; D_X) = &\operatorname{trace}(Z^{\mathrm{T}} V Z) - 2\operatorname{trace}(Z^{\mathrm{T}} B(Z; D_X) Z) \\ &+ \sum_{i>j} d_X^2(x_i, x_j),\end{aligned} \tag{7.4}$$

where V is a constant $N \times N$ matrix with elements

$$v_{ij} = \begin{cases} -1 & i \neq j \\ N-1 & i = j, \end{cases}$$

and $B(Z; D_X)$ is an $N \times N$ matrix depending on Z and D_X with elements,

$$b_{ij}(Z; D_X) = \begin{cases} -d_X(x_i, x_j) d_{ij}^{-1}(Z) & i \neq j \text{ and } d_{ij}(Z) \neq 0 \\ 0 & i \neq j \text{ and } d_{ij}(Z) = 0 \\ -\sum_{k\neq i} b_{ik} & i = j. \end{cases}$$

(In the following, we will sometimes omit the dependence on D_X for brevity). A particular property of matrices B and V is that they are zero-mean, i.e.,

the sum of the rows and the columns is zero. We leave the derivation of the matrix form of the stress to the reader as Problem 7.2.

Solution of the MDS problem requires the minimization of $\sigma_2(Z; D_X)$ with respect to Z. The most straightforward way to do it is by using a first-order gradient descent algorithm. Recall that gradient descent consists of making a step in the negative gradient direction, which requires computing the gradient of the stress at each iteration. Using matrix function differentiation similar to that shown in Example 5.1 in Chapter 5, the gradient of $\sigma_2(Z)$ can be expressed as

$$\nabla_Z \sigma_2(Z) = 2VZ - 2B(Z; D_X)Z, \tag{7.5}$$

see Problem 7.3. The gradient descent iteration has the following form,

$$\begin{aligned} Z^{(k+1)} &= Z^{(k)} - \alpha^{(k)} \nabla_Z \sigma_2(Z^{(k)}) \\ &= Z^{(k)} - 2\alpha^{(k)} \left(VZ^{(k)} - B(Z^{(k)}; D_X)Z^{(k)} \right). \end{aligned} \tag{7.6}$$

The step size $\alpha^{(k)}$ can be selected either as a constant, or computed at each iteration using line search. The use of line search requires evaluating the stress and its gradient a number of times. In our problem, the complexity of computing $\sigma_2(Z)$ and $\nabla_Z \sigma_2(Z)$ is $\mathcal{O}(N^2)$, which implies that for a large value of N line search is usually disadvantageous.

Jan de Leeuw [125, 129, 126] noticed that the term $\mathrm{trace}(Z^{\mathrm{T}} B(Z) Z)$ of the stress can be bounded below by $\mathrm{trace}(Z^{\mathrm{T}} B(Q) Q)$ for all $Q \in \mathbb{R}^{N \times m}$, which leads to the inequality

$$\begin{aligned} h(Z, Q) &= \mathrm{trace}(Z^{\mathrm{T}} V Z) - 2\mathrm{trace}(Z^{\mathrm{T}} B(Q) Q) + \sum_{i>j} d_X^2(x_i, x_j) \\ &\geq \mathrm{trace}(Z^{\mathrm{T}} V Z) - 2\mathrm{trace}(Z^{\mathrm{T}} B(Z) Z) + \sum_{i>j} d_X^2(x_i, x_j) \end{aligned} \tag{7.7}$$

(the reader is invited to prove the inequality in Problem 7.5). The function $h(Z, Q)$ is convex and quadratic with respect to Z and touches $\sigma_2(Z)$ at the point $Q = Z$, i.e., $h(Z, Z) = \sigma_2(Z)$.

By virtue of inequality (7.7), $h(Z, Q)$ serves as a majorizing function for the stress. We can resort to the iterative majorization algorithm for the solution of the MDS problem. At the $(k+1)$st iteration of the majorization algorithm, the solution $Z^{(k+1)}$ is found as the minimizer of $h(Z, Z^{(k)})$ with respect to Z. Because the majorizing function is quadratic, the minimizer can be easily expressed analytically by imposing

$$\nabla_Z h(Z, Z^{(k)}) = 2VZ - 2B(Z^{(k)}; D_X)Z^{(k)} = 0.$$

This leads to the following multiplicative update formula

$$Z^{(k+1)} = V^{\dagger} B(Z^{(k)}; D_X) Z^{(k)}, \tag{7.8}$$

input : $N \times N$ matrix of geodesic distances D_X.
output : canonical form Z^*.
initialization: some initial $Z^{(0)}$ and $k = 0$.

1 **repeat**
2 Multiplicative update: $Z^{(k+1)} = \frac{1}{N} B(Z^{(k)}; D_X) Z^{(k)}$.
3 $k \longleftarrow k + 1$.
4 **until** *convergence*
5 $Z^* = Z^{(k)}$.

Algorithm 7.2. SMACOF algorithm.

where $V^\dagger = (V^{\mathrm{T}}V)^{-1}V^{\mathrm{T}}$ denotes the pseudoinverse of V (as the matrix V has rank $N-1$, it is not invertible). Further noticing that the pseudoinverse can be written as $V^\dagger = \frac{1}{N}(I - \frac{1}{N}1_{N\times N})$, where $1_{N\times N}$ denotes an $N \times N$ matrix of ones, and keeping in mind that $B(Z^{(k)}; D_X)$ is zero mean, such that $1_{N\times N}B(Z^{(k)}; D_X) = 0$, we can rewrite the update formula (7.8) as

$$Z^{(k+1)} = \frac{1}{N} B(Z^{(k)}; D_X) Z^{(k)}.$$

This multiplicative update was given the name SMACOF, standing for *scaling by minimizing a convex function*.[2] The entire algorithm can be summarized as shown in Algorithm 7.2. SMACOF has become one of the most successful and widely used MDS methods, mainly due to its simplicity and public availability of efficiently implemented code [189].

Though not straightforward to observe, a simple manipulation of the SMACOF multiplicative update (7.8) leads to

$$\begin{aligned} Z^{(k+1)} &= V^\dagger B(Z^{(k)}) Z^{(k)} \\ &= Z^{(k)} - Z^{(k)} + V^\dagger B(Z^{(k)}; D_X) Z^{(k)} \\ &= Z^{(k)} - \frac{1}{2} V^\dagger \left(2VZ^{(k)} - 2B(Z^{(k)}; D_X) Z^{(k)} \right) \\ &= Z^{(k)} - \frac{1}{2} V^\dagger \nabla_Z \sigma_2(Z^{(k)}). \end{aligned}$$

Using the zero-mean property once more, we can make the final retouch to the update formula,

$$Z^{(k+1)} = Z^{(k)} - \frac{1}{2N} \nabla_Z \sigma_2(Z^{(k)}; D_X).$$

This is a somewhat surprising conclusion: the SMACOF algorithm appears to be nothing but a gradient descent with a constant step size, $\alpha^{(k)} = \frac{1}{2N}$. In a sense, the whole idea of majorization has been reduced in our problem to the specific choice of the step size in the gradient descent algorithm. On the other hand, the majorization algorithm guarantees that the sequence

$Z^{(1)}, Z^{(2)}, \ldots$ produces decreasing values of the stress, and that the iterative process converges to a (possibly local) minimum of $\sigma_2(Z)$. Such a behavior is a particularly interesting property of the MDS problem, rarely observed in other optimization problems – in general, it is uncommon that constant-step gradient descent produces a monotonic decrease of the objective function.

As a historical remark, we shall note that the SMACOF iteration was derived as early as in 1968 by Louis Guttman (1916–1987), who observed [196] that the first-order optimality condition $\nabla_Z \sigma_2(Z^*; D_X) = 0$ can be written as

$$Z^* = V^\dagger B(Z^*; D_X) Z^*.$$

This condition can be thought of as the fixed point of the multiplicative update formula (7.8). This observation is a third way toward deriving the SMACOF algorithm. Recognizing Guttman's prior work, de Leeuw and Heiser [128] dubbed their multiplicative update as the *Guttman transform*.

7.4⋆ Second-order methods

As an alternative to the first-order SMACOF algorithm, we can use second-order methods for the solution of the MDS problem. Such methods have been studied by Kearsley *et al.* [220]. The basic Newton iteration in our problem takes the form

$$Z^{(k+1)} = Z^{(k)} + \alpha^{(k)} \mathcal{H}^{-1}(Z^{(k)}; D_X) \nabla_Z \sigma_2(Z^{(k)}; D_X).$$

In this notation, the Hessian $\mathcal{H}$ is a tensor, which can be thought of as a four-dimensional matrix with elements

$$h_{ij}^{kl}(Z; D_X) = \begin{cases} \tilde{h}_{ij}^{kl}(Z; D_X) & \text{if } k \neq l, \\ \tilde{h}_{ij}^{kl}(Z; D_X) + 2(v_{ij} - b_{ij}(Z; D_X)) & \text{if } k = l, \end{cases}$$

for $1 \leq k, l \leq m$ and $1 \leq i, j \leq N$, where $\tilde{h}_{ij}^{kl}(Z; D_X)$ are given by

$$\tilde{h}_{ij}^{kl}(Z; D_X) = -\begin{cases} 2(z_i^k - z_j^k)(z_i^l - z_j^l) d_X(x_i, x_j) d_{ij}^{-3}(Z) & \text{if } i \neq j,\ d_{ij}(Z) \neq 0, \\ 0 & \text{if } i \neq j,\ d_{ij}(Z) = 0, \\ \sum_{n \neq i} \tilde{h}_{in}^{kl} & \text{if } i = j. \end{cases}$$

$\mathcal{H}$ is symmetric with respect to the indices i, j and k, l. The derivation of the Hessian is tedious and is left to the reader (Problem 7.6).

Because working with such a structure can sometimes be cumbersome, it is common to convert the matrix variable into a vector one, by parsing the $N \times m$ matrix Z into an $Nm \times 1$ column vector in column-stack order. We denote this transformation by $z = \mathrm{vec}(Z)$. The Hessian in this representation is an $Nm \times Nm$ symmetric block matrix, consisting of m^2 blocks $H^{kl}(z; D_X) = (h_{ij}^{kl}(Z; D_X))$ of size $N \times N$,

$$H(z;D_X) = \begin{pmatrix} H^{11}(Z;D_X) & H^{12}(Z;D_X) & \dots & H^{1m}(Z;D_X) \\ \vdots & & & \vdots \\ H^{m1}(Z;D_X) & H^{m2}(Z;D_X) & \dots & H^{mm}(Z;D_X) \end{pmatrix}. \quad (7.9)$$

The Newton iteration, in turn, can be rewritten as

$$z^{(k+1)} = z^{(k)} + \alpha^{(k)} H^{-1}(z^{(k)};D_X)\nabla_z \sigma_2(z^{(k)};D_X),$$

where $\nabla_z \sigma_2(z;D_X) = \mathrm{vec}(\nabla_Z \sigma_2(Z;D_X))$. Note that the Hessian is not invertible: because each of the blocks is zero-mean, it has zero eigenvalues. The Hessian may also contain negative eigenvalues because the problem is non-convex. In order to invert such a matrix, we can modify its eigenvalues to make all the negative and zero ones to be positive, e.g., by adding a small value to its diagonal, $H + \epsilon I_{Nm\times Nm}$.[3] The step size $\alpha^{(k)}$ is selected using line search. Because of high computational complexity of the stress and its gradient, inexact line search (for example, the Armijo rule) is preferred over exact one. The main complexity of the Newton iteration is due to the Hessian construction ($\mathcal{O}(N^2m^2)$ operations) and inversion. Because the Hessian is full, solution of the Newton system requires $\mathcal{O}(N^3m^3)$ operations.

Example 7.2 (SMACOF vs. Newton). We exemplify the difference between the SMACOF and the Newton MDS algorithms on the problem of embedding the *Swiss roll surface* (shown in Figure 7.4) into $\mathbb{R}^3$. The Swiss roll can be thought of as a rolled piece of paper, therefore, it is isometric to the plane. In our example, the surface is given in a triangular mesh representation and the geodesic distances computed using fast marching, therefore, the isometry is only approximate. Both algorithms are implemented in MATLAB. For the Newton algorithm, Armijo rule with $\alpha, \beta = 0.3$ is used. Hessian inversion is performed using eigendecomposition of the Hessian matrix with a modification of the negative eigenvalues forcing them to be positive. Both algorithms are initialized with the same random configuration of points. The stopping condition uses the normalized gradient norm, $\|\nabla_Z \sigma_2(Z^{(k)})/N^2\| \le 10^{-3}$ (the scaling is necessary for the stopping condition to be independent of N).

Figure 7.5 (first row) depicts the normalized gradient norm on the logarithmic scale versus time for the SMACOF and Newton algorithms in embedding of the Swiss roll sampled at $N = 50$ points. The Newton algorithm shows a classic case of quadratic convergence, whereas SMACOF has linear convergence, typical for a first-order algorithm. Overall, the convergence of the Newton algorithm is much faster: 24 iterations (1.45 sec) compared with 1094 iterations (3.6 sec) for SMACOF.

Yet, the situation changes dramatically when the number of variables becomes larger ($N = 200$, see Figure 7.5 second row). Because the complexity of the Newton system solution is proportional to N^3, the iteration complexity increases by about 64 times. As the result, the overall convergence time increases significantly to 119.4 sec (at the same time, the number of iterations of the Newton algorithm is now 43, i.e., does not grow so significantly). Unlikely,

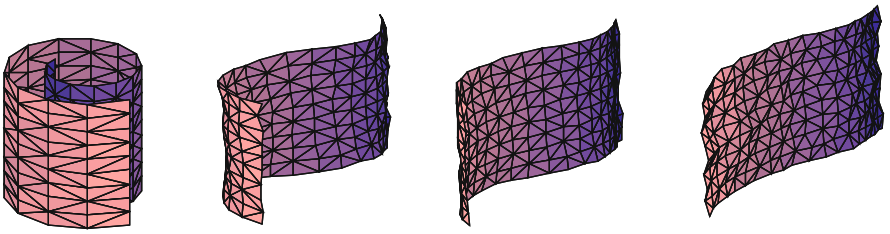


Figure 7.4. The Swiss roll surface sampled at 200 points and represented as a triangular mesh (left) and the result of its embedding into $\mathbb{R}^3$ using a few iterations of the SMACOF algorithm, starting from a random initialization.

the complexity of the SMACOF iteration is $\mathcal{O}(N^2)$, a 16 times increase. The SMACOF algorithm converges in 29.6 sec (1216 iterations).

Our first conclusion from Example 7.2 is that in large-scale MDS problems, SMACOF is advantageous over the Newton algorithm. In particular, in our problems, where typically $N \sim 1000$, performing the Newton iteration straightforwardly becomes almost impractical. Secondly, the MDS problem usually does not require reaching a very high accuracy, therefore, the condition on the gradient norm is rather inappropriate. In the MDS literature, a condition on the relative change of the stress is often used instead. If we stop the optimization when $\sigma_2(Z^{(k+1)})/\sigma_2(Z^{(k)}) \leq 0.001$, we will see that SMACOF terminates much faster than does the Newton algorithm. This phenomenon is also observed in both of our experiments. In the case of $N = 50$ points, SMACOF reaches this stopping criterion in 0.4 sec compared with 1.4 sec required for the Newton algorithm. In the second case ($N = 200$), the corresponding numbers are 4.5 sec for SMACOF and 119.4 sec for Newton.

7.5 Variations on the stress theme

Thus far, we have considered embedding obtained by minimizing the L_2-stress criterion. Let us say a few words about other possibilities. Besides $p = 2$ yielding the L_2-stress we have already seen, the cases $p = 1$ and $p = \infty$ are the most interesting among the possible choices of p in our definition of (7.2). The L_1-stress,

$$\sigma_1(Z; D_X) = \sum_{i>j} |d_{ij}(Z) - d_X(x_i, x_j)|,$$

is similar to the L_2-stress, with the exception that the sum of squared differences is replaced with the sum of absolute differences. Yet, optimization of

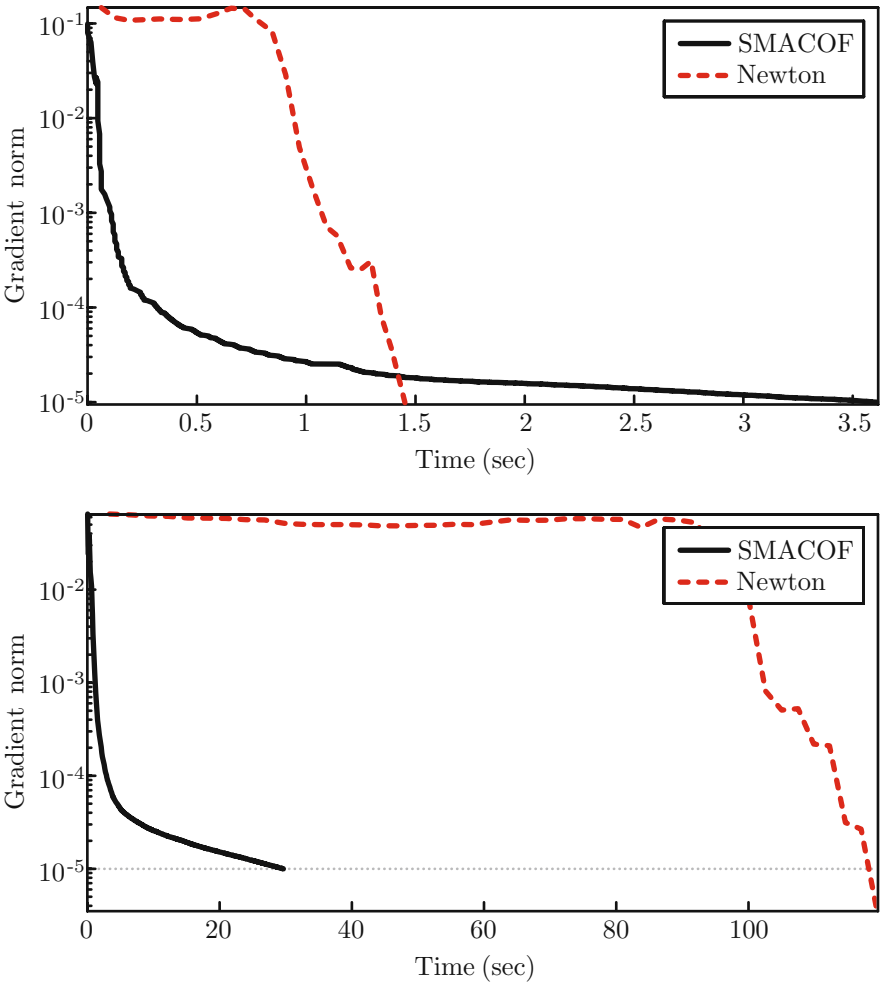


Figure 7.5. Convergence plots of the SMACOF (solid) and Newton (dashed) MDS algorithms in the problem of Swiss roll embedding with $N = 50$ (top) and $N = 200$ (bottom) points. Shown is the normalized gradient norm $\|\nabla_Z \sigma_2(Z^{(k)})\|_2/N^2$ (bottom) as a function of CPU time. The horizontal line denotes the stopping criterion, $\|\nabla_Z \sigma_2(Z^{(k)})\|_2/N^2 \leq 10^{-5}$.

the L_1-stress is more difficult, as the absolute value function $|t|$ is not differentiable at $t = 0$. It is possible to *smooth* the absolute value by replacing it with a differentiable function, e.g., $\varphi(t) = \sqrt{t^2 + \epsilon}$, such that $\varphi(t) \approx |t|$ for a small positive ϵ. An alternative is resorting to more complicated optimization algorithms referred to as *subgradient methods*, based on a generalization of the notion of gradient for non-differentiable functions [315].

In the case of $p = \infty$, minimization of σ_∞ is a *min-max problem*,

$$Z^*_\infty = \operatorname*{argmin}_Z \max_{i,j=1,\dots,N} |d_{ij}(Z) - d_X(x_i, x_j)|. \tag{7.10}$$

A common trick in optimization is to reformulate this problem as a constrained one, introducing an *artificial variable* τ,

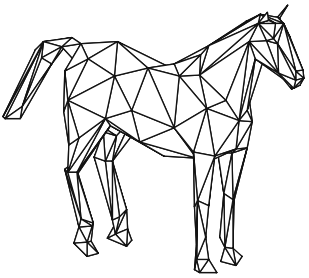
$$\begin{aligned} Z^*_\infty &= \operatorname*{argmin}_{Z,\tau} \tau \quad \text{s.t.} \quad |d_{ij}(Z) - d_X(x_i, x_j)| \le \tau \\ &= \operatorname*{argmin}_{Z,\tau} \tau \quad \text{s.t.} \quad \begin{cases} d_{ij}(Z) - d_X(x_i, x_j) - \tau \le 0, \\ -d_{ij}(Z) + d_X(x_i, x_j) - \tau \le 0, \end{cases} \end{aligned} \tag{7.11}$$

where $i > j$. It can be shown that the two problems (7.10) and (7.11) are equivalent. In the new problem, we have $Nm + 1$ variables, a linear objective function and $\frac{1}{2}N(N-1)$ constraints. Parsing the matrix variable into an $Nm \times 1$ vector, the gradients of the constraints can be written as an $(Nm + 1) \times \frac{1}{2}N(N-1)$ sparse matrix, in which each column is a gradient of $d_{ij}(Z)$ with respect to (τ, Z).

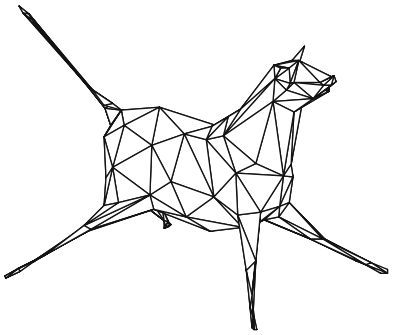
Example 7.3 (L_2- vs L_∞-stress). We exemplify the difference between distortion criteria by showing an embedding of the horse shape into $\mathbb{R}^3$. The horse is represented as a triangular mesh with $N = 153$ points (see Figure 7.6, top). The distances are measured using fast marching on the triangular mesh. Figure 7.6 (bottom left) shows the canonical form Z^*_2 obtained using the L_2-stress. The average distortion of the distances at the minimum is the average stress, $(2\sigma_2(Z^*_2)/N(N-1))^{1/2} = 0.037$. Figure 7.6 (bottom right) shows the canonical form Z^*_∞ obtained using the L_∞-stress. The value of the stress obtained in this case is $\sigma_\infty(Z^*_\infty) = 0.105$.

Carefully observing the canonical forms obtained in Example 7.3, we notice that the L_2 canonical form is much smoother than its L_∞ counterpart. Such behavior is typical and highlights the difference between the two distortion criteria. Note that in our constrained problem, the value of τ is essentially the L_∞-stress, as τ gives a tight upper bound on the distance distortion. In our example, only a small number of constraints are active in the L_∞ problem. This means that there are few "problematic" pairs of points, the distances between which determine the value of the L_∞-stress. Other distances result in a smaller stress, such that the corresponding constraints are inactive.

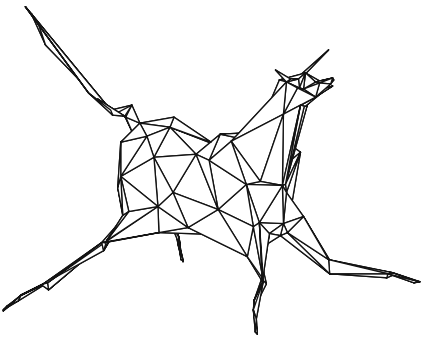
It may often happen that a few "bad" distances (arising, for example, from numerical inaccuracies) will result in a large L_∞ measure, whereas the other distances have significantly smaller distortion. Such a situation is improbable in the L_2 problem, as a single distance does not contribute much to the L_2-stress, and the effects of numerical inaccuracies are "distributed" among all the points. In other words, the L_∞ problem is more sensitive to *outliers*, which makes the L_∞-stress disadvantageous in practical applications. On the other hand, theoretical analysis of the L_∞ problem is simpler than that of the L_2



Horse



L_2 canonical form



L_∞ canonical form

Figure 7.6. Embedding of the horse shape into $\mathbb{R}^3$ using the L_2-stress and the L_∞-stress as the distortion criterion.

one. As we will see in the next chapters, the L_∞-stress allows the use of the powerful tools of metric geometry.

Besides the L_p-stress, there exist other distortion criteria, often encountered in the MDS literature. For example, the *relative stress*,

$$\sigma_{\mathrm{REL}}(Z; D_X) = \sum_{i>j} \frac{|d_{ij}(Z) - d_X(x_i, x_j)|^2}{d_{ij}^2(Z)},$$

can be used to measure the relative distortion of the distances. In a sense, it can be thought of as the discrete L_2 version of the dilation $\operatorname{dil} f$, in the same way we think of σ_2 as of an L_2 approximation of $\operatorname{dis} f$. If we wish to attribute more importance to some distances (e.g., if it is known that the accuracy of some distances may be lower), we can use the *weighted stress*,

$$\sigma_{\mathrm{W}}(Z; D_X, W) = \sum_{i>j} w_{ij} |d_{ij}(Z) - d_X(x_i, x_j)|^2,$$

where w_{ij} are some non-negative weights. For example, if the shape is sampled non-uniformly, we can define $w_{ij} = a_i a_j$, where a_i, a_j are discrete area ele-

ments. The weights w_{ij} can be represented as a symmetric $N \times N$ matrix W. The SMACOF algorithm can still be used in this case, with the only difference that in (7.8) we use the matrix V defined as

$$v_{ij} = \begin{cases} -w_{ij} & i \neq j \\ \sum_{k \neq i} w_{ik} & i = j, \end{cases}$$

(observe that our previous definition of V is a particular case corresponding with the choice $w_{ij} = 1$).

A notable use of the weighted stress is the so-called *iteratively reweighted least squares* technique (IRLS for short), often used in statistics to approximate the solution of problems with robust norms [199, 164, 212]. To demonstrate the use of IRLS for the solution of MDS problems, we consider the following general stress function,

$$\sigma_\rho(Z; D_X) = \sum_{i>j} \rho(d_{ij}(Z) - d_X(x_i, x_j)), \tag{7.12}$$

with $\rho(t)$ being some norm. For example, setting $\rho(t) = |t|^p$ gives the L_p norm; other notable choices are the German-McLure function

$$\rho_{\mathrm{GM}}(t) = \frac{t^2}{t^2 + \epsilon^2}, \tag{7.13}$$

and the quadratic-linear Huber function

$$\rho_{\mathrm{H}}(t) = \begin{cases} \frac{t^2}{2\epsilon} & : |t| \leq \epsilon \\ |t| - 0.5\epsilon & : |t| > \epsilon, \end{cases} \tag{7.14}$$

where ϵ is a positive constant, The necessary condition for Z to be a local minimizer of σ_ρ is

$$\nabla_Z \sigma_\rho(Z^*; D_X) = \sum_{i>j} \rho'(d_{ij}(Z^*) - d_X(x_i, x_j))\, \nabla_Z d_{ij}(Z^*) \;=\; 0.$$

Instead of minimizing σ_ρ, we can minimize the weighted L_2 stress (7.12), whose minimizer has to satisfy

$$\nabla_Z \sigma(Z^*; D_X, W) = \sum_{i>j} 2\, w_{ij} (d_{ij}(Z^*) - d_X(x_i, x_j))\, \nabla_Z d_{ij}(Z^*) \;=\; 0.$$

If we could selecting the weights in $\sigma(Z; D_X, W)$ according to

$$w_{ij} = \frac{\rho'(d_{ij}(Z^*) - d_X(x_i, x_j))}{2(d_{ij}(Z^*) - d_X(x_i, x_j))}, \tag{7.15}$$

the two minimizers would coincide and we could solve the weighted L_2 MDS problem instead of the general one. However, such a selection of the weights

input : $N \times N$ matrix of geodesic distances D_X, the derivative of ρ
output : canonical form Z^*.
initialization: some initial $Z^{(0)}$, $w_{ij}^{(0)} = 1$, and $k = 0$.

1 **repeat**
2 Find $Z^{(k+1)} = \arg\min \sigma(Z; D_X, W^{(k)})$ using $Z^{(k)}$ as the initialization.
3 Update weights according to

$$w_{ij}^{(k+1)} = \frac{\rho'(d_{ij}(Z^{(k+1)}) - d_X(x_i, x_j))}{2(d_{ij}(Z^{(k+1)}) - d_X(x_i, x_j))}$$

4 $k \longleftarrow k + 1$.
5 **until** *convergence*
6 $Z^* = Z^{(k)}$.

Algorithm 7.3. Iteratively reweighted least squares MDS.

requires the knowledge of the minimizer Z^*, which is, of course, unknown. A possible remedy is to start by solving the un-weighted L_2 problem (all $w_{ij} = 1$), use the solution to update the weights, and iterate the process until convergence. This iteratively reweighting procedure can be summarized as shown in Algorithm 7.3.

Step 2 can be performed using the SMACOF algorithm. The reweighting in Step 3 depends on the specific selection of the function ρ. For example, if $\rho(t) = |t|^p$ is used, the update formula becomes

$$w_{ij}^{(k+1)} = \frac{1}{2} p \, |d_{ij}(Z^{(k+1)}) - d_X(x_i, x_j)|^{p-2}.$$

7.6 Multiresolution methods

High computational complexity and the risk of local convergence are problems common to all the MDS algorithms we have discussed. Trying to reduce the complexity of MDS has recently become an important research trend in the MDS community, where the growth of the data sets dealt with (sometimes exceeding $N \sim 10^6$) has led to the development of approximate large-scale MDS methods. The idea of most of these methods is reducing the complexity by considering a part of the data. For example, Chalmers [95], Morrison *et al.* [283], and Williams and Munzner [398] showed an approximation to the MDS problem, in which a distinction is made between near and far neighbors for each point. In our formulation, this approach is equivalent to removing a major part of the distances from the stress computation, leaving only the distances to the near neighbors of each point and only a few to the far ones. Faloutsos and Lin [155], Wang *et al.* [391], and de Silva and Tennenbaum [132] proposed

performing MDS on a small subset of points (the so-called *landmarks*) and then finding the rest of the points by means of interpolation.[4]

The common denominator of approximate MDS approaches is posing a problem whose solution is close to the solution of the original MDS problem, but which is much simpler to solve. Such methods are applicable in problems of data visualization, where usually only qualitative results are required. In our problem of non-rigid shape comparison, the use of approximations of this kind may compromise the accuracy of shape representation [149]. Hence, in most cases, we have no choice but to solve the full MDS problem. At the same time, an approximate MDS solution can be used as a good initialization.

As an illustration, consider the landmarks method. Assume that we have $N = 1000$ points (a typical order of magnitude of the number of samples in our problems) and use 250 landmarks. We first solve the small MDS problem to embed the 250 landmarks, and then add the other 750 points by interpolation. Though the canonical form obtained this way may be insufficiently accurate for comparison of shapes, it is still close to the one obtained by solving the full MDS problem with 1000 points and can therefore be used for its initialization. This way, we spend more effort at the coarse level, where each iteration costs 16 times less, in order to save computations at the fine resolution level. Besides the complexity advantage, we also reduce the risk of local convergence: when solving the MDS problem at a coarse resolution, small local minima of the stress are usually avoided.

This idea, which we call *multiresolution optimization*, can be extended to more than two resolution levels. It is not limited to the specific choice of the distortion criterion but rather acts as an external framework, into which any iterative MDS algorithm can be incorporated. Formally, we assume to be given a hierarchy of *grids*, indexed by $l = 0, \ldots, L$, which successively approximate the full MDS problem. The Lth level is the coarsest one, while the zeroth level is the finest one, corresponding with the full problem. At each level, we work with a grid consisting of points with indices $\Omega_L \subset \Omega_{L-1} \subset \cdots \subset \Omega_0 = \{1, \ldots, N\}$. We denote the number of points at each level by $N_L < N_{L-1} < \cdots < N_0 = N$. At the lth level, the data is represented as an $N_l \times N_l$ matrix D_l, obtained by extracting the rows and columns of $D_0 = D_X$, corresponding with the indices Ω_l. The MDS problem at the lth-level is $Z_l^* = \operatorname{argmin}_{Z_l} \sigma(Z_l; D_l)$; we write $\sigma(Z; D)$ as a generic distortion criterion, emphasizing its dependence on the distance data D. The solution Z_l^* 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. The whole multiresolution MDS algorithm can be summarized as Algorithm 7.4.

Initialization of the multiresolution optimization algorithm actually conceals two problems: construction of the hierarchy of resolution levels and interpolation operators. The hierarchy of resolution levels can be constructed using the farthest point sampling strategy we have described in Chapter 3. This approach naturally allows creating a set of "nested" subsamplings, guaranteeing that every level is an r_l-covering of the shape. Typically, the number

input : $(L+1)$-level hierarchy of data $D_0, \ldots, D_L$, interpolation operators $P_1^0, \ldots, P_L^{L-1}$.
output : canonical form Z^*.
initialization: some initial $Z_L^{(0)}$ at the coarsest grid.

for $l = L, \ldots, 0$ **do**
    Solve the lth level MDS problem,

$$Z_l^* = \underset{Z_l \in \mathbb{R}^{N_l \times m}}{\operatorname{argmin}} \sigma(Z_l; D_l),$$

    using $Z_l^{(0)}$ as the initialization for the optimization algorithm.
    Interpolate the solution to the next level, $Z_{l-1}^{(0)} = P_l^{l-1} Z_l^*$.
end
Solve the finest level MDS problem,

$$Z^* = \underset{Z \in \mathbb{R}^{N \times m}}{\operatorname{argmin}} \sigma(Z; D_X),$$

using $Z_0^{(0)}$ as the initialization for the optimization algorithm.

Algorithm 7.4. Multiresolution MDS algorithm.

of points at each level is chosen such that $2 \leq N_{l-1}/N_l \leq 4$. The fact that FPS is a purely intrinsic geometric algorithm allows us to employ it in the most general MDS setting, where the matrix of geodesic distances D_X is the only information available, and no extrinsic geometry can be assumed. The cost of the hierarchy construction is of $\mathcal{O}(N^2)$ complexity, usually significantly smaller compared with the MDS algorithm itself.[5]

The interpolation operators are constructed based on the hierarchy of grids. In order to pass from the lth level to the $(l-1)$st level, we need to interpolate the points of the fine grid Ω_{l-1} from the points of the coarse grid Ω_l. For each fine grid point $z_{i \in \Omega_{l-1}}$, we define the neighborhood $\mathcal{N}_l(i) \subset \Omega_l$ of coarse grid points, from which the point z_i is interpolated. Points common to the fine and the coarse grid have a trivial neighborhood (equal to the point itself, $\mathcal{N}_l(i \in \Omega_l) = \{i\}$), and are transferred unchanged. The coordinates of the rest of the points, $i \in \Omega_{l-1} \setminus \Omega_l$, are interpolated according to

$$z_i = \sum_{j \in \mathcal{N}_l(i)} p_{ij} z_j.$$

The scalars p_{ij} are called *interpolation coefficients* or *weights* and can be identified with the elements of the interpolation matrix P_l^{l-1}. Because usually the number of neighbors used for interpolation is small ($|\mathcal{N}_l(i)| \ll N_l$), the matrix P_l^{l-1} is sparse. Consequently, the interpolation operation can be carried out efficiently, with $\mathcal{O}(N_{l-1})$ complexity.

The specific choice of $\mathcal{N}_l(i)$ and p_{ij} depends on the interpolation method. The neighborhood $\mathcal{N}_l(i)$ can be defined as the K nearest neighbors of x_i on the

coarse grid, or alternatively, as the metric ball of radius proportional to the sampling radius. If the shape is represented as a triangular mesh, the neighborhood can be inferred from the triangulation. In the case of parametric surfaces, the interpolation coefficients can be computed in the parameterization domain. There are numerous ways to define the interpolation coefficients, on which we will not spend our attention. The simplest choice is $p_{ij} = |\mathcal{N}_l(i)|^{-1}$, giving z_i as the center of mass of the points $\{z_j\}_{j\in\mathcal{N}_l(i)}$.

7.7⋆ Multigrid MDS

Assume that we have a good initialization $Z_0^{(0)} \approx \operatorname{argmin}_{Z_0} \sigma(Z_0; D_0)$ for the MDS problem on the finest grid and wish to improve it. $Z_0^{(0)}$ can be improved by performing a few optimization steps minimizing the stress $\sigma(Z_0; D_0)$ on the fine grid, but such a minimization can be computationally expensive, especially if N is large. Instead, we would like to solve a smaller and computationally cheaper problem on the coarse grid and use the obtained solution to improve $Z_0^{(0)}$ (here, we assume two levels of grids, $L = 1$). For this purpose, we first need to transfer the solution from the fine grid to the coarse grid using a *decimation operator* P_0^1, which, much like the interpolation operator P_1^0, can be thought of as an $N_1 \times N_0$ sparse matrix. In a sense, decimation is dual to interpolation, and in many cases, the decimation operator is obtained as the transpose of the interpolation operator, $P_0^1 = (P_1^0)^{\mathrm{T}}$. For the sake of simplicity, we assume this relation in the following discussion.

Having transferred $Z_0^{(0)}$ to the coarse grid and obtained $Z_1^{(0)} = P_0^1 Z_0^{(0)}$, we solve the coarse grid problem, which yields the solution Z_1^*. Using Z_1^*, we can correct the fine grid solution by transferring the difference in the coarse grid solution $Z_1^* - Z_1^{(0)}$ to the fine grid,

$$\begin{aligned} Z_0^{(1)} &= Z_0^{(0)} + P_1^0(Z_1^* - Z_1^{(0)}) \\ &= Z_0^{(0)} + P_1^0(Z_1^* - P_0^1 Z_0^{(0)}). \end{aligned}$$

Obviously, we would like the new fine grid solution $Z_0^{(1)}$ to be at least as good as $Z_0^{(0)}$, i.e., $\sigma(Z_0^{(1)}) \le \sigma(Z_0^{(0)})$. Yet, this would not necessarily hold, as the stress functions on the fine and coarse grids may be inconsistent: their minima do not necessarily coincide. More precisely, given the fine grid minimizer Z_0^*, for which $\nabla_{Z_0}\sigma(Z_0^*) = 0$, transferring it to the coarse grid, we will generally have $\nabla_{Z_1}\sigma(P_0^1 Z_0^*) = T_1 \neq 0$. The term T_1 is the *residual*, reflecting the inconsistency of the coarse and fine grid problems. To cancel the residual, we have to make an adjustment of the coarse grid problem by introducing a *correction term* to the stress

$$\sigma(Z_1; D_1) - \langle T_1, Z_1\rangle = \sigma(Z_1; D_1) - \operatorname{trace}(Z_1^{\mathrm{T}} T_1).$$

After this correction, the coarse grid problem becomes

$$\min_{Z_1} \sigma(Z_1; D_1) - \langle T_1, Z_1 \rangle.$$

In order to guarantee consistency, T_1 must satisfy

$$\nabla_{Z_1} \sigma(Z_1; D_1) - T_1 = \nabla_{Z_1} \sigma \left(Z_0^{(0)} + P_1^0 (Z_1 - P_0^1 Z_0^{(0)}); D_0 \right),$$

for $Z_1 = P_0^1 Z_0^{(0)}$. Using the chain rule, we obtain

$$\begin{aligned} \nabla_{Z_1} \sigma(Z_1; D_1) - T_1 &= \nabla_{Z_1} \sigma \left(Z_0^{(0)} + P_1^0 (Z_1 - P_0^1 Z_0^{(0)}); D_0 \right) \Big|_{Z_1 = P_0^1 Z_0^{(0)}} \\ &= (P_1^0)^{\mathrm{T}} \nabla_{Z_0} \sigma(Z_0^{(0)}; D_0) \\ &= P_0^1 \nabla_{Z_0} \sigma(Z_0^{(0)}; D_0), \end{aligned} \tag{7.16}$$

which leads to the expression for the correction term,

$$T_1 = \nabla_{Z_1} \sigma(P_0^1 Z_0^{(0)}; D_1) - P_0^1 \nabla_{Z_0} \sigma(Z_0^{(0)}; D_0).$$

In particular, when we take $Z_0^{(0)}$ to be the fine grid minimizer Z_0^*, it is easy to verify that $\nabla_{Z_1} \left(\sigma(P_0^1 Z_0^*; D_1) - \mathrm{trace}((P_0^1 Z_0^*)^{\mathrm{T}} T_1) \right) = 0$, i.e., that $P_0^1 Z_0^*$ is the coarse grid minimizer.

Yet, resolving one problem, we have created a new one: the coarse grid problem $Z^* = \mathrm{argmin}_Z \sigma(Z) - \mathrm{trace}(Z^{\mathrm{T}} T)$, after the introduction of the correction term becomes unbounded. Indeed, we can add an arbitrarily large scalar to the coordinates Z without changing $\sigma(Z)$, yet, if $T \neq 0$, the term $-\mathrm{trace}(Z^{\mathrm{T}} T)$ can decrease arbitrarily. In order to overcome this difficulty, in [79, 80] Irad Yavneh and the authors introduced the *modified stress* by adding a quadratic penalty to $\sigma(Z)$,

$$\begin{aligned} \hat{\sigma}(Z) &= \sigma(Z) + \lambda \sum_{k=1}^{m} \left(\sum_{i=1}^{N} z_i^k \right)^2 \\ &= \sigma(Z) + \lambda \, \mathrm{trace}(Z^{\mathrm{T}} 1_{N \times N} Z). \end{aligned} \tag{7.17}$$

Here, λ is some positive constant. Recall that the solution to the MDS problem is defined up to an isometry in the embedding space. Our modification merely resolves the translation ambiguity by restricting the center of mass of the resulting canonical form to the origin. The penalty would not change the solution, and we can therefore use $\hat{\sigma}(Z)$ instead of $\sigma(Z)$ in our problem. Because the penalty term is quadratic, it grows faster than does the linear term $\mathrm{trace}(Z^{\mathrm{T}} T)$. Consequently, the function $\hat{\sigma}(Z) - \mathrm{trace}(Z^{\mathrm{T}} T)$ is bounded. The gradient of the modified stress has practically the same computational complexity as that of the standard stress and is given by

$$\nabla_Z \hat{\sigma}(Z) = \nabla_Z \sigma(Z) + 2\lambda 1_{N \times N} Z.$$

Now we can combine the pieces of the puzzle together into a scheme, usually referred as the *two-grid algorithm* (Algorithm 7.5). The scheme consists of

input : two-level hierarchy of data D_0, D_1, interpolation operator P_1^0, decimation operator P_0^1.
output : canonical form Z^*.
initialization: some initial $Z_0^{(0)}$ at the fine grid.

1 **repeat**
2 Compute an improved fine grid solution $Z_0^{(1)}$ by performing N_R optimization iterations on $\sigma(Z_0)$, initialized with $Z_0^{(0)}$.
3 Decimate the fine grid solution: $Z_1^{(1)} = P_0^1 Z_0^{(1)}$.
4 Compute the correction: $T_1 = \nabla_{Z_1}\hat{\sigma}(Z_1^{(1)}; D_1) - P_0^1 \nabla_{Z_0}\hat{\sigma}(Z_0^{(1)}; D_0)$.
5 Solve the coarse grid problem,

$$Z_1^* = \underset{Z_1 \in \mathbb{R}^{N_1 \times m}}{\operatorname{argmin}} \hat{\sigma}(Z_1; D_1) - \operatorname{trace}(Z_1^{\mathrm{T}} T_1),$$

using $Z_1^{(1)}$ as the initialization.
6 Correct the fine grid solution, $Z_0^{(2)} = Z_0^{(1)} + \alpha_1 P_1^0 (Z_1^* - P_0^1 Z_0^{(1)})$.
7 Set $Z_0^{(0)} \longleftarrow Z_0^{(2)}$.
8 **until** *convergence*
9 $Z^* = Z_0^{(0)}$.

Algorithm 7.5. Two-grid MDS algorithm.

repeatedly improving the fine grid solution by performing a few optimization steps on the fine grid, decimating the result to the coarse grid and solving the coarse grid problem with an appropriate correction term, and then using the coarse grid solution to improve the fine grid result.

At the correction stage (Step 6 of the two-grid algorithm), a step size α_1 must be determined using line search (for example, Armijo rule) in order to guarantee a descent direction [291]. Step 5 is generic, and the choice of the optimization algorithm usually depends on the stress used. For example, if the L_2-stress is used as the distortion criterion, we can employ SMACOF-type iteration,

$$\begin{aligned} Z^{(k+1)} &= Z^{(k)} - \frac{1}{2N} \nabla_Z \hat{\sigma}_2(Z^{(k)}) \\ &= \frac{1}{N} B(Z^{(k)}) Z^{(k)} - T + 2\lambda 1_{N \times N} Z^{(k)}. \end{aligned}$$

Applying the two-grid algorithm repeatedly, we converge to its fixed point, the fine grid minimizer Z_0^*.

The two-grid algorithm we have described appears to be a particular case of a *multigrid* algorithm. The idea of multigrid dates back to the papers of Fedorenko [157] and Bakhvalov [18] in the 1960s, though the method as we know it today was formulated a decade later by Brandt [49, 50] and Hackbusch [197]. Originally, multigrid methods were applied to partial differential equations and introduced relatively late to the field of numerical optimization

input : $(L+1)$-level hierarchy of data $D_0, \ldots, D_L$, interpolation operators $P_1^0, \ldots, P_L^{L-1}$, decimation operators $P_0^1, \ldots, P_{L-1}^L$, constants N_R, N_R'.
output : canonical form Z^*.
initialization: some initial $Z^{(0)}$ and $k = 0$.

1 **repeat**
2 $Z^{(k+1)} \leftarrow$ `V-cycle`$(Z^{(k)}, 0, D_0, N_R, N_R')$.
3 $k \longleftarrow k+1$.
4 **until** *convergence*
5 $Z^* = Z^{(k)}$.

Algorithm 7.6. MG-MDS algorithm with V-cycle outer iterations.

[135, 291]. In the multigrid jargon, Step 5 is called *relaxation*; interpolation and decimation are often referred to respectively as *prolongation* and *restriction*.

The two-grid algorithm can be easily extended by applying it on the coarse grid repeatedly in a recursive manner. In a general multigrid MDS algorithm (referred to as MG-MDS for brevity), we have a hierarchy of grids $\Omega_0, \ldots, \Omega_L$ and data $D_0, \ldots, D_L$, interpolation operators $P_1^0, \ldots, P_L^{L-1}$, and decimation

1 **if** $l = L$ *(coarsest resolution)* **then**
2 Solve the Lth level problem,

$$Z_L^* = \underset{Z_L \in \mathbb{R}^{N_L \times m}}{\operatorname{argmin}} \hat{\sigma}(Z_L; D_L) - \operatorname{trace}(Z_L^{\mathrm{T}} T_L),$$

using $Z_L^{(0)}$ as the initialization.
3 **return** Z_L^*.
4 **else**
5 Relaxation: apply N_R optimization iterations on $\hat{\sigma}(Z_l; D_l) - \operatorname{trace}(Z_l^{\mathrm{T}} T_l)$ initialized with $Z_l^{(0)}$, obtaining $Z_l^{(1)}$.
6 Restriction: $Z_{l+1}^{(1)} = P_l^{l+1} Z_l^{(1)}$.
7 Recursively apply the V-cycle on the coarser level,

$$Z_{l+1}^{(2)} \leftarrow \texttt{V-cycle}(Z_{l+1}^{(1)}, T_{l+1}, D_{l+1}, N_R, N_R').$$

8 Correction: $Z_l^{(2)} = Z_l^{(1)} + \alpha_l P_{l+1}^l (Z_{l+1}^{(2)} - Z_{l+1}^{(1)})$.
9 Relaxation: apply N_R' optimization iterations on $\hat{\sigma}(Z_l; D_l) - \operatorname{trace}(Z_l^{\mathrm{T}} T_l)$ initialized with $Z_l^{(2)}$, obtaining $Z_l^{(3)}$.
10 **return** $Z_l^{(3)}$
11 **end**

Algorithm 7.7. Function `V-cycle`$(Z_l^{(0)}, T_l, D_l, N_R, N_R')$

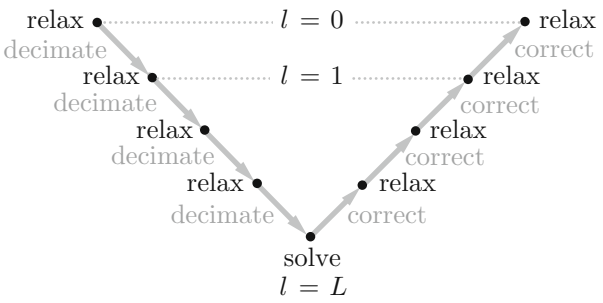


Figure 7.7. Illustration of the V-cycle iterative process.

operators $P_0^1, \ldots, P_{L-1}^L$. The algorithm first goes down from the finest grid to the coarsest grid, performing N_R relaxation iterations at each level, and then back to the finest grid, with N'_R relaxation iterations (typically, N_R and N'_R range from 1 to 5). Because of the resemblance of the outer iterative process to the letter "V" (Figure 7.7), it has been named the *V-cycle*, or, being more specific, the $V(N_R, N'_R)$-cycle. As in the two-grid algorithm, the V-cycles are repeated until convergence (Algorithms 7.6 and 7.7).

The cost of a V-cycle is usually comparable with a few SMACOF iterations. Yet, the MG-MDS algorithm requires much fewer iterations to converge. More importantly, the number of cycles (outer iterations) is approximately constant with N, unlike SMACOF, whose number of iterations tends to grow with N. This behavior is often observed in multigrid methods. For this reason, the use of multigrid offers a significant speedup of MDS algorithms, especially pronounced in large-scale problems.

Example 7.4 (multigrid acceleration). We demonstrate the MG-MDS method on the problem we have already seen in Example 7.2, Swiss roll embedding into $\mathbb{R}^3$. The Swiss roll surface is sampled at $N = 2145$ points, which results in a relatively large-scale MDS problem. We use a multigrid $V(3,3)$-cycle with SMACOF-type relaxation and compare it with the standard SMACOF algorithm. Both algorithms are initialized by the coordinates of the points on the original surface. To ensure a fair comparison, we first run SMACOF with the stopping criterion $\sigma_2(Z^{(k+1)})/\sigma_2(Z^{(k)}) \leq 0.01$, and then let the MG-MDS algorithm reach the same stress. Figure 7.8 depicts the L_2-stress on a logarithmic scale versus time for the SMACOF and multigrid algorithms. The MG-MDS algorithm converges in 229.46 sec (7 cycles), compared with 2.03×10^3 sec (341 iterations) of the SMACOF algorithm – almost an order of magnitude speedup.

7.8⋆ Vector extrapolation

As an alternative way to accelerate the convergence of the SMACOF iterations, Guy Rosman proposed using *vector extrapolation* [327, 326]. Such meth-

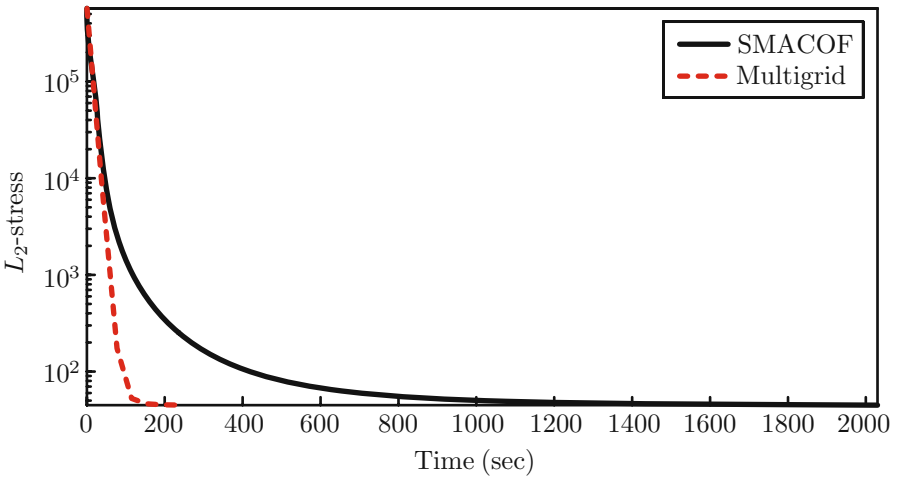


Figure 7.8. Convergence plots of the standard SMACOF algorithm (solid) and the multigrid V(3,3)-cycle with SMACOF-type relaxation (dashed) in the problem of Swiss roll embedding with $N = 2145$. Shown is the L_2-stress value $\sigma_2(Z^{(k)})$ as a function of CPU time.

ods were originally derived for speeding up slowly converging linear iterative minimization and numerical approximation schemes,[6] though in practice they also work well for nonlinear problems [51], like certain processes in computational fluid dynamics [359] and reconstruction in tomography [321, 386].

Assume that we have a sequence of iterates $Z^{(0)}, Z^{(1)}, \ldots$ produced by some optimization algorithm (e.g., SMACOF), which converges to the minimizer $Z^* = \lim_{k\to\infty} Z^{(k)}$. The key idea of vector extrapolation is the following: if $Z^{(k)}$ converges slowly, we can construct another sequence, denoted by $\hat{Z}^{(k)}$, which will converge to the same minimizer Z^*, but much faster. Writing the original sequence as $Z^{(k)} = Z^* + E^{(k)}$, where $E^{(k)}$ is the *remainder*, we are looking for a new sequence $\hat{Z}^{(k)} = Z^* + \hat{E}^{(k)}$ such that

$$\lim_{k\to\infty} \frac{\|\hat{E}^{(k)}\|}{\|E^{(k)}\|} = 0. \tag{7.18}$$

The new sequence is given as some transformation of the form $\hat{Z}^{(k)} = T(Z^{(k)}, \ldots, Z^{(k+K)})$, applied to $K+1$ iterates $Z^{(k)}, \ldots, Z^{(k+K)}$ produced by a standard optimization algorithm. The extrapolation is used as a new initialization to restart the optimization. The entire process is repeated for a few cycles (Algorithm 7.8). Because in general there is no guarantee for the extrapolation to succeed, a safeguard (Steps 4–8) must be used to ensure that the stress value does not increase.

It is common to construct $\hat{Z}^{(k)}$ as a linear combination of $K+1$ iterates,

$$\begin{aligned}\hat{Z}^{(k)} &= \gamma_0 Z^{(k)} + \ldots + \gamma_K Z^{(k+K)} \\ &= (\gamma_0 + \cdots + \gamma_K) Z^* + \gamma_0 E^{(k)} + \ldots + \gamma_K E^{(k+K)}\end{aligned} \tag{7.19}$$

input : $N \times N$ matrix of geodesic distances D_X.
output : canonical form Z^*.
initialization: some initial $Z^{(0)}$ and $k = 0$.

repeat
  Generate the sequence of iterates $Z^{(k+1)}, \ldots, Z^{(k+K)}$ by some optimization algorithm on $\sigma(Z; D_X)$, using $Z^{(k)}$ as initialization.
  Extrapolate $\hat{Z}^{(k)}$ from the iterates $Z^{(k)}, Z^{(k+1)}, \ldots, Z^{(k+K)}$.
  if $\sigma(\hat{Z}^{(k)}; D_X) > \sigma(Z^{(k+K)}; D_X)$ **then**
    $Z^{(k+K+1)} \longleftarrow Z^{(k+K)}$
  else
    $Z^{(k+K+1)} \longleftarrow \hat{Z}^{(k)}$
  end
  $k \longleftarrow k + K + 1$.
until *convergence of* $Z^{(k)}$
$Z^* = Z^{(k)}$.

Algorithm 7.8. MDS algorithm with vector extrapolation acceleration.

where γ_i are some coefficients. Various vector extrapolation algorithms differ in the definition of these coefficients. Ideally, the new sequence should be $\hat{Z}^{(k)} = Z^*$, such that one can try to select the coefficients γ_i in order to satisfy this requirement. Usually this is impossible to achieve, thus we will try to reduce the residual terms $\gamma_0 E^{(k)} + \ldots + \gamma_K E^{(k+K)}$ as much as possible, at the same time requiring $\gamma_0 + \cdots + \gamma_K = 1$ in equation (7.19).

Note that $E^{(k)} = Z^{(k)} - Z^*$ depends on the unknown Z^*. In order to eliminate this dependence, we can consider the difference $\Delta\hat{Z}^{(k)} = \hat{Z}^{(k+1)} - \hat{Z}^{(k)}$, which ideally should vanish. This leads to a constrained linear system,

$$\gamma_0 \Delta Z^{(k)} + \ldots + \gamma_K \Delta Z^{(k+K)} = 0 \quad \text{s.t.} \quad \gamma_0 + \ldots + \gamma_K = 1, \tag{7.20}$$

with Nm equations and $K+1$ variables. Because typically $K \ll Nm$, the system is over-determined and usually cannot be satisfied. We therefore compute the coefficients γ_i by finding a least-square solution to (7.20). This method of vector extrapolation is known as *reduced rank extrapolation* (RRE) [270, 144].[7]

If we parse the $N \times m$ matrices $\Delta Z^{(k)}, \ldots, \Delta Z^{(k+K)}$ into $Nm \times 1$ vectors $\Delta z^{(i)} = \mathrm{vec}(\Delta Z^{(i)})$ and denote $A_{K+1} = (\Delta z^{(k)} \ldots \Delta z^{(k+K)})$, we can formulate the RRE method as finding a least-squares solution to the following constrained over-determined linear system,

$$A_{K+1}\gamma = 0 \quad \text{s.t.} \quad \gamma_0 + \ldots + \gamma_K = 1. \tag{7.21}$$

which can be solved by first solving

$$A^{\mathrm{T}}_{(K+1)} A_{(K+1)} \tilde{\gamma} = 1_{(K+1)\times(K+1)}, \tag{7.22}$$

and then setting

$$\gamma = \frac{\tilde{\gamma}}{\tilde{\gamma}_0 + \ldots \tilde{\gamma}_K}$$

(prove this result in Problem 7.11).

For a numeric solution, the matrix A_{K+1} can be represented as $A_{K+1} = Q_{K+1}R_{K+1}$ using *QR factorization*, where Q_{K+1} is an $Nm \times (K+1)$ unitary matrix and R_{K+1} is a $(K+1) \times (K+1)$ upper triangular matrix [359]. This type of factorization can be carried out efficiently using the *modified Gram-Schmidt* algorithm [177]. Because $Q_{K+1}^{\mathrm{T}} Q_{K+1} = I$, equation (7.22) becomes $R_{K+1}^{\mathrm{T}} R_{K+1}\tilde{\gamma} = 1_{(K+1)\times(K+1)}$. Because of the triangular form of the matrix R_{K+1}, we can employ forward and backward substitutions, similarly to the solution of the Newton system with Cholesky factorization we have encountered in Chapter 5. The entire RRE method can be summarized as shown in Algorithm 7.9.

Example 7.5 (RRE acceleration). We demonstrate the vector extrapolation method on the problem of Swiss roll embedding into $\mathbb{R}^3$, with the same data and the same settings as in the previous Example 7.4. We compare the SMACOF algorithm with and without vector extrapolation acceleration. In the accelerated version, we use the RRE method with $K = 10$. Figure 7.9 (top) depicts the L_2-stress on a logarithmic scale versus time for both algorithms. The SMACOF algorithm without acceleration converges after 2.03×10^3 sec (341 iterations) compared with 256.59 sec (4 cycles) for the RRE-accelerated version. Figure 7.9 (bottom) shows the inner iterations of the RRE algorithm. The "jumps" in the stress values correspond with extrapolated values in each cycle.

The RRE approach gives nearly the same acceleration to the SMACOF algorithm as the multigrid scheme but is advantageous being significantly simpler and easier to implement. In addition to RRE, many other vector extrapolation methods exists, the most popular choices being *minimal polynomial extrapolation* (MPE) [91] and the *topological ϵ-algorithm* (TEA) [51]. We refer the reader to the above references for additional details.

input : sequence of iterates $\Delta Z^{(k)}, \ldots, \Delta Z^{(k+K)}$.
output : extrapolation $\hat{Z}^*$.

1 Compute the matrix $A_{K+1} = (\Delta z^{(k)} \ldots \Delta z^{(k+K)})$ and find its QR factorization $A_{K+1} = Q_{K+1}R_{K+1}$.
2 Forward substitution: solve $R_{K+1}^{\mathrm{T}} y = 1_{(K+1)\times(K+1)}$ for y.
3 Backward substitution: solve $R_{K+1}\tilde{\gamma} = y$ for $\tilde{\gamma}$.
4 Compute $\gamma = \tilde{\gamma}/(\tilde{\gamma}_0 + \ldots + \tilde{\gamma}_K)$.
5 Compute the extrapolation $\hat{Z}^{(k)} = \gamma_0 Z^{(k)} + \ldots + \gamma_K Z^{(k+K)}$.

Algorithm 7.9. Reduced Rank Extrapolation (RRE).

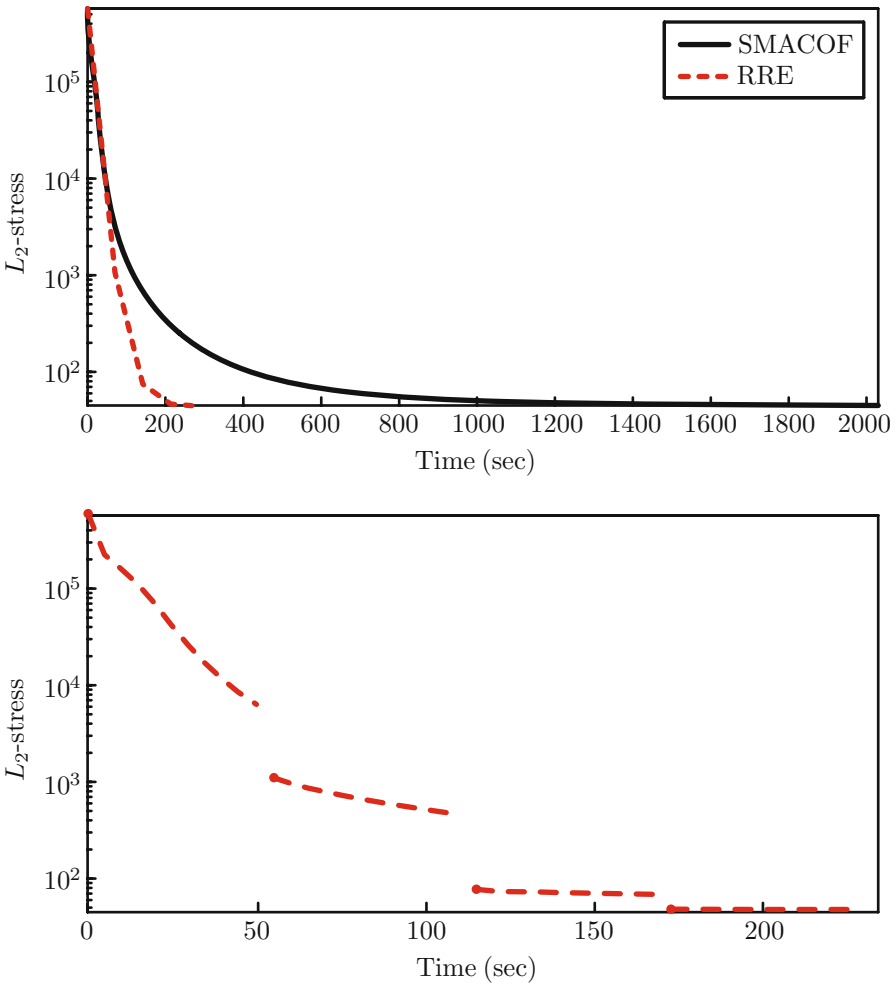


Figure 7.9. Convergence plots of the SMACOF algorithm with (top, dashed) and without (top, solid) RRE acceleration in the problem of Swiss roll embedding with $N = 2145$. Shown is the L_2-stress value in outer cycles as a function of CPU time. Values of the stress in inner iterations (bottom, dotted) and outer cycles (bottom, bold dots) of the RRE algorithm are shown in the bottom plot.

7.9 A trouble with topology

Using intrinsic similarity criterion for the comparison of non-rigid shapes, we have been tacitly assuming that the shapes have similar *topology*. Said differently, we did not allow our deformations to make holes or "glue" parts of the shapes. In certain situations, however, this assumption does not necessarily hold.

For example, when acquiring three-dimensional objects using range cameras, a phenomenon called *topological noise* is a well-known problem. A range camera estimates the three-dimensional coordinates of the object at a discrete set of points, producing a point cloud. The point cloud is then triangulated in order to obtain a triangular mesh representation of the object. Yet, if no additional information is assumed, the problem of constructing a mesh out of a point cloud is not well-defined. The same object can be triangulated differently, in some cases producing meshes with different topology. Typically, topological noise is manifested in the form of connections between vertices that should not be connected or vice versa.

Topological noise affects the intrinsic geometry of the shape. Connectivity changes can significantly alter the geodesics, and consequently, if we try to compute the canonical forms of two topologically different shapes, they may differ substantially (see example in Figure 7.10). We therefore conclude that intrinsic similarity can be used only for shapes undergoing topology-preserving deformations.

At the other end, extrinsic similarity is insensitive to topological noise, yet, as we have already seen, is sensitive to non-rigid deformations. In [74], we showed that it is possible to define a new criterion of similarity insensitive to some types of topological noise by combining extrinsic and intrinsic similarity criteria. We must note, however, that the general problem of shape similarity invariant under deformation not preserving topology appears as a very challenging problem, to which, to the best of our knowledge, no complete solution has been proposed thus far.

Suggested reading

For a comprehensive overview of MDS problems and algorithms, the reader is referred to the books by Cox and Cox [119] and Borg and Groenen [44]. A good treatment of MDS with a theoretical emphasis is the book by Young and Hamer [405]. In the fields of machine learning and neural networks, the concept of *self-organizing maps* introduced by Teuvo Kohonen [231] is closely related to MDS. A classic reading on multigrid methods is Briggs [54], McCormick [264], and Wesseling [393]. A more recent overview is the book by Trottenberg *et al.* [381]. A detailed description and experimental validation of the multigrid MDS method can be found in [79, 80]. For an introduction and overview of vector extrapolation methods, we refer to review papers by Sidi and coauthors [366, 362, 360, 363, 361] A different approach to vector extrapolation through the Shanks-Schmidt transformation is presented in [339, 351, 52]. The use of vector extrapolation for MDS problems is described in [327]. A different acceleration of the SMACOF algorithm was proposed by de Leeuw [127]. An attempt to address the problem of comparison of topologically different shapes using a combination of extrinsic and intrinsic similarity criteria is presented by the authors in [74].

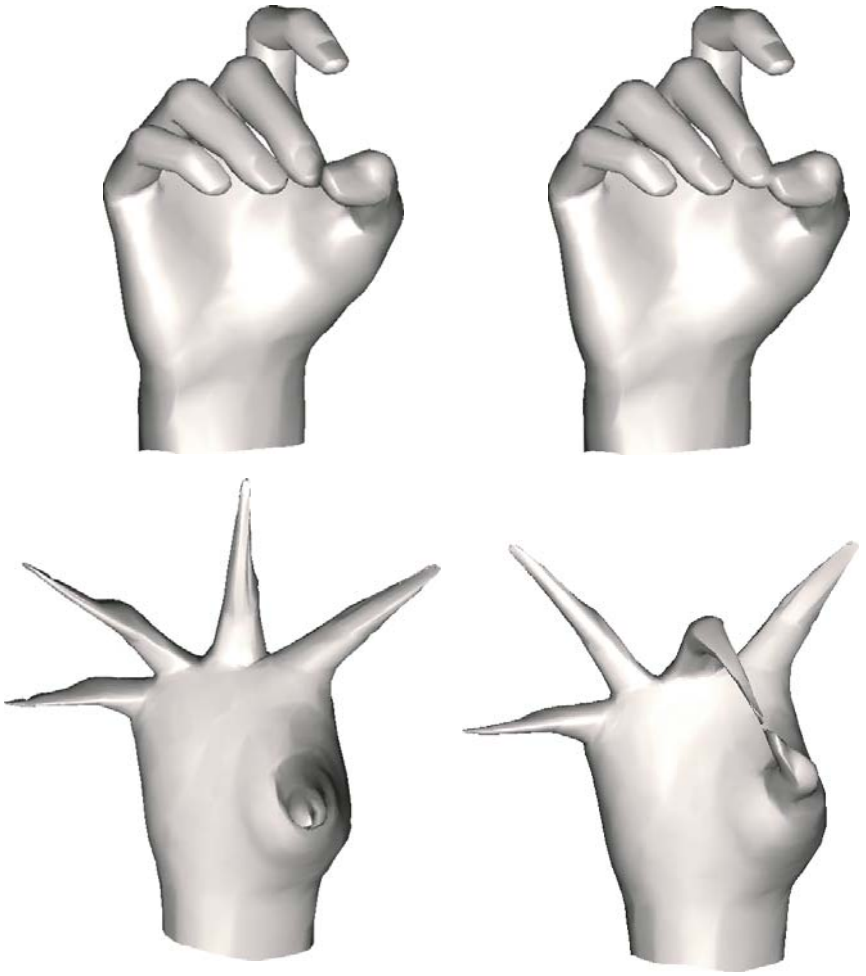


Figure 7.10. Illustration of the effect of the topological noise on the computation of canonical forms. Gluing the fingertips of the hand (right) changes the intrinsic geometry of this shape and as a result, the canonical form is substantially different from that of the original shape (left).

Software

A free MATLAB implementation of the SMACOF algorithm with vector extrapolation acceleration is distributed as part of the TOSCA toolbox. Mark Steyvers' Nonmetric MDS for MATLAB allows minimization of different variants of stress. The function `mdscale` supports different types of MDS problems and is distributed as part of the commercial MATLAB Statistics toolbox.

Problems

7.1. Verify that the L_2-stress function is non-convex.

7.2. Derive the matrix form of the L_2-stress in equation (7.4).

7.3. ✓ Derive the matrix form of the gradient of the L_2-stress in equation (7.5).

7.4. How does the fact that the matrix $B(Z)$ is zero-mean relate to the degrees of freedom in the MDS problem?

7.5. ✓ Prove the bound on the L_2-stress in inequality (7.7).

7.6. Derive the Hessian of the L_2-stress.

7.7. For the case $m = 2$, show a way to invert the Hessian matrix (7.9) using the inversion formula for block matrices.

7.8. Show that the constrained optimization problem (7.11) is equivalent to the L_∞-stress minimization optimization problem (7.10).

7.9. Derive the matrix form of the modified stress (7.17) and show that the addition of the quadratic penalty does not change the solution of the MDS problem.

7.10. ✓ Derive the relation (7.16).

7.11. ✓ Derive the equation (7.22).

Notes

[1]One peculiar example of the distortion introduced by cartographic projections is the fact that river Lena appears to be the longest river in the world when viewed on a map using a Mercator projection. In reality, Amazon and Nile hold this title.

[2]Borg and Groenen [44] give a different meaning to this acronym, *scaling by majorizing a complex function*. We find this a slight misnomer, as the term "complex function" usually refers to a complex-valued function.

[3]The presence of zero eigenvalues is directly related to the degrees of freedom in the MDS problem. An alternative way to resolve the degrees of freedom is to fix the positions of three non-collinear points [220].

[4]It was shown by Platt [314] that these three methods are equivalent.

[5]As we have already mentioned in Chapter 3, this cost can be reduced to $\mathcal{O}(N \log N)$ using efficient data structures. The complexity of MDS is often given as $\mathcal{O}(N^3)$, though, more rigorously, it should be proportional to N^2 multiplied by the number of iterations.

[6]There are evidences that prototypes of extrapolation methods are very old. Brezinsky [53] traces the history of speeding up convergence of sequences by means of extrapolation back to Christian Huygens (1629–1695).

[7]The derivation of RRE is done assuming that the sequence $Z^{(k+1)}, \ldots, Z^{(k+K)}$ is produced by a linear iteration formula [366]. In the nonlinear case, the algorithm can still be used, but some of its properties are lost.