Discrete differential geometry and dimensionality reduction

Claire Schnoebelen

June 8, 2022

Supervised by E. Franck, E. Opshtein, L.Navoret.

Contents

Introduction

In what follows, we consider a partial differential equation.

$$\begin{cases} \partial_t u = \mathcal{F}(u, \partial_x u, \partial_{xx} u, ...) & \forall x \in \Omega \quad \forall t \in I, \\ u(x,0) = u_0(x) & \forall x \in \Omega. \end{cases} \tag{1}$$

We make the hypothesis that the problem can in fact be written in lower dimension, say d, that is the long-term evolution of the system depends only on few variables and not on the full space of the initial conditions. Then, the equation (1) can be written

$$\frac{\partial}{\partial t} v = \tilde{\mathcal{F}}(v, \partial_x v, \partial_{xx} v, ...),$$

where v is the solution projected in the low dimensional space.

After space discretisation, the problem becomes

$$\begin{cases} \frac{d}{dt} X(t) = F(X(t)) & \forall t \in I, \\ X(0) = X^0 & \forall x \in \Omega. \end{cases} \tag{2}$$

with $X \in \mathbb{R}^m$ and $F : \mathbb{R}^m \to \mathbb{R}$ a function that depends on the scheme we have chosen for space discretisation. In the same way, the reduced version of the equation becomes

$$\frac{d}{dt} Z(t) = \tilde{F}(Z(t)) \qquad \forall t \in I. \tag{3}$$

Under our assumptions, there exists an *encoding operator* $E : \mathbb{R}^m \to \mathbb{R}^d$ and a *decoding* one $D : \mathbb{R}^d \to \mathbb{R}^m$, such that $D(Z) = X$ and $E(X) = Z$ with X the solution of (2) and Z the solution of the reduced equation.

The chain rule gives

$$\frac{d}{dt} X(t) = \mathcal{J}(D)(Z(t)) \frac{d}{dt} Z(t).$$

Let's consider $f : z \in \mathbb{R}^d \mapsto \frac{1}{2}\|x - Jz\|^2$ with $J \in \mathcal{M}_{m,d}(\mathbb{R})$. We have that $f(z+h) = f(z) + \langle h, J^T(Jz - x)\rangle + \langle h, J^T J h\rangle$, so f is twice differentiable with $\nabla f(z) = J^T(Jz - x)$ and $\nabla^2 f(z) = J^T J$. Let's consider the svd decomposition of $J = U\Sigma V$, with $U \in \mathcal{O}(m), V \in \mathcal{O}(d)$ and $\Sigma \in \mathcal{M}_{m,d}(\mathbb{R})$ such that

$$\Sigma = \begin{pmatrix} \lambda_1 & & \\ & ... & \\ & & \lambda_d \\ & 0 & \end{pmatrix},$$

with $\lambda_1 \leq ... \leq \lambda_d$.

Then, $J^T J = V^T \Lambda^2 V \in \mathcal{S}_d^+(\mathbb{R})$ with $\Lambda = diag(\lambda_1, ..., \lambda_d)$. If J has full column rank, $\lambda_1 > 0$ and $J^T J \in \mathcal{S}_d^{++}(\mathbb{R})$. It implies that f is λ_1-convexe and in partcular that it is strictly convexe and coercive. Then, f reaches its unique minimum on $\mathbb{R}^d$. It satisfies Euler's equation, that is $\nabla f(z) = 0$, ie $J^T(Jz - x) = 0$, which happens if and only if $z = (J^T J)^{-1} J^T x$. We will note $J^+ = (J^T J)^{-1} J^T$ the Moore-Penrose pseudo-inverse of J.

Applying this results on $z = \frac{d}{dt} Z, x = \frac{d}{dt} X$ and $J = \mathcal{J}(D)(Z(t))$, we have

$$\frac{d}{dt} Z(t) = \mathcal{J}(D)(Z(t))^+ F(D(Z(t))). \tag{4}$$

We want to build a numerical model to solve the reduced equation. Start by supposing that E and D are linear. In that case, (4) becomes

$$\frac{d}{dt}Z = A^+\tilde{F}(AZ), \tag{5}$$

with $A \in \mathcal{M}_{m,d}$ the matrix of the projection into the low dimensional space.

To build our model, we first consider a model to solve the original equation (1). We discretize the spacial domain Ω into m points x_i, the time interval into n points t^n and we note $\hat{X}^n \in \mathbb{R}^m$ the solution at a time t^n and at the point x_i. In what follows, we will use Euler scheme to approximate the time derivative.

Our model gives a discrete operator $\tilde{F} : \mathbb{R}^m \to \mathbb{R}^m$ such that :

$$\frac{X_i^{n+1} - X_i^n}{\Delta t} = \hat{F}(X^n).$$

Then, the corresponding scheme in low dimension is

$$\frac{Z_i^{n+1} - Z_i^n}{\Delta t} = A^+\hat{F}(AZ^n), \tag{6}$$

where $\hat{Z}^n \in \mathbb{R}^d$ denotes the solution of (5) at time $t = n\Delta t$. The initial condition Z^0 is given by $A^+X_0(x)$.

It then remains to find A. Recall that the solution of (1) can be express in d variables. In the linear case, it means that

$$X(x,t) = \sum_{i=1}^{d} a_i(t)\psi_i(x). \tag{7}$$

Then, if we compute the solution for a lot of different times t^n, we can expect that the eigenvectors of the resulting matrix $\hat{X} = (X_i^n)_{i,n}$ give a basis on which $X(x,t)$ can be decomposed as in (7) at any time t. The operator A that will be used in (6) is then the orthogonal matrix whose columns are the d eigenvectors associated to the d largest eigenvalues of $\hat{X}$. This process is known as the Proper Orthogonal Decomposition (POD).

Here, we will consider Burger's equation

$$\partial_t X(x,t) = \frac{\partial_x X^2}{2}(x,t) - \epsilon\partial_{xx}X(x,t) \qquad \forall x \in J \subset \mathbb{R} \quad \forall t \in I \subset \mathbb{R} \tag{8}$$

We will use Lax-Friedrichs scheme to build $\hat{F}$:

$$\hat{F}(X^n) = \frac{\mathcal{F}_{i+\frac{1}{2}} - \mathcal{F}_{i-\frac{1}{2}}}{2} - \epsilon\frac{X_{i+1}^n - 2X_i^n + X_{i-1}^n}{\Delta_x^2},$$

with

$$\mathcal{F}_{i+\frac{1}{2}} = \frac{\frac{(X_{i+1}^n)^2}{2} + \frac{(X_i^n)^2}{2}}{2} - c_{i+\frac{1}{2}}\frac{X_{i+1}^n - X_i^n}{2},$$

where $c_{i+\frac{1}{2}} = \max\{|X_{i+1}^n|, |X_i^n|\}$.

When ϵ is large enough, that is where there is enough diffusion in the equation to regularize the solution, this gives satisfying results, see Figure 1. This is not the case anymore when ϵ is too small, as we see in Figure 2. This means that the hypothesis that the solution is linear in time doesn't hold anymore.

Therefore, we should look for an operator E with less restrictive conditions than the linearity. Instead, we will make the assumption that the solution at any time t lies on a manifold and look for a map $\phi : X \in \mathcal{M}_{m,n} \to Y \in \mathcal{M}_{d,n}$ that best preserves the geometry of the samples X.

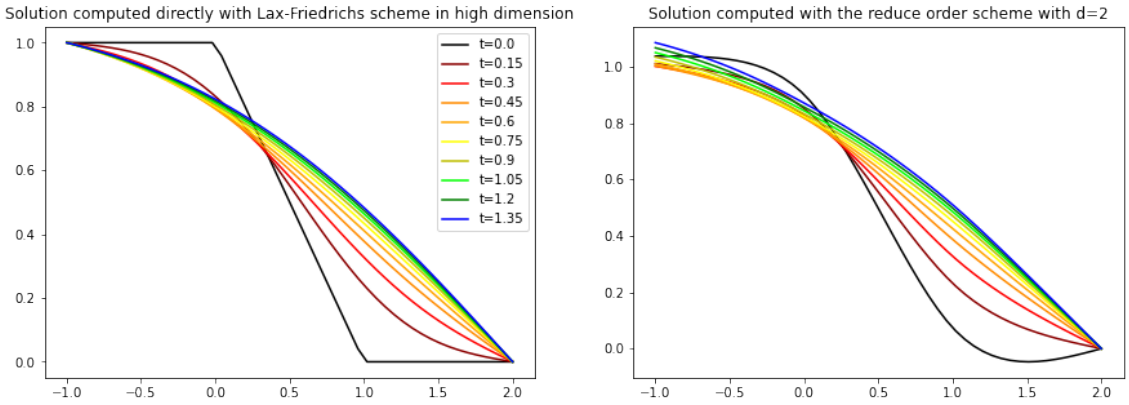

Figure 1: Numerical solution of Burger's equation with $\epsilon = 1$.The reduce order model gives results very similar to what we obtain with Lax-Friedrichs model in low dimension.

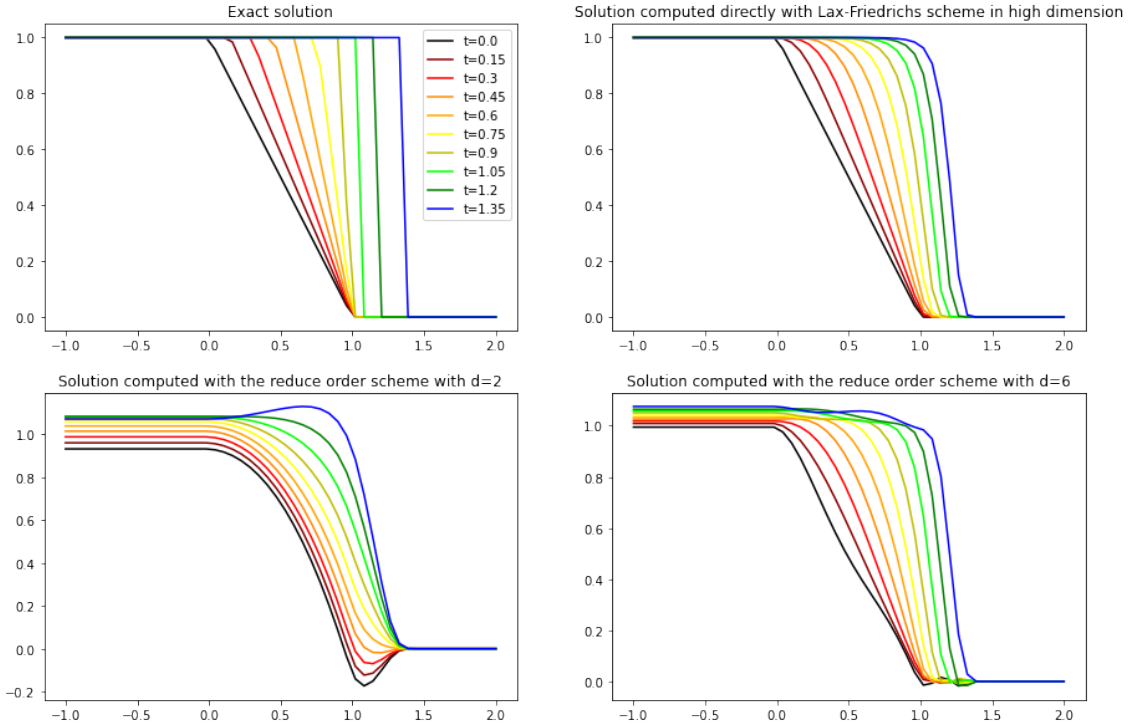

Figure 2: Numerical solution of Burger's equation with $\epsilon = 0$. The solution obtained with the reduce order model with $d = 2$ gives unaccurate results and with $d = 6$, it seems "noisy" for highest values of t next to singular points of the exact solution.

More generally, the question we will study is the following. Let χ be a set of n samples in $\mathbb{R}^m$ that we assume to lie on a d-dimensional manifold M. Although the dataset is described with m coordinates, its "intrinsic" dimension is much lower than m and the dataset can be described with only $m \gg d$ coordinates. How can we find, without any other information on the underlying manifold than the sampling χ, d variables to describe the dataset while preserving its geometry ?

1 Dimensionality reduction

What follows present three methods that are used to solve this kind of problem : Isomap [11], Parallel Transport Unfolding [4] and Eigenmap ([1]). The two first seek to preserve distances between the points in χ and in $\phi(\chi)$ while the last takes advantage of information provided by the Laplace-Beltrami operator of the underlying manifold.

All of them start by building a graph $\mathcal{G}$ that meshes the underlying manifold M. There are two ways to do this : we can either link all vertices at a distance in $\mathbb{R}^m$ inferior to a certain bound δ or link each vertice to its k-nearest neighbors in $\mathbb{R}^m$. The first way may have more geometrical sense but in practice, it is hard to make sure that the resulting graph is connected. This is why we often proceed in the second way.

Before introducing these methods, let first introduce MDS method, on which Isomap and PTU are based.

1.1 Multidimentional Scaling (MDS)

Let χ be a dataset composed of n samples. Assume that we have evaluated the similarity, or the dissimilarity, between each pair of χ. For a given $d \in \mathbb{N}$ the MDS method consists in finding a set of n points in $\mathbb{R}^d$ which represents as accurately as possible χ from the point of view of similarity. We present here how works the method, according to [12]. In other words, if two samples x_i and x_j are close, we want the distance between the corresponding $y_i, y_j \in \mathbb{R}^d$ to be small.

Let $D \in \mathcal{M}_{n,n}(\mathbb{R})$ be the matrix whose coefficients D_{ij} are equal to the similarity between the individuals x_i and x_j. We will call it *similarity matrix*. The optimal $Y \in \mathcal{M}_{d,n}(\mathbb{R})$ according to previous considerations minimizes the loss function

$$E = \sum_{i,j=1}^{n} (D_{i,j}^2 - d_2(y_i, y_j)^2)^2.$$

Note that what we call "similarity measure" may not be a distance : we can think of travel times between cities, which doesn't satisfy triangular inequality.

From now on, we will only consider datasets in $\mathbb{R}^m$ with $m \gg d$ and use the euclidian distance to compute similarities. We will denote by X the matrix whose columns x_i are the points of χ.

Definition 1.1 (Gram matrix)**.** *Let $X \in \mathcal{M}_{m,n}(\mathbb{R})$. The matrix given by $G = X^T X$ is called the Gram matrix associated to X. Conversely, every $G \in \mathcal{M}_{n,n}(\mathbb{R})$ which can be written this way is called a Gram matrix and the corresponding set X the configuration matrix of G.*

In an euclidian space, G corresponds to the matrix whose coefficients are the scalar products between the x_i. The polar formula gives the following relation between D and G :

$$D_{ij} = \sqrt{G_{ii} + G_{jj} - 2G_{ij}}. \tag{9}$$

Contrary to the distance matrix D, G is not invariant by translations. In the following, we will work with centered data, that is data whose barycenter is 0.

Definition 1.2 (Centering)**.** *1. The operator $H = I_n - \frac{1}{n}E$, with $E = \mathbb{1}^T\mathbb{1}$ is called centering operator.*

2. *We say that a dataset X is centered if $XH = X$.*

3. *A Gram matrix is said centered if it can be written $G = X^T X$ with X centered.*

Proposition 1.1. 1. *The centering operator is idempotent : $H^2 = H$.*

2. *The left multiplication of datasets by H is called centering and produces centered datasets.*

3. *The left and right multiplication of Gram matrix by H is called centering and produces centered Gram matrix.*

Proof. First, note that the first point implies the two following, since we have $(XH)H = XH$ and $H(HGH)H = HGH$. Let's prove the first point. For all $i, j \in \{1, ..., n\}$, $H^2_{ij} = \sum_{k=1}^{n} H_{ik}H_{kj}$. If $i = j$, exactly one term is equal to $(1 - \frac{1}{n})^2$ while the $n - 1$ others are $\frac{1}{n^2}$ so the sum is 1. If $i \neq j$, exactly two terms are $-\frac{1}{n}(1 - \frac{1}{n})$ while the $n - 2$ others are $\frac{1}{n^2}$ and the sum is 0. □

In some cases, X is not explicitly given and the advantage of centered data is that we can obtain G directly from D.

Proposition 1.2. 1. $D_{ij} = \sqrt{G^c_{ii} + G^c_{jj} - 2G^c_{ij}}$

2. $G^c = -\frac{1}{2}S^c$,

where G^c denote the centered matrix of the Gram matrix G and S is such that $S_{ij} = D^2_{ij}$.

Proof. As D is translation invariant, the first equality follows from (9). The second equality holds as we have for all $i, j \in \{1, ..., n\}$

$$\begin{aligned} S^c_{ij} =& (G_{ii} + G_{jj} - 2G_{ij}) - \frac{1}{n}\sum_k (G_{kk} + G_{jj} - 2G_{kj}) - \frac{1}{n}\sum_l (G_{ii} + G_{ll} - 2G_{il}) + \frac{1}{n}\sum_{kl}(G_{kk} + G_{ll} - 2G_{kl}) \\ =& -2G_{ij} + \frac{2}{n}\sum_k G_{kj} + \frac{2}{n}\sum_l G_{il} - \frac{2}{n^2}\sum_{kl} G_{kl} \\ =& -2G^c_{ij} \end{aligned}$$

□

The loss function can be written as

$$E(Y) = \|S - S_Y\|_F = 2\|G^c - G^2_Y\|_F,$$

where $\|\cdot\|_F$ denotes the Foebenius norm and S_Y, G_Y the squared distances and the Gram matrices associated to the projected data Y.

Proposition 1.3. *For all $G \in S^{++}_n(\mathbb{R})$, the problem $min\|G - A\|_F$ admits an unique solution on $\{A \in \mathcal{M}_{n,n}(\mathbb{R}) \quad | \quad rg(A) = d\}$.*

Proof. As G is symmetric, it can be written as $P^T \Lambda P$, with $P \in O(n)$ and Λ the diagonal matrix whose coefficients $\lambda_1 < ... < \lambda_n$ are the eigenvalues of G. As G is positive semidefinite, those coefficients are non-negative. Note $B = PAP^T$. We have

$$\|G-A\|^2_F = Tr((G-A)^T(G-A)) = Tr(P(G-A)^T(G-A)P^T) = \|\Lambda - B\|^2_F = \sum_i (\lambda_i - B_{ii})^2 + \sum_{ij, j\neq i} B^2_{ij}.$$

It is obvious that $\|\Lambda - B\|^2_F \geq \|\Lambda - diag(B)\|^2_F$ so the minimum, if it exists, should be such that B is a diagonal matrix. Moreover, as $rg(B) = rg(A)$, $n - d$ diagonal coefficients of B should be 0. Note I the set of indices where $B_{ii} \neq 0$. We have :

$$\|\Lambda - B\|^2_F = \sum_{i\in I}(\lambda_i - B_{ii})^2 + \sum_{i\notin I}\lambda^2_i \geq \sum_{i\notin I}\lambda^2_i \geq \sum_{i=1}^{n-d}\lambda^2_i.$$

The first inequality is an equality if and only if $B_{ii} = \lambda_i$ for all $i \in I$ and the second if and only if $I = 1, ..., n-d$. Then, $\|G - A\|_F$ admits a unique minimum and it is given by $P^T \tilde{\Lambda} P$, where $\tilde{\Lambda}$ is the diagonal matrix whose last d coefficients are equal to those of Λ and the others are 0. □

It follows from this that

Proposition 1.4. *The problem* $\min_{Y \in \mathcal{M}_{d,n}(\mathbb{R})} E(Y)$ *admits a unique solution up to orthogonal transformations.*

Proof. We have seen that minimizing E is equivalent to minimizing $\|G^c - G_Y^c\|$. But if Y is in $\mathcal{M}_{d,n}(\mathbb{R})$, then G_Y has rank at most d. By the previous proposition, $\|G^c - A\|$ is minimal if and only if $A = P^T \tilde{\Lambda} P$. As G^c is positive semidefinite, its eigenvalues are positive and $\tilde{\Lambda} = \Sigma^2$, where Σ is the diagonal matrix whose coefficients are the square root of the λ_i. Then, $A = (\Sigma P)^T (\Sigma P)$ is the Gram matrix associated to $Y = \Sigma P$ and this dataset is in $\mathcal{M}_{d,n}(\mathbb{R})$.

If Y' is another minimizer of E, then it is associated to the same Gram matrix, ie $Y'^T Y' = A_{min} = Y^T Y$. As such a decomposition for a semidefinite matrix is unique up to orthogonal transformations, we conclude that there exists $Q \in O(d)$ such that $Y' = QY$. □

Finally, to find the reduction Y of a dataset X minimizing E, one has to :

1. Compute the centered Gram matrix associated to X : $G^c = -\frac{1}{2} HSH$.
2. Decompose G as $P^T \Lambda P$.
3. Take the d eigenvectors associated to the d largest eigenvalues and multiply them by the square root of their associated eigenvalues.

1.2 Isomap

Now, consider $\chi \in M$, where (M, g) is a d-dimensional riemannian manifold embedded in $\mathbb{R}^m$, with $m \gg d$. The idea of the Isomap, proposed in [11], is to apply MDS to a distance matrix that approaches distances in the manifold.

Suppose that M is isometric to an open set of $\mathbb{R}^d$. In that case, there exists a map $\phi : M \to U \in \mathbb{R}^d$ such that $dist_M(x, y) = dist_{\mathbb{R}^d}(\phi(x), \phi(y))$ for all $x, y \in M$. Without further information about M we want to find a map $\tilde{\phi}$ that verifies the above equality on the x_i.

Then, $Y = \tilde{\phi}(X)$ minimizes the same loss function that we have introduced for MDS :

$$E(Y) = \|S_M - S_Y\|_F,$$

where S_M is the matrix of the squared distances in M.

By the hypothesis we made on M, $D_M = D_d$, with $(D_d)_{ij} = dist_{\mathbb{R}^d}(\phi(x_i), \phi(x_j))$. A solution to the problem is then obtained by applying MDS method on D_M.

Note that under our assumptions on M, the result Y of MDS verifies $E(Y) = 0$. Indeed, as $G_M^c = G_d^c = \phi(X)^T \phi(X)$ with $\phi(X) \in \mathcal{M}_{d,m}$, the Y found with the MDS is equal to $\phi(X)$ up to orthogonal transformations, which means that $D_Y = D_{\phi(X)} = D_M$ and then that $E(Y) = E(\phi(X)) = 0$.

In practice, as we don't have any information about M, exact distances on the manifold are not available and we should estimate them. To do this, we use that by definition

$$d_M(x, y) = \inf_{\gamma \in \Gamma} \int_0^1 \|\dot{\gamma}\| dt,$$

where $\Gamma = \{\gamma : [0,1] \to M \in \mathcal{C}^1_{pm} \quad | \quad \gamma(0) = x \wedge \gamma(1) = y\}$.

If the graph $\mathcal{G}$ meshes M sufficiently well, then one can expect that the curves on M can be approximated by paths on G, and that the length of the shortest path on $\mathcal{G}$ between two vertices x and y

approximates well the exact distance between this two points on M. It can be proved that, under some assumptions on the sampling and the manifold, the first distance converges to the second as the number of points increases.

More precisely, let d_S and d_G such that for all $x, y \in V$,

$$d_G(x,y) = \min_P \sum_i \|x_i - x_{i+1}\|_n$$
$$d_S(x,y) = \min_P \sum_i d_M(x_i, x_{i+1}),$$

where P represents all the paths in $\mathcal{G}$ linking x and y.

We have :

Theorem 1. *Let $\epsilon > 0$ and $\delta > 0$ such that $\epsilon > 4\delta$. Suppose that :*

1. *All edges xy such that $d(x,y) < \epsilon$ are in $\mathcal{G}$.*
2. *For every point m in M, there is a vertice x of $\mathcal{G}$ such that $d_M(x,m) < \delta$.*

Then, for all vertices x, y, we have

$$d_M(x,y) \leq d_S(x,y) \leq (1 + 4\frac{\delta}{\epsilon}) d_M(x,y).$$

and

Theorem 2. *Let $\lambda_1, \lambda_2 < 1$. Assume that M is compact and geodesically convex.*

Assume furthermore that there exists $\epsilon_{min}, \epsilon_{max}$ and $\delta > 0$ such that :

1. *All edges xy such that $\|x,y\|_n < \epsilon_{min}$ are in $\mathcal{G}$.*
2. *All edges xy of $\mathcal{G}$ satisfy $\|x,y\|_n < \epsilon_{max}$.*
3. *For every point m in M, there is a vertice x of $\mathcal{G}$ such that $d_M(x,m) < \delta$.*

with

1. *$\epsilon_{max} \leq \frac{2}{\pi} r_0 \sqrt{24\lambda_1}$, where r_0 is the minimum radius of curvature of M,*
2. *$\epsilon_{max} < s_0$, where s_0 is the minimum branch separation of M, ie the largest positive number for which $\|x - y\| < s_0$ implies $d_M(x,y) < \pi r_0$,*
3. *$\delta \leq \lambda_2 \frac{\epsilon_{min}}{4}$.*

Then

$$(1 - \lambda_1) d_M(x,y) \leq d_G(x,y) \leq (1 - \lambda_2) d_M(x,y).$$

Proofs of those theorems are given in [3].

Estimating the distances on M between two vertices x and y now reduces to find the shortest path on $\mathcal{G}$ between them. This can be done using Dijkstra algorithm, described for example in [12].

It consists in walking through all paths starting from point x by visiting at each step all branches starting from a point whose distance to x has already been established and by updating the distance of nodes that have not yet been processed. Before the first iteration, all the points except x are considered to be at a distance infinite from x. During the first iteration, we associate to every neighbors of x the weight given by the edge which links it to x. Then x is removed from the list of untreated points and the point with the smallest distance to x is considered.

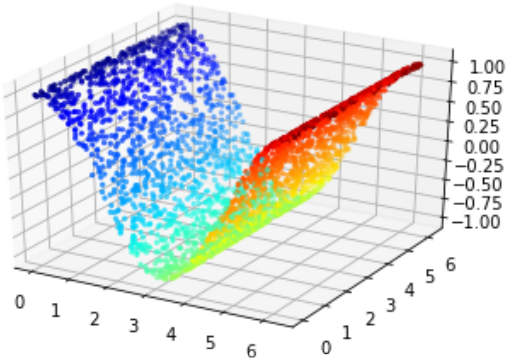

Figure 3: Sampling of 3000 points of a "V" surface, colored according to the second coordinate.

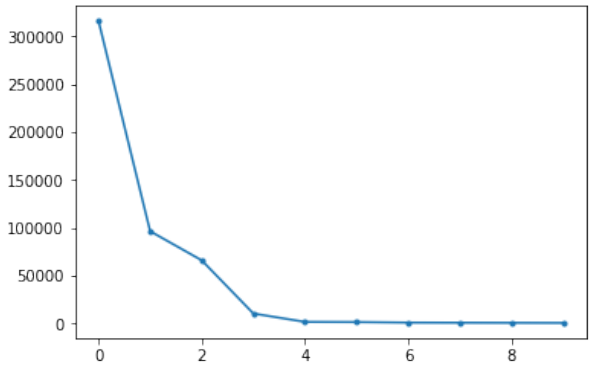

Figure 4: Eigenvalues of the kernel G obtained from the 7-neighbors graph of a sampling of 3000 of the V-shape. We see a gap between the first three ones and the others and an elbow between the second and the third, which means that the process succeed to catch the two dimensionality of the data.

More generally, the k-th iteration deals with the untreated point z whose distance to x after the step $k-1$ is minimal. This point is added to the list of treated points and its distance to x is given by the last update we made. Then, we consider all its untreated neighbors z' and update their respective distances to x by taking for each of them the minimum between the one that has already been found and $d_G(x,z) + d_G(z,z')$.

To find each of the n^2 distances in the graph, this should be done for each vertice x.

Finally, implementing Isomap on a dataset $X \in \mathcal{M}_{m,n}(\mathbb{R})$ requires the following steps :

1. Construction of the graph $\mathcal{G}$.
2. Use of Dijkstra algorithm.
3. Application of MDS on the distance matrix D given by the previous step.

Let's now see how Isomap works on a test example. We will consider a sampling of 3000 points of a "V" surface, shown in Figure 3.

We construct our graph using k-nearest neighborhoods and then apply Dijkstra algorithm. This gives a kernel G with $2-3$ strong eigenvalues (see Figure 4).

The projection on the first two eigenvectors gives a rectangle, as we expected (Figure 5).

Although Isomap has been built under the assumption that the underlying manifold is isometric to an open set of $\mathbb{R}^d$, the method can give good results even if this is not true. In practice, this is in fact

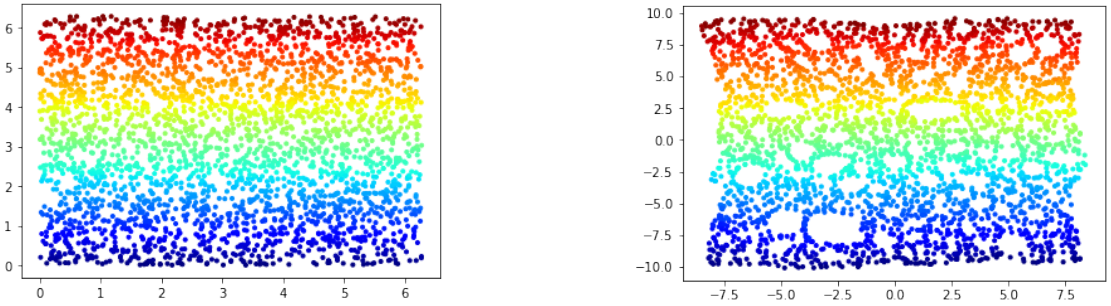

Figure 5: Comparison between conform projection on two dimensions (left) and Isomap result (right). Colors represent the position of the point on the enrolling. This shows that Isomap succeeded to "unroll" the swiss roll, which is exactly what we expected.

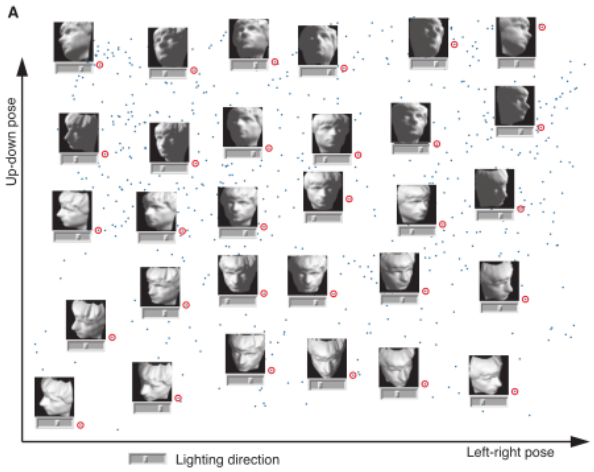

Figure 6: Projection of the heads pictures on the first two eigenvectors of the Isomap kernel, obtained from a 6-neighbors graph, from [11]. We see clearly that the two axes are correlated with anges of the camera, which is what was expected.

impossible to know if this hypothesis is verified by the dataset we are working on. In [11] is shown an example where Isomap was executed on a dataset composed of 698 pictures of the same head which differ from view angle and illumination. Even if the variations of angle in the shots may suggest that these images could be points of a portion of a sphere, there is no way to verify this. And even then, no portion of a sphere is isometric to a plane. This dataset has therefore no chance to verify the assumptions we made about the underlying manifold before building the method. And yet, Isomap works.

All of those pictures are composed of 64×64 pixels of different gray values but we expect that the "intrinsic" dimension of the dataset is 3, ie the degrees of freedom of the camera angle (2) and the light (1). This is in fact really what we see when we apply Isomap : the residual variance, that is $E(Y) = \sum_{i=d}^{n} \lambda_i^2$ with λ_i the eigenvalues of the kernel G, decreases very strongly between $d = 1$ and $d = 3$ (see [11]), which means that Isomap succeeded to capture the 3-dimensional caracteristic of the dataset. The projection on the two first eigenvectors shows that the first two coordinates of the projected sample Y corresponds to the variations of the camera angle (Figure 6, taken from [11]).

1.3 Parallel Transport Unfolding

As we can see in Figure 5, when there is a "hole" in the sampling, because there are not enough points or because a whole part of the possible values has not been sampled, distortions appears on the projection made with Isomap.

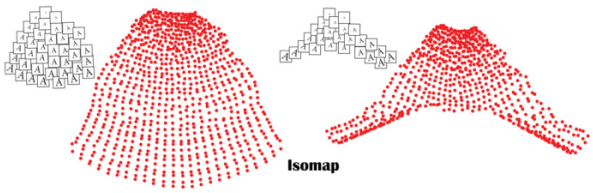

Figure 7: Projection of a sample of 888 images of 120×120 pixels of rotated and resized letters 'A', with (right) and without (left) removing a part of the sample, from [4]. We clearly see the distorsions induced by the removing of a part of the training data.

Suppose that when was made the sampling, a whole part of possible values that can be taken by the x_i has not been sampled, as it has been made in [4] (Figure 7, form [4]). This "hole" in the data forces all paths in the graph connecting points at two distinct ends of the hole to deviate considerably from the initial shortest path. As a result, the distance between two points at two ends of the hole is considerably overestimated by the Dijkstra algorithm.

On the contrary, the distance between two points next to each other and close to the edge of the hole is not modified, which means that they look closer compared to the other points. As one can see on Figure 9, taken from [4], all of this leads to a "flattening" of the points around the hole, which is enlarged.

Therefore, the insertion of new points in the data is problematic. Indeed, the projected data completed with removed points is unlikely to look like it would have if the projection had been built with the full dataset: the model generalizes poorly.

Distance computation as made by Dijkstra algorithm is problematic even if there is no hole in the sampling. Take a plane sampled with a regular mesh. If the number of neighbors in the graph is inferior to 5, the distance between two points on the same diagonal will be dilated by a factor of $\sqrt{2}$, while the one between two points on the same horizontal or vertical line will be correctly computed. At the end, Isomap will give a very distorted projection, no matter how refine was the sampling, as one can see in Figure 8.

These problems come from the way Dijkstra algorithm estimates distances. In the second case, Dijkstra alogrithm sums the norms of the edges, whereas we would rather sum vectors before taking the norm. But if we generalize this, it would lead to take euclidian distances in $\mathbb{R}^m$, what we precisely don't want to.

Indeed, the whole point of calculating distances with Dijkstra's algorithm is to take into account the curvature of M. What is problematic is that it also takes into account the curvature resulting from the particular paths used on M. If we were able to compute distances by reducing the overestimation induced by the curvature of the path while retaining the effect of the curvature of the manifold, we would have a more accurate estimate of the true distances on the manifold.

The first example underlines another point. Where we construct Isomap, we said that we wanted to estimate

$$d_M(x, y) = \inf_{\gamma \in \Gamma} \int_0^1 \|\dot{\gamma}\| dt,$$

where $\Gamma = \{\gamma : [0, 1] \to M \in \mathcal{C}_p^1 m | \gamma(0) = x \wedge \gamma(1) = y\}$. Here, Γ depends on what we consider for M. If we made the assumption that the whole manifold is contained in a neighborhood of all our samples, we fix Γ, and then d_M, and we forbid ourselves to get away from the samples. This is not problematic if we have a good sample or if we won't try to generalize our projection. In that cases, it could be interesting to consider a manifold not necessarily "stuck" to the sample.

Thus, the first idea in Parallel Transport Unfolding (PTU), proposed in [4], is to assume that the underlying manifold is *geodesically complete*.

Definition 1.3. *A riemannian manifold (M, g) is said to be geodesically complete if all the geodesics are defined on $\mathbb{R}$.*

In particular, it means that there is no "hole" in the manifold. In that case, the theorem of Hopf-Rinow says that for any two points in M, the distance between them is reached by a geodesic.

These curves parallel transport their own tangent vector field, which means that the vectors tangent to the curve at each point are identifiable, or that this vector field does not vary for the manifold.

It can be shown that for all $p \in M$ and all $v \in T_pM$, there is a unique geodesic γ verifying $\gamma(0) = p$ and $\gamma'(0) = v$. We will denote it by $\gamma_{p,v}$. This property leads us to introduce the *exponential map*

$$exp_p : \begin{cases} T_pM & \to & M \\ v & \mapsto & \gamma_{p,v}(1). \end{cases}$$

If $q = exp_p(v)$ is close enough to p, then $dist_M(p, q) = \|v\|$. Furthermore, if $r = exp_q(w)$ is close enough from q, then the distance between r and $exp_p(v + \tilde{w})$ with $\tilde{w}$ the parallel transport of w to T_pM along $\gamma_{p,v}$, is bounded by $R\epsilon^2$, where ϵ is such that $\|v\|, \|w\| \leq \epsilon$ and R is a constant which depends on the curvature of M at p.

Suppose that we know the vectors verifying $exp_{x_k}(v_k) = x_{k+1}$ for all x_k on the shortest path from x to y. Suppose also that we are able to parallel transport the vectors of one tangent space to another along the piecewise geodesic curve connecting x to y through the x_k. We can then approximate $d_M(x, y)$ by $dist(x, exp_x(v_1 + \tilde{v}_2 + ... + \tilde{v}_l))$, which is $\|v_1 + \tilde{v}_2 + ... + \tilde{v}_l\|$. Following the direction $v_1 + \tilde{v}_2 + ... + \tilde{v}_l$, we take a geodesic which leads us close to y. We then make few diversions from the minimal geodesic between x and y and we obtain a good approximation of the distance between them. However, the fact of passing through many points of M makes it possible to take into account the curvature of the manifold : we "roll" a path around it. This is the addition of the vectors that compensate the deviations made from the shortest curve by passing through this intermediate points.

It now remains to compute the v_k and to find a way to parallel transport them from one tangent space to another. Let us start by computing the tangent spaces at the vertices x of the graph. This is the d-dimensional vector space that best approximates the manifold in the neighborhood of x. It is also the set of vectors tangent to the curves drawn on M at the moment they pass through x. To estimate it, we consider all the edges that start from x and join its K nearest neighbors. Each edge is assumed to approach a portion of a curve on the manifold, itself close to its tangent line at x.

By diagonalizing the matrix composed of the K vectors corresponding to these edges and considering the d eigenvectors associated with the largest eigenvalues, we obtain the preferred directions of variation of the manifold around x, in other words a basis of the tangent space at x. This basis is orthonormal. Note that K can be different from k if $\mathcal{G}$ has been constructed with k neighborhoods.

The estimation of the v_k can then be done in two ways. The first way to proceed consists in projecting edges onto $T_{x_k}M$, where the projection involved is the orthogonal projection on $\mathbb{R}^m$. The second one consists in taking the same projection $\Pi x_k x_{k+1}$ and renormalizing it in order to obtain a vector whose norm is equal to the length of the projected edge : $\frac{\|x_k x_{k+1}\|}{\|\Pi x_k x_{k+1}\|} \Pi x_k x_{k+1}$. Note that in the second case we take the edge for the minimizing geodesic, whose image by exp_{x_k} is v_k.

All that remains now is to find a way to connect two tangent spaces. Let us start by considering the case where M is a vector space. Then, all the tangent spaces are parallel in $\mathbb{R}^m$ and the parallel transport simply translates vectors in $\mathbb{R}^m$. It is then sufficient to find the orthogonal transformation which sends the basis of tangent space to the one of the other to be able to compare the vectors expressed in these two bases.

Now, if M is not a vector space but a curved and smooth submanifold of $\mathbb{R}^m$, it is in particular of class $\mathcal{C}^2$. Then, if two points are close, then their tangent space are close too, that is "almost parallel" as subspaces of $\mathbb{R}^m$. If there is enough vertices in the graph, computed tangent spaces approach the variety well enough to suppose that this assumption still holds, i.e. that the two tangent spaces that we have computed for two neighboring points are close. The idea is then to find the orthogonal transformation R in $\mathbb{R}^d$ that best align the two spaces, in other words, which minimizes the loss function

$$F(R) = \|T_i - T_j R\|_F,$$

where T_k denotes the tangent space at x_k.

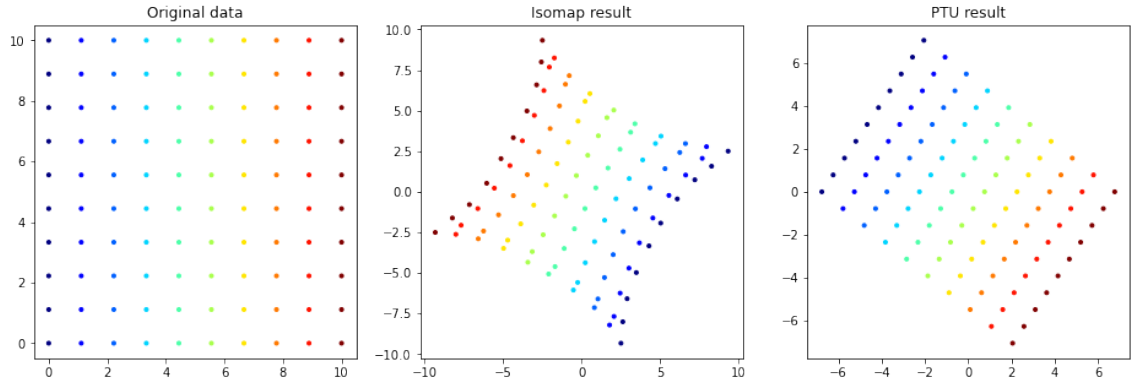

Figure 8: Comparison between PTU and Isomap results when applied to a plane sampled with a regular grid (n=100). Both methods have been used with $k = 4$. The resulting graphe is a grid composed of vertical and horizontal lines regularly spaced. Isomap estimations of distances leads to overestimate distances between two points located on the same diagonal while the one between two points on the same horizontal or vertical line is exactly computed, hence distorsions visible here. On the contrary, adding vectors before taking the norm in PTU avoids overestimations due to path detours on the graph.

Proposition 1.5. *The loss function F reaches its minimum at $R = VU^T$, where U and V are such that $T_i^T Tj = U\Lambda V^T$.*

Proof. As $R \in \mathcal{O}(d)$, we have $\|T_j R\|_F^2 = \|T_j\|_F^2$ so

$$\|T_i - T_j R\|_F^2 = \|T_i\|_F^2 - 2Tr(T_i^T T_j R) + \|T_j\|_F^2.$$

Then, minimizing F is equivalent to maximize $Tr(T_i^T T_j R)$. The Cauchy-Schwartz inequality gives

$$Tr(T_i^T T_j R) = Tr(U\Lambda V^T R) = Tr(V^T RU\Lambda) = \langle V^T RU, \Lambda\rangle \leq \|V^T RU\|_F \|\Lambda\|_F,$$

and the equality holds if and only if $V^T RU = \mu\Lambda$, that is iff $V^T RU$ is a diagonal matrix. As $V^T RU \in \mathcal{O}(d)$, this happens if and only if $V^T RU = Id$. Finally, $F(R)$ is minimal iff $R = VU^T$. □

Thus, PTU is achieved after the following steps :

1. Computation of $\mathcal{G}$,
2. Computation of tangent spaces T_i,
3. Computation of rotations R_{ij},
4. Use of Dijkstra algorithm with the additional multiplications by the R_{ij} and the sum of the vectors instead of the sum of the distances.

On Figure 8 and 9 we compared PTU and Isomap results when applied to sample made on a plane. In 8 we present what return the two methods if the sampling is regular. In 9 we show what happens if the points are uniformly distributed on the plane. On both cases, we see that PTU succeed to reproduce exactly the data configurations, up to an orthogonal transformation. For the reasons we detailed below, Isomap results are distorted.

1.4 Eigenmap

If Isomap and PTU give good results, these two methods have a high computational cost. This is due to Dijkstra algorithm, whose complexity is in $\mathcal{O}(n^2 \log(n))$ ([4]). Thus, it can be interesting to see if we couldn't find a "good" projection without computing distances in the manifold.

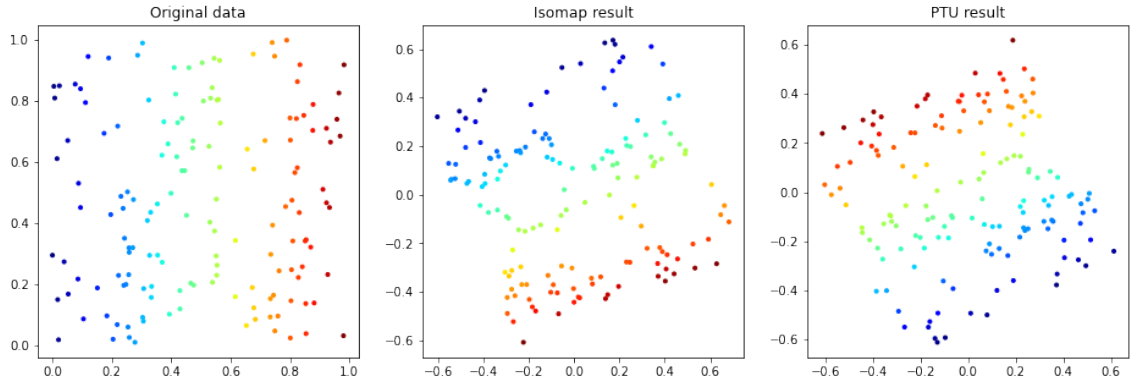

Figure 9: Comparison between PTU and Isomap results when applied to a plane uniformly sampled (n=150). We have taken $k = 7$ for Isomap and $k = 4$ for PTU. The same problem as in Figure (8) appears as the sampling is sparse : paths on the graph deviates of lot from the minimizing trajectorie. As a result, holes in the data are enlarged and points around it are squashed in it. This does not happen in PTU.

In the Eigenmap method, proposed in [1], the idea is to exploit information provided by the Laplacian of a manifold. Another way to justify this approach is to consider the following inequality :

Proposition 1.6. *Let $f : M \to \mathbb{R}$ twice differentiable and x, z two close points in M. Then,*

$$|f(x) - f(z)| \leq dist_M(x, z)\|\nabla f(x)\| + o(dist_M(x, z)).$$

The proof is taken from [1].

Proof. Let $l = dist_M(x, z)$ and c the minimal geodesic curve connecting x and z. Then, by Cauchy-Schwartz inequality,

$$\begin{aligned} f(z) =& f(x) + \int_0^l d_{c(t)} f(c'(t))dt \\ =& f(x) + \int_0^l \langle \nabla f(c(t)), c'(t)\rangle dt \\ \leq& f(x) + \int_0^l \|\nabla f(c(t))\|\|c'(t)\|dt. \end{aligned}$$

Since c is parametrized by length, we have $\|c'(t)\| = 1$ for all $t \in [0, l]$. On another hand, Taylor's approximation gives $\|\nabla f(c(t))\| = \|\nabla f(x)\| + o(l)$. We have then

$$f(z) \leq f(x) + dist_M(x, z)\|\nabla f(x)\| + o(l),$$

hence the desired inequality. □

This shows that ∇f controls the distance where two points x and z of M are send by a function $f : M \to \mathbb{R}$. If we want that two close points in M are send by f to two close points in $\mathbb{R}$, a good way to choose f is to take a minimizer of

$$E(f) = \int_M \|\nabla f(x)\|^2 dx. \tag{10}$$

To remove a scaling factor, we impose that $\|f\| = 1$. We also want to avoid f send all the manifold on a single point, which is the case if we consider the minimum of E in the unit sphere. Therefore, we impose another condition on f, which is $\langle f, \mathbb{1}\rangle = 0$. By doing this, we remove a translation invariance.

After integrating by parts, $E(f)$ can be written as

$$\int_M \mathcal{L}f \cdot f,$$

where $\mathcal{L}$ denotes the Laplace-Beltrami operator on the manifold M. It is a positive semidefinite operator whose *spectrum* is discrete [5]. Thus, the function that minimizes E on the unit sphere is the eigenfunction associated to the smallest eigenvalue of this operator. As it is $\mathbb{1}$, associated to the eigenvalue 0, and as eigenfunctions associated to different eigenvalues are orthogonal, the minimum of E under the specified conditions is reached on the unit eigenfunction associated to the second smallest eigenvalue.

This argument can be generalized to the case where we look for $f : M \to \mathbb{R}^d$, with $d > 1$. The solution is then given by the d first unit eigenfunctions associated to the d smallest positive eigenvalues of $\mathcal{L}$.

The question is now to know how to approach the eigenfunctions of the Laplace-Beltrami operator. To do that, we build an operator on the graph $\mathcal{G}$ whose eigenvectors converge, in a certain sense, to eigenfunctions of $\mathcal{L}$.

The idea, given in [8], is to define this operator as $\Delta = d^*d$, where d is an operator build as the gradient operator on a graph and d^* is its adjoint for a certain inner product. The following construction of the Laplacian of the graph is taken from [8].

Let's start by defining a structure of Hilbert space on $\mathcal{G}$. Denote by V the set of vertices of $\mathcal{G}$ and $E = V \times V$. Let $\mathbb{R}^V$ and $\mathbb{R}^E$ be the sets of functions from V and E to $\mathbb{R}$. We associate to each edge x_ix_j in $\mathcal{G}$ a positive weight w_{ij} and to each vertice x_i the weight $d_i = \sum_j w_{ij}$. For our purpose, it is sufficient to consider only undirected graphs, that is such that $w_{ij} = w_{ji}$.

Proposition 1.7. *The maps*

$$\langle \cdot, \cdot \rangle_V : \begin{cases} \mathbb{R}^V \times \mathbb{R}^V & \to & \mathbb{R} \\ (f, g) & \mapsto & \frac{1}{n} \sum_{i=1}^n f_i g_i \end{cases}$$

and

$$\langle \cdot, \cdot \rangle_E : \begin{cases} \mathbb{R}^E \times \mathbb{R}^E & \to & \mathbb{R} \\ (f, g) & \mapsto & \frac{1}{2n} \sum_{i,j=1}^n f_{ij} g_{ij} \end{cases}$$

define inner products on $\mathbb{R}^V$ and $\mathbb{R}^E$.

Proof. It follows from the fact that we can identify $\mathbb{R}^V$ to $\mathbb{R}^n$ and $\mathbb{R}^E$ to $\mathbb{R}^{n^2}$. Then, the two maps corresponds to the euclidian inner product on these spaces, up to a constant. □

We note $\mathcal{H}_V$ and $\mathcal{H}_E$ the spaces $\mathbb{R}^V$ and $\mathbb{R}^E$ endowed with their corresponding inner products. We can now define the operators d and d^*.

Definition 1.4. *1. Let γ be a real positive function. We call difference operator the operator $d : \mathcal{H}_E \to \mathcal{H}_V$ such that*

$$df(e_{ij}) = \gamma(w_{ij})(f_j - f_i).$$

2. We call adjoint difference operator the operator $d^* : \mathcal{H}_V \to \mathcal{H}_E$ such that

$$\langle df, u \rangle_E = \langle f, d^*u \rangle_V \ \forall u \in \mathcal{H}_E \ \forall f \in \mathcal{H}_V.$$

Proposition 1.8. *The adjoint diffence operator is given by*

$$d^*u(x_k) = \frac{1}{2n} \sum_{i=1}^n \gamma(w_{ij})(u_{ik} - u_{ki}).$$

Proof. In the definition of the adjoint, we take $f_l(j) = \mathbb{1}_{j=l}$. On the right-hand side of the equation that defines the adjoint we have :

$$\langle f_l, d^*u\rangle_V = \sum_i f_l(i)(d^*u)(i) = \frac{1}{n}d^*u(i).$$

On the other side :

$$\langle df_l, u\rangle_E = \frac{1}{2n^2}\sum_{i,j} df_l(e_{ij})u_(ij) = \frac{1}{2n^2}\sum_{i,j}\gamma(w_{ij})(f_l(i) - f_l(j))u_(ij) = \frac{1}{2n^2}\sum_i(\gamma(w_{il})u_{il} - \gamma(w_{li})u_{li}).$$

Therefore, we have

$$d^*u(i) = \frac{1}{2n}\sum_{i=1}^n \gamma(w_{il})(u_{il} - u_{li}).$$

□

We can now define the Laplacian on the graph as proposed below.

Proposition 1.9. *The Laplacian on the graph $\Delta : \mathcal{H}_V \to \mathcal{H}_V$ is defined as $\Delta = d^*d$. It is explicitly given by :*

$$\Delta f(i) = \frac{f(i)}{n}\sum_{j=1}^n \gamma^2(w_{ij}) - \frac{1}{n}\sum_{j=1}^n f(j)\gamma^2(w_{ij}).$$

Proof. Let $f \in \mathcal{H}_V$. We have

$$\begin{aligned}\Delta f(i) =& d^*df(i) = \frac{1}{2n}\sum_{l=1}^n \gamma(w_{il})(df(il) - df(li))\\ =& \frac{1}{2n}\sum_{l=1}^n \gamma(w_{il})(\gamma(w_{il})(f_l - f_i) - \gamma(w_{li})(f_i - f_l))\\ =& \frac{1}{n}\sum_{l=1}^n \gamma^2(w_{il})(f_l - f_i).\end{aligned}$$

□

The definition of Δ depends on the choice we make for γ. In the following, we will consider the Laplacian obtained with $\gamma(w_{ij}) = \sqrt{\frac{w_{ij}}{d_i}}$.

A good reason to do so is that it can be shown that under certain conditions on M, W and the sampling, this Δ converges (on a certain sense) to $\frac{1}{p}div(p^s grad)$, where p is the probability density from which the x_i where sampled. The exact statement as well as the proof can be found in [8].

Then, the operator is given by

$$\Delta f(i) = f(i) - \frac{1}{n}\sum_{l=1}^n \frac{w_{ij}}{d_i}f(j),$$

that is $\Delta = Id - D^{-1}W$, with D the diagonal matrix whose coefficients are $D_{ii} = d_i$. We verify that Δ is symmetric positive semidefinite, and that its first eigenvector is $\mathbb{1}$, as we expected. Indeed, for all $x \in \mathbb{R}^n$,

$$x^T\Delta x = \sum_{i=1}^n x_i^2 - \sum_{i\neq j=1}^n x_i x_j \frac{W_{ij}}{d_i} \geq \sum_{i=1}^n x_i^2 - \frac{1}{2}\sum_{i\neq j=1}^n (x_i^2 + x_j^2)\frac{W_{ij}}{d_i} \geq \sum_{i=1}^n x_i^2 - \sum_{i=1}^n x_i^2(\sum_{j=1}^n \frac{W_{ij}}{d_i}) = \sum_{i=1}^n x_i^2 - \sum_{i=1}^n x_i^2 = 0.$$

The initial problem reformulated in the discrete case is

$$\min_K y^T(Id - D^{-1}W)y,$$

with

$$K = \{v \in \mathbb{R}^n | \quad \|y\| = 1 \wedge \langle y, \mathbb{1} \rangle = 0\}.$$

It admits a unique solution, given by the eigenvector associated to the smallest positive eigenvalue. If we look for $Y \in \mathcal{M}_{d,n}$, we have to take the d eigenvectors associated to the smallest positive eigenvalues.

The last thing to do to compute these eigenvectors is to choose the weight w_{ij}. We will follow the choice made in [1]. Lets go back to the continuous case and consider the heat equation :

$$\begin{cases} \partial_t u + \mathcal{L} u = 0 \\ u(\cdot, 0) = f, \end{cases}$$

for $f \in L^2(M)$.

We know that the solution to this equation is given by

$$u(x,t) = \int_M H_t(x,y) f(y) dy,$$

where $H_t(x,y) = (4\pi t)^{-d/2} e^{-d_M(x,y)^2/4t} f(y) \Big(\phi(x,y) + \mathcal{O}(t) \Big)$, with ϕ such that $\phi(x,x) = 1$, is the *heat kernel*. Proofs can be found in [5].

We have then

$$\mathcal{L} f(x) = \mathcal{L} u(x,0) = -\frac{\partial}{\partial t} u(x,t)|_{t=0} = -\frac{\partial}{\partial t} \left[\int_M H_t(x,y) f(y) \right] |_{t=0}.$$

If x and y are close and if t is small,

$$\mathcal{L} f(x) \approx \frac{1}{t} \Big(f(x) - (4\pi t)^{-d/2} \int_M e^{-\|x-y\|^2/4t} f(y) dy \Big).$$

The integral is finally approched by $\frac{1}{n} \sum_{i=1}^n e^{-\|x_j - x_i\|^2/4t} f(x_i)$.

This motivates us to take $W_{ij} = e^{-\|x_j - x_i\|^2/4t}$ if x_i and x_j are at distance inferior to t in $\mathbb{R}^m$ and $W_{ij} = 0$ if it is not the case.

In [6] and [10] is proposed an interpretation of the parameter t. If we consider that our samples represent states of a physical system such as moving particles, what we look for is dynamically meaningful slow variables, that is coordinates that represent the long-term evolution of the system. Then, t represent the diffusion time during which we observe the system. When t is too small, we may not capture evolution of the system for a long enough time but when t is too large, the previous approximations don't hold anymore. A method to choose the optimal value for t is proposed in [10].

Finally, the method require the following steps :

1. Construction of $\mathcal{G}$.
2. Construction of W and D.
3. Computation of the eigenvectors associated to the smallest positive eigenvalues of $Id - D^{-1}W$.

In practice, if we know the dimension of the manifold we consider, it can be interesting to consider more than d eigenvectors. In fact, if M contains two connected components, or if its mass is mainly contained in two separated parts, then two eigenvectors associated to the smallest eigenvalues are constant on each of these components. This is easy to see if we consider the initial optimisation problem (10). In the same way, if M contains k connected components, the k first eigenvectors are constant on theses components. This phenomenon is illustrated on Figure 10. This is very meaningful if we are interested in doing clustering but it doesn't give good reduction coordinates as it does not provides information about the shape of each component. We have to search this information on the following eigenvectors.

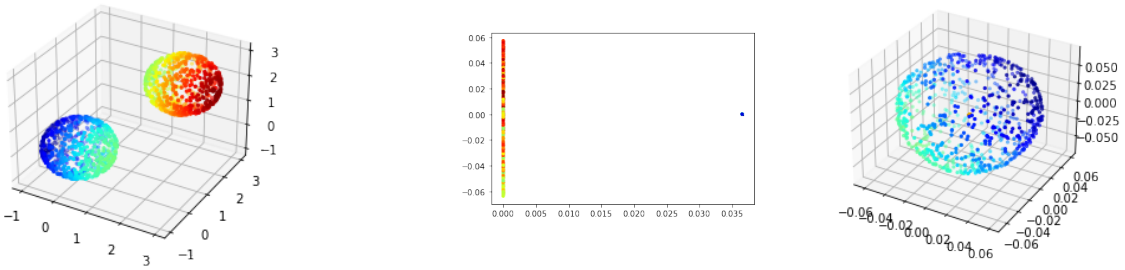

Figure 10: Eigenmap results for a manifold composed of two disjoined spheres (left). The first eigenvector separates the two connected components but gives any further information about their shapes (enter). If we take three of the six following eigenvectors, we can recreate one of the two sphere. The other is can be recreated from the three others.

Another case where it could be interesting to consider more eigenvectors is the case where M is anisotropic. If the manifold embedded in $\mathbb{M}$ is very elongated in a certain direction, then the second eigenvalue will be a function of the first one. Then, the information carried by the two first eigenvectors is redundant and it would be more meaningful to consider the following eigenvector. For instance, in $[0, l] \times [0, m]$, the eigenfuctions of the Laplacian are

$$\sin(\frac{j\pi}{l}x)\sin(\frac{k\pi}{m}y),$$

associated to the eigenvalues $(\frac{j\pi}{l})^2 + (\frac{k\pi}{m})^2$. We wee that if $l > 2m$, the two first eigenfunctions will be functions of y only. The Figure 11 illustrates the same problem on an ellipsoid : if we take the two first eigenvectors as prescribed in the method, we obtain a 1-dimensional figure, far from what we could expect. Taking the first and the third gives a better result.

2 Generalization to out-of-sample points

Methods presented in the previous section give a low-dimensional representation of a sample, optimal according to several criteria, but they don't provide a simple way to compute the projection of a new point in the reduced space. The only way to proceed is to apply the method again, on the sample enlarged with the new point. Given thoe computational cost of these methods, this is not an option. We would rather like to construct a map $f : \mathbb{R}^m \to \mathbb{R}^d$ which generalizes the projection found on the sample. An efficient way to proceed is to use reproducing kernels. The following results are taken from [9] and [7].

Definition 2.1 (Kernels, PDS kernels). *1. Given a set χ, a map $K : \chi \times \chi \to \mathbb{R}$ is called a kernel.*

2. A kernel is said positive definite symmetric (PDS) if for all $\{x_1, ..., x_n\} \in \chi$ the matrix $(K(x_i, x_j))_{ij}$ is symmetric positive semidefinite.

What makes these map interesting is the following property :

Theorem 3 (Reproducing Hilbert space). *Let K be a PDS kernel. Then, there exists a Hilbert space H and a mapping $\phi : \chi \to H$ such that*

$$\forall x, y \in \chi \quad K(x, y) = \langle \phi(x), \phi(y) \rangle.$$

The Hilbert space H is called the reproducing kernel Hilbert space (RKHS) associated to K and verifies

$$\forall h \in H, \quad \forall x \in \chi \quad h(x) = \langle h, K(x, .) \rangle.$$

Before starting the proof of the theorem, let's state the following lemma, which is the equivalent of Cauchy-Schwartz inequality for kernels :

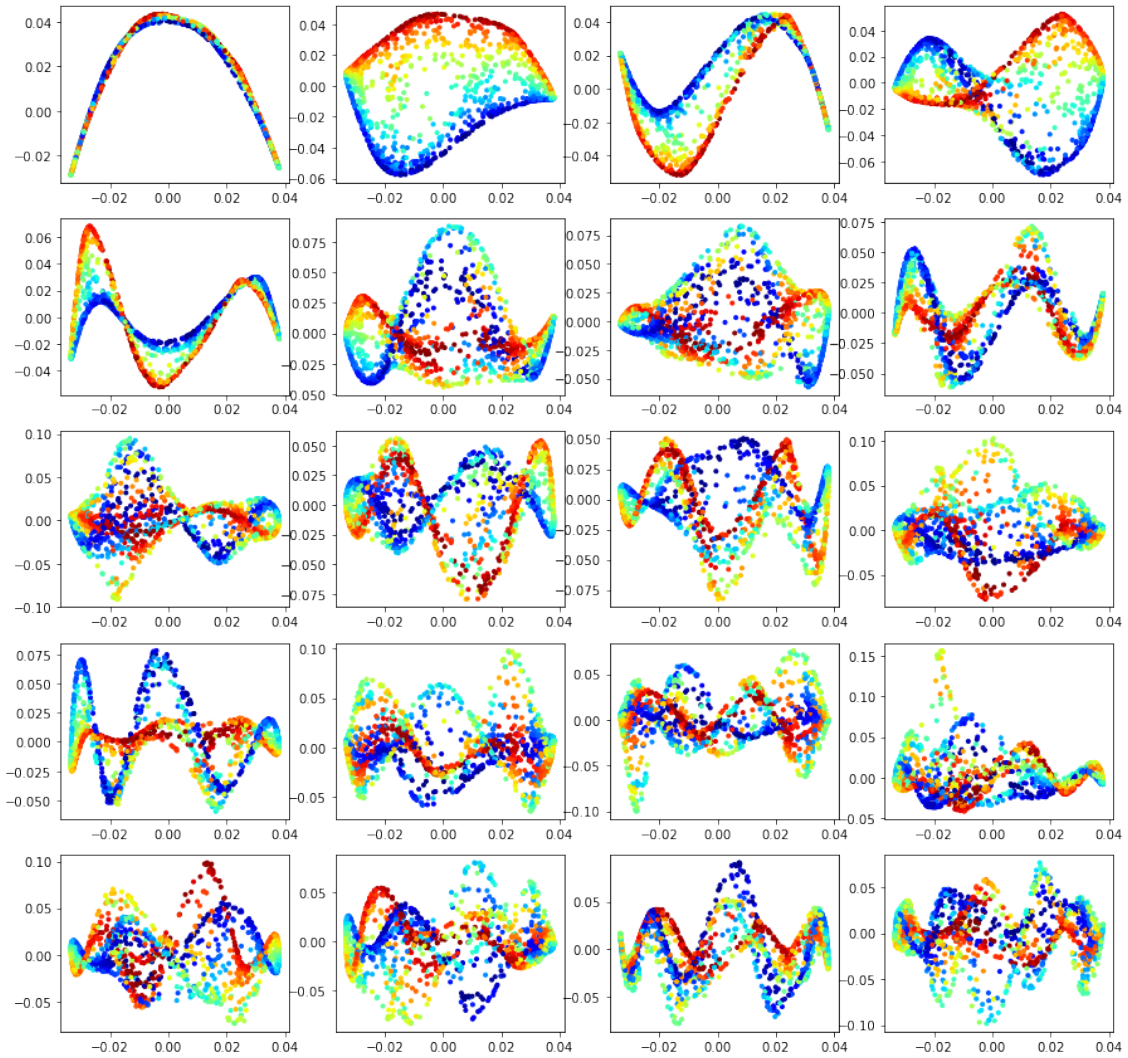

Figure 11: Eigenmap results for an ellipsoid twice longer than larger. On each graphic is represented the data projected on the first and the second, third, etc. eigenvectors. One can see that taking the second eignevector gives no further information about data. It has more sense to choose the third eigenvector, which seems not to be correlated with the first.

Lemma 2.1 (Cauchy-Schwartz inequality for kernels)**.** *Let K be a PDS kernel. Then, for any $x, y \in \chi$, $K(x,y)^2 \leq K(x,x)K(y,y)$.*

Proof. As K is PDS, the matrix

$$A = \begin{pmatrix} K(x,x) & K(x,y) \\ K(y,x) & K(y,y) \end{pmatrix}$$

is symmetric positive semidefinite. Then, $det(A) = K(x,x)K(y,y) - K(x,y)^2 \geq 0$. □

Proof. For any $x \in \chi$, set $\phi(x) : \chi \to \mathbb{R}^\chi$ such that $\phi(x)(y) = K(x,y)$. Consider the set of finite linear combinations of those functions

$$H_0 := \{\sum_{i \in I} a_i \phi(x_i) \quad | \quad \forall i \in I x_i \in \chi \quad |I| < \infty\}$$

and endow it with $\langle \cdot, \cdot \rangle$ such that

$$\forall f = \sum_{i \in I} a_i \phi(x_i),\ g = \sum_{j \in J} b_j \phi(x_j), \quad \langle f, g \rangle = \sum_{i \in I, j \in J} a_i b_j K(x_i, x_j).$$

Note that $\langle f, g \rangle = \sum_{i \in I} a_i g(x_i) = \sum_{j \in J} b_j f(x_j)$, so $\langle f, g \rangle$ is well-defined. It verifies the reproducing property, as $f(x) = \sum_{i \in I} a_i K(x_i, x) = \langle f, \phi(x) \rangle$ for all $x \in \chi$ and $f \in H_0$.

It it clearly bilinear. As K is PDS, $(K(x_i, x_j))_{i \in I, j \in J}$ is a symmetric matrix, which implies that $\langle \cdot, \cdot \rangle$ is symmetric. The PDS hypothesis also implies that $(K(x_i, x_j))_{i,j \in I}$ is a positive semidefinite matrix so

$$\langle f, f \rangle = \sum_{i,j \in I} a_i a_j K(x_i, x_j) \geq 0$$

for all $f \in H_0$, which means that $\langle \cdot, \cdot \rangle$ is positive semidefinite.

To prove the definiteness of $\langle \cdot, \cdot \rangle$, notice that for all $\{f_i\}_{i \in I} \subset H_0$ and $\{c_i\}_{i \in I} \subset \mathbb{R}$,

$$\sum_{i,j \in I} c_i c_j \langle f_i, f_j \rangle = \langle \sum_{i \in I} c_i f_i, \sum_{i \in I} c_i f_i \rangle \geq 0.$$

This implies that $\langle \cdot, \cdot \rangle$ is a PDS kernel. We can apply Cauchy-Schwartz inequality for kernels and have for all $x \in \chi$

$$f(x)^2 = \langle f, \phi(x) \rangle^2 \leq \langle f, f \rangle K(x,x),$$

which show the definiteness of $\langle \cdot, \cdot \rangle$. Then $(H_0, \langle \cdot, \cdot \rangle)$ is a pre-Hilbert space.

We set H as the completed of H_0. By the Cauchy-Schwartz inequality, $f \mapsto \langle f, \phi(x) \rangle$ is continuous for all $x \in \chi$. As H_0 is dense in H, the reproducing property also holds in H. □

The other result that is particularly interesting for our purpose is the following

Theorem 4 (Representer theorem)**.** *Let K be a PDS kernel over $\chi = \{x_1, ..., x_n\}$ and H its RKHS. For any loss function $L : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$, the optimization problem*

$$\underset{h \in H}{\operatorname{argmin}} L(h(x_1), ..., h(x_m))$$

has a solution of the form $\sum_{i=1}^n a_i K(x_i, \cdot)$.

Proof. Let H_1 be the vector subspace generated by the $K(x_i, \cdot)$. For any $h \in H$, let h_1 denote its orthogonal projection on H_1. By the reproducing property, $h(x_i) = \langle h, K(x_i, \cdot) \rangle = h_1(x_i)$, so $L(h) = L(h_1)$. □

This result allows us to reduce an optimization problem in H, which can be of infinite dimension, to a problem in $\mathbb{R}^n$. The one we are interested in here is to minimize on a space of function whose regularity will be specified the loss function

$$L(h) = \sum_{i \in I} \|y_i^k - h^k(x_i)\|,$$

where the x_i are the data in the high dimensional space and the y_i^k the kth coordinate of there projections in $\mathbb{R}^d$.

If we take H as the RKHS of a certain kernel, the previous theorem say we can rewrite the problem as

$$\underset{\alpha \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{n} \|y_i^k - \sum_{j=1}^{n} a_j^k K(x_i, x_j)\|.$$

Thus, what we look for is $A \in \mathcal{M}_{d,n}(\mathbb{R})$ such that

$$L(A) = \sum_{k=1}^{d} \sum_{i=1}^{n} |Y_{ik} - (AK)_{ik}|^2 = \|Y - AK\|_F^2,$$

where $K_{ij} = K(x_i, x_j)$ is minimal.

The function E is differentiable and its differential function is given by

$$d_A L(H) = -2\langle (Y - AK)K^T, H \rangle.$$

It is 0 for all $H \in \mathcal{M}_{n,d}(\mathbb{R})$ if and only if $(Y - AK)K^T = 0$. Then, $A = YK^T(KK^T)^{-1}$.

It is possible that KK^T is singular. Then, we reformulate the problem by adding a regularizing term to the loss function. We then look for the minimum of

$$L'(A) = \sum_{k=1}^{d} \sum_{i=1}^{n} |Y_{ik} - (AK)_{ik}|^2 + \alpha \sum_{k=1}^{d} \sum_{i=1}^{n} A_{ik}^2.$$

The differential of this function is

$$d_A L'(H) = -2\langle (Y - AK)K^T - A, H \rangle$$

so it is 0 if and only if $A = YK^T(KK^T + \alpha Id)^{-1}$.

The matrix A computed, we set $h(x)_k = \sum_{i=1}^{n} A_{ki} K(x_i, x)$.

Now, it remains to choose the kernel K. In practice, we often choose the gaussian kernel, defined as $K_\sigma(x, y) = e^{-\|x-y\|^2/2\sigma^2}$. It is a good choice as it is possible to prove that the gaussian kernel RKHS is dense in $\mathcal{C}^0(\mathbb{R}^m)$ ([7]). The space in which we look for h is then sufficiently large to avoid underlearning.

We will also use the laplacian kernel, defined as $K_\gamma(x, y) = e^{-\gamma\|x-y\|_1}$.

Let's show that the gaussian kernel is PDS. To do this, we will first introduce new properties of PDS kernels.

Proposition 2.1. *PDS kernels are closed under sum, product, pointwise limit and composition with a power series with positive coefficients.*

Proof. Let K and K' two PDS kernels and $\mathbb{K}, \mathbb{K}'$ the corresponding matrices generated from a set $\{x_1, ..., x_p\} \subset \chi$. As $x^T \mathbb{K} x \geq 0$ and $x^T \mathbb{K}' x \geq 0$, $x^T(\mathbb{K} + \mathbb{K}')x \geq 0$, which shows that $K + K'$ is PDS.

As $\mathbb{K}$ and $\mathbb{K}'$ are PDS, they can be written as $\mathbb{K} = MM^T$ and $\mathbb{K}' = MM^T$. The kernel matrix associated to KK' is $(K_{ij}K'_{ij})_{ij}$. Then, for any $x \in \mathbb{R}^m$

$$\sum_{i=1}^{m} x_i x_j \mathbb{K}_{ij} \mathbb{K}'_{ij} = \sum_{i=1}^{m} \sum_{k=1}^{m} x_i x_j M_{ik} M_{jk} \mathbb{K}'_{ij} = \sum_{k=1}^{m} z_k^T \mathbb{K}' z_k \geq 0$$

with $(z_k)_i = c_i M_i k$. This shows that KK' is PDS.

Let $(K_n)_n$ a sequence of PDS kernels whose pointwise limit is K. We have that $\mathbb{K}_n$ is positive semidefinite for all $n \in \mathbb{N}$, so $x^T \mathbb{K}_n x \geq 0$. It follows that $x^T \mathbb{K} x \geq 0$, which shows that K is PDS.

Finally, let $\sum_{n\in\mathbb{N}} a_n x^n$ be a power serie with positive radius of convergence ρ and such that all the a_n are non-negative. Let K a PDS kernel such that $K(x,y) < \rho$ for all $x, y \in \mathbb{R}^m$. For all $N \in \mathbb{N}$, $\sum_{n=1}^{N} a_n K^n$ is a PDS kernel as sum and product of PDS kernels. Therefore, $\sum_{n\in\mathbb{N}} a_n K^n$ is a PDS kernel as pointwise limit of PDS kernels. □

This shows that the laplacian kernel is PDS, as

$$K_\gamma(x,y) = e^{-\gamma\|x-y\|_1} = \prod_{i=1}^{n} e^{-\gamma|x_i-y_i|} = \prod_{i=1}^{n}\sum_{k=0}^{\infty} \frac{|x_i-y_i|^k}{k!},$$

with exp a power serie of infinite radius of convergence.

We also introduce the notion of normalized kernel.

Proposition 2.2. *Let K be a kernel. Then*

$$\frac{K(x,y)}{\sqrt{K(x,x)K(y,y)}}$$

is the normalized kernel associated to K. If K is PDS, then its normalized kernel is PDS too.

Proof. Let $c \in \mathbb{R}^m$. We have

$$\sum_{i=1}^{m} c_i c_j \frac{K(x_i,x_j)}{\sqrt{K(x_i,x_i)K(x_j,x_j)}} = \sum_{i=1}^{m} c_i c_j \frac{\langle\phi(x_i),\phi(x_j)\rangle}{\|\phi(x_i)\|\|\phi(x_j)\|} = \|\sum_{i=1}^{m} c_i \frac{\phi(x_i)}{\|\phi(x_i)\|}\|^2 \geq 0.$$

□

The gaussian kernel is the normalized kernel of the exponential one, given by $K'_\sigma(x,y) = e^{x\cdot y/\sigma^2}$ and this one can be developed as

$$K'_\sigma(x,y) = \sum_{n\in\mathbb{N}} \frac{(x\cdot y)^n}{\sigma^2 n!}$$

for all $x, y \in \mathbb{R}^m$ as the radius of convergence of exp is infinite. Thanks to the closure properties of PDS kernels, it is sufficient to show that the kernel $(x,y) \mapsto x \cdot y$ is PDS. This is immediate as the inner product in $\mathbb{R}^m$ is positive definite.

The gaussian kernel has the advantage to be easily computed, which allows to get the projection of an out-of-sample point in $\mathcal{O}(n)$. Other methods have been proposed to generalize projections obtained with manifold learning methods, such as Isomap or Eigenmap. The idea is to use a kernel consistent with the construction of the projection : the k-th coordinate of an out-of-sample point is then given by

$$y_k(x) = \frac{1}{\lambda_k}\sum_{i=1}^{n} v_{ki} K(x,x_i) \tag{11}$$

with K verifying $K(x_i,x_j) = M_{ij}$, where M is the matrix whose eigenvectors v_k has been taken to project the x_i and with λ_k the corresponding eigenvalues. We give below the kernels proposed in [2] to extend Isomap and Eigenmap :

- **Isomap** :

$$K(x,y) = -\frac{1}{2}(K'(x,y) - \mathbb{E}_{x'}[K'(x',y)] - \mathbb{E}_{y'}[K'(x,y')] + \mathbb{E}_{x',y'}[K'(x',y')],$$

where $K'(x,y) \approx d_M(x,y)$ is the approximation of the distance on the manifold that we estimate with Dijkstra algorithm using only the x_i as intermediate points and where the expectations are taken on the initial dataset $\{x_i\}$.

The projection (11) of a new point can be written as

$$\frac{1}{2\sqrt{\lambda_k}} \sum_{i=1}^{n} v_{ki}(\mathbb{E}_{x'}[K'(x',x)] - K'(x,x_i)).$$

- **Eigenmap** :

$$K(x,y) = \frac{K'(x,y)}{\sqrt{K'(x,x)K'(y,y)}},$$

with $K'(x,y) = e^{-\|x-y\|^2/4t\sigma^2}$.

Conclusion

We have seen three methods in manifold learning used to project datasets in low dimension while preserving their geometric structure. Two of them, Isomap and PTU, are based on geodesic distances while the third, Eigenmap, use the properties of the Laplacian on the manifold. To generalize the projection found with these methods, we have have seen the kernel regression method, which takes advantage of the kernel trick. With these tools, we can build a non-linear reduce order model for Burger equation.

Results presented in Figure 12 show that this non-linear model succeed to reproduce almost perfectly the solution computed in high dimension, regardless of the reduction method used. In both cases, we have chosen the laplacian kernels. To be efficient, these models should be built with α and γ small, as one can see on Figure 13 and Figure 14.

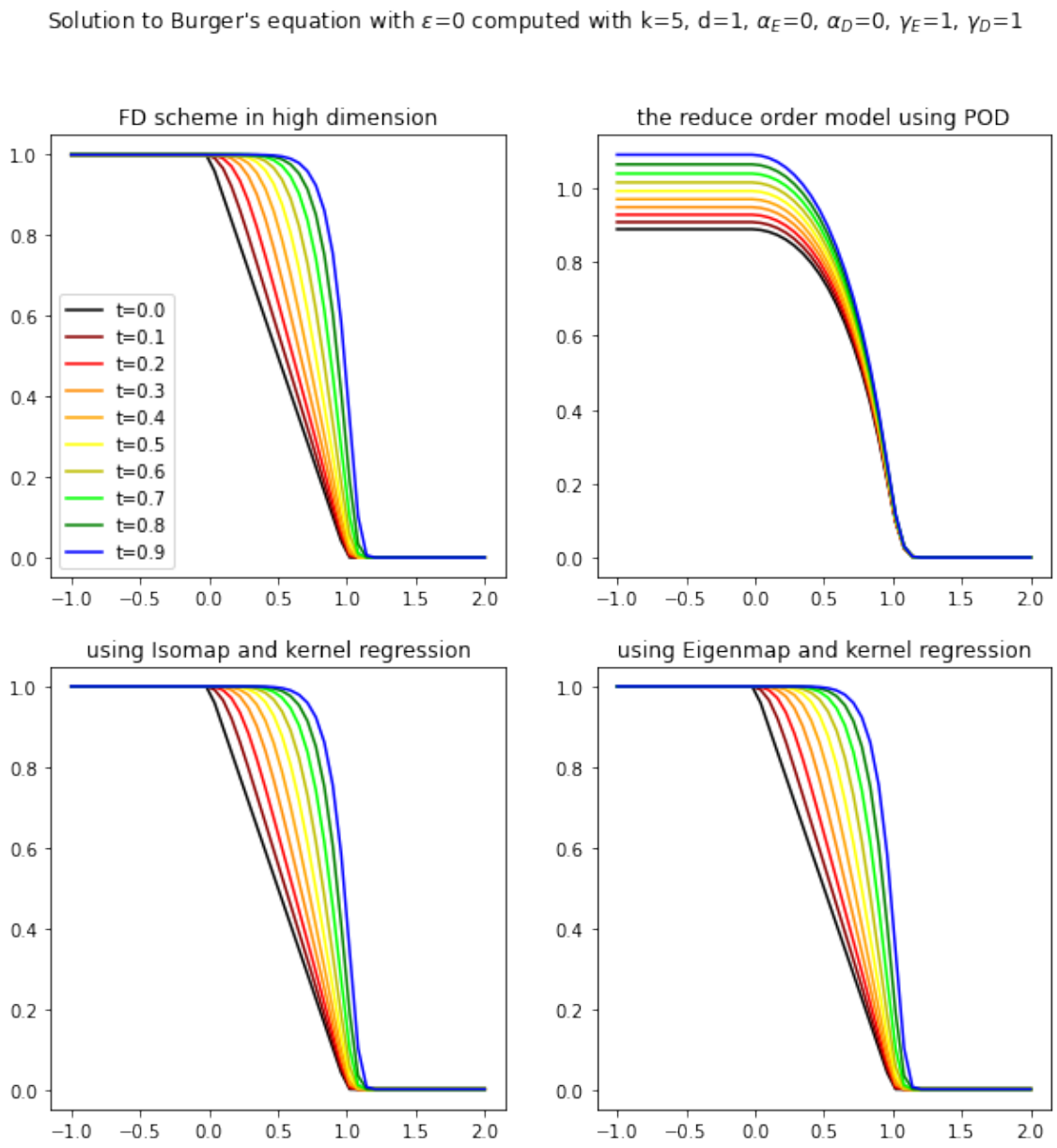

Figure 12: Numerical solution of Burger's equation with $\epsilon = 0$. Samples were computed using Lax-Friedrichs scheme with $n = 1000$ points in time on the interval $[0, 1]$ and $m = 50$ points in space. Parameters for non-linear models are $k = 5$, $d = 1$, $\alpha = 0$ and $\gamma = 0.1$ (we have taken the same parameters values for encoding and decodinng operators). The solution computed with the non-linear models are closer to the solution computed in high dimension with Lax-Friedrichs schema than the one give nby the POD.

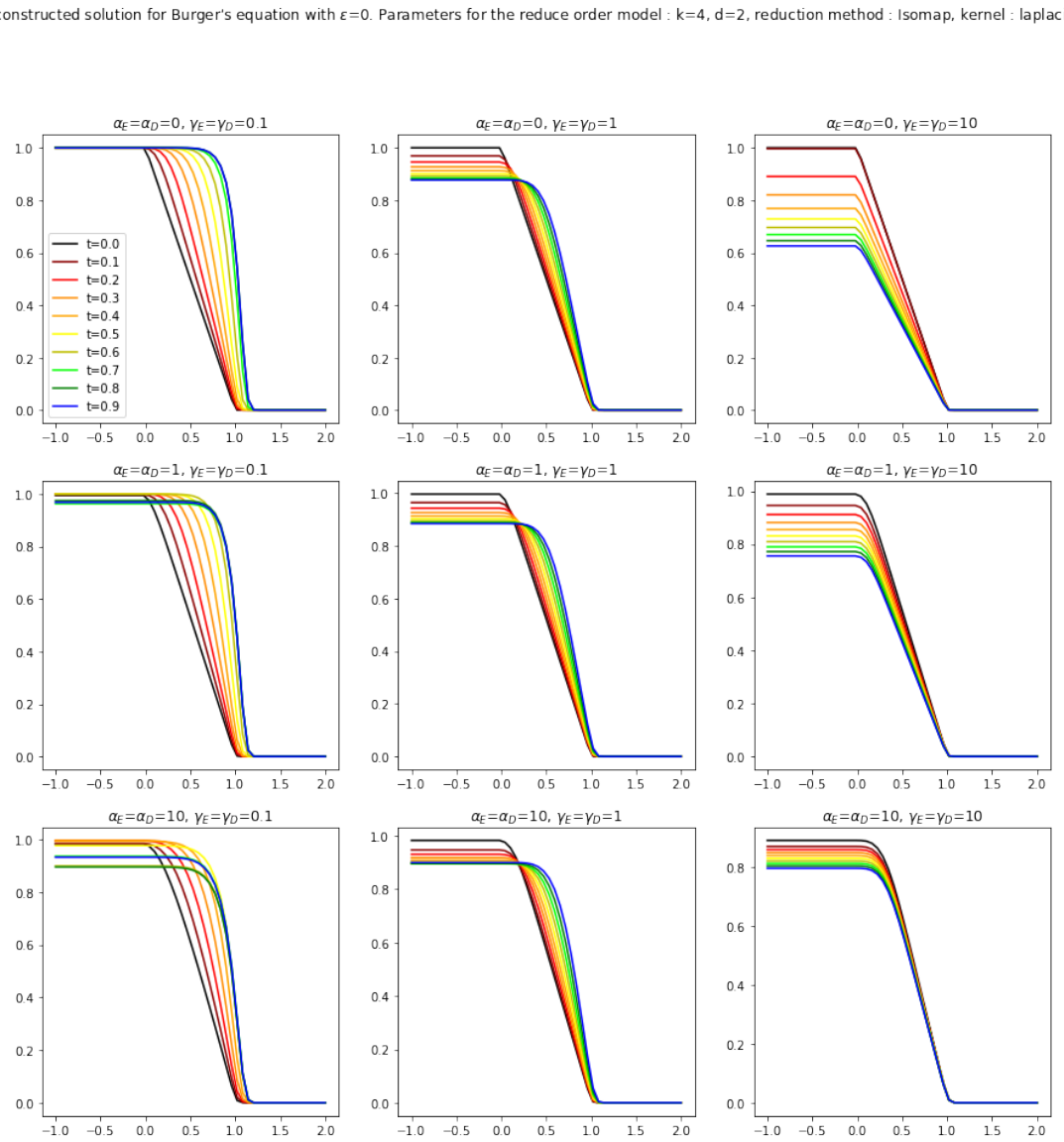

Figure 13: Numerical solution of Burger's equation with $\epsilon = 0$. Samples were computed using Lax-Friedrichs scheme with $n = 1000$ points in time on the interval $[0, 1]$ and $m = 50$ points in space. Parameters for the Isomap reduction are $k = 4$, $d = 2$. We have tested threevalues for the kernel regression parameters, α and $\gamma =$ (we have taken the same parameters values for encoding and decodinng operators). We see that the reduce order model is better when these two parameters are small.

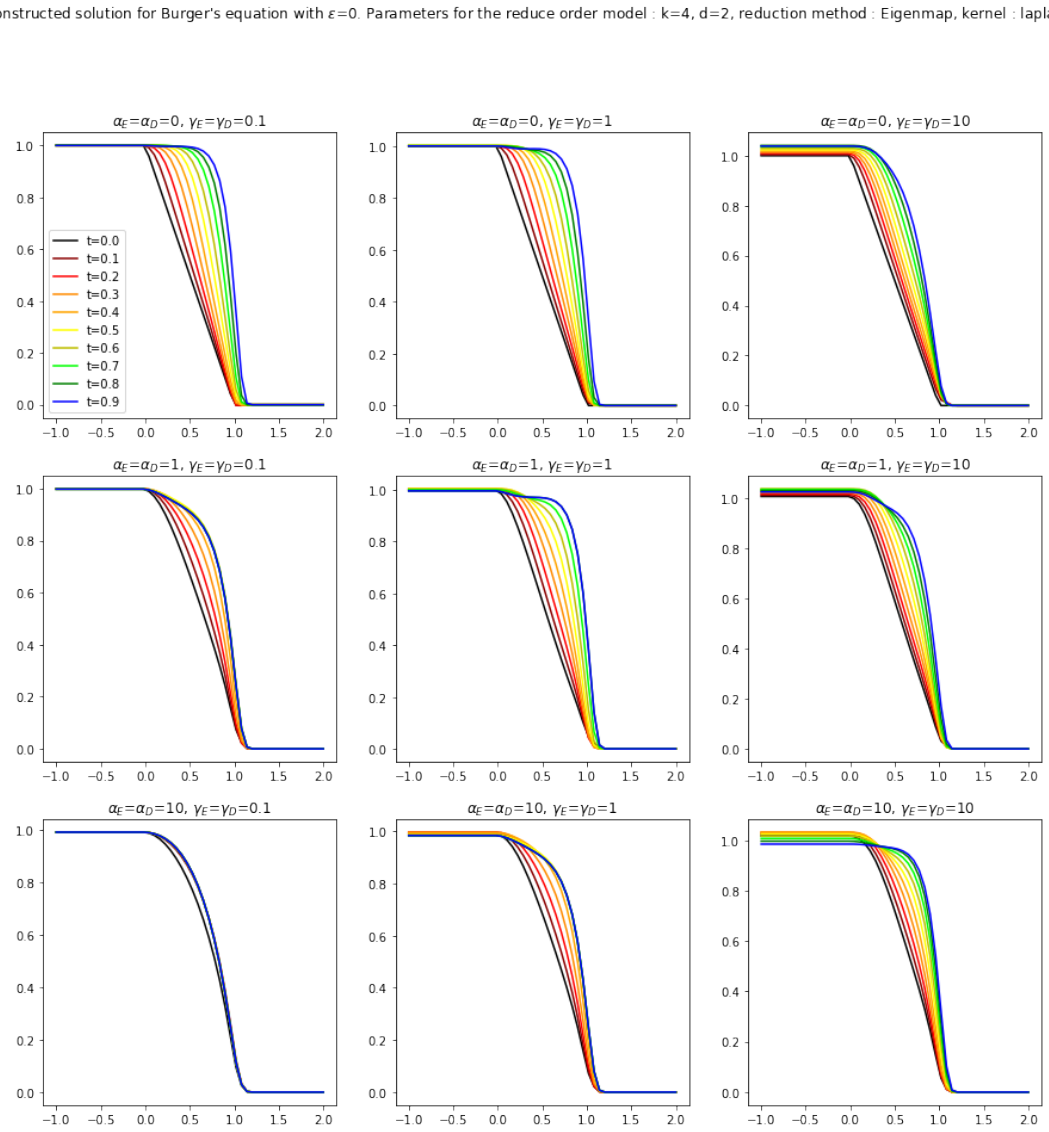

Figure 14: Numerical solution of Burger's equation with $\epsilon = 0$. Samples were computed using Lax-Friedrichs scheme with $n = 1000$ points in time on the interval $[0, 1]$ and $m = 50$ points in space. Parameters for the Eigenmap reduction are $k = 4$, $d = 2$. We have tested threevalues for the kernel regression parameters, α and γ = (we have taken the same parameters values for encoding and decodinng operators). We see that the reduce order model is better when these two parameters are small.

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[2] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel pca. Letter communicated by Sam Roweis, 2004.

[3] M. Bernstein, V. de Silva, Langford J., and Tenenbaum J. Graph approximations to geodesics on embedded manifolds, 2000.

[4] B. Budninskiy, G. Yin, L. Feng, Y. Tong, and M. Desbrun. Parallel transport unfolding: A connection-based manifold learning approach. Preprint at https://arxiv.org/abs/1806.09039v2, 2018.

[5] Y. Canzani. *Analysis on manifold via the Laplacian, lecture notes, Harvard University*, 2013.

[6] R.R. Coifman, I.G. Kevrekidis, S. Lafon, M. Maggioni, and B. Nadler. Diffusion maps, reduction coordinates and low dimensional representation of stochastic systems. *Multiscale Model Simulation*, 7(2):842–864, 2008.

[7] A. Cornuéjols and L. Miclet. *Apprentissage artificiel*, chapter Méthodes à noyaux. Eyrolles, 2010.

[8] M. Hein, J.Y. Audibert, and U. von Luxbourg. Graph laplacians and their convergence on random neighborhood graphs. *Journal of Machine Learning Research*, 8:1325–1368, 2007.

[9] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Fundations of machine learning*, chapter Kernel Methods. The MIT Press, 2018.

[10] S. Shan and I. Daubechies. Diffusion maps : Using the semigroup property for parameter tuning. Preprint at https://arxiv.org/abs/2203.02867v1, 2022.

[11] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[12] J. Wang. *Geometric Structure of High-Dimensional Data and Dimensionality Reduction*, chapter 8, 12, 14, pages 151–180, 203–220, 235–248. Springer, 2012.

Appendices

Here we give codes we have written to test the reduction methods. Isomap and Eigenmap are also implemented in the library sklearn of Python.

A Graphs constructions

A.1 With k-nearest neighbors

```
def data_neighborhood_k(X,k) :
    n = np.shape(X)[1]
    D2 = np.zeros((n,n))
    DV = np.inf*np.ones((n,n))

    for i in range(n) :
        DV[i,i] = 0
        for j in range(i+1,n) :
            D2[i,j] = np.linalg.norm(X[:,i]-X[:,j],2)
            D2[j,i] = D2[i,j]

    for i in range(n) :
        mask = np.argpartition(D2[i,:],k)[1:k+1]
        DV[i,mask] = D2[i,mask]

    DV = np.fmin(DV, DV.T)

    return DV
```

A.2 With δ-neighborhoods

```
def data_neighborhood_eps(X,eps) :
    n = np.shape(X)[1]
    D2 = np.zeros((n,n))

    for i in range(n) :
        for j in range(i+1,n) :
            d = np.linalg.norm(X[:,i]-X[:,j],2)
            if (d < eps) :
                D2[i,j] = d
                D2[j,i] = d
            else :
                D2[i,j] = np.inf
                D2[j,i] = np.inf

    return D2
```

B Isomap

B.1 Dijkstra's algorithm

This is an implementation in Python of the algorithm described in [12].

```
def symdijkstra(D) :
```

```
    n = np.shape(D)[1]
    DG = np.zeros((n,n))
    DG[:,:] = D[:,:]

    for i in range(n) :
        notOut = list(range(n))

        while len(notOut) > 0 :
            j = notOut[ np.argmin(DG[i,notOut]) ]
            notOut.remove(j)
            DG[i,:] = np.fmin(DG[i,j] + D[j,:], DG[i,:])

    return DG
```

B.2 Isomap algorithm

Here is an implementation in Python of the method proposed in [11].

```
def drisomap(X, ndims, mode='k_nearest', k=1, eps=1) :

    #Part 1 : construct data graph
    if mode == 'epsilon' :
        D = np.sqrt(data_neighborhood_eps(X,eps))
    elif mode == 'k_nearest' :
        D = np.sqrt(data_neighborhood_k(X,k))
    else :
        print('invalid mode')
        return None

    #Part 2 : compute graph distance
    N = np.shape(X)[1]
    DG = symdijkstra(D)

    #Part 3 : generate Isomap kernel
    DG = DG**2
    SUM = np.sum(DG, axis=0)
    GC = -0.5*(DG-(2./N)*SUM-(1./N**2)*np.sum(SUM))

    #Part 4 : kernel decomposition
    val, Y = np.linalg.eig(GC)
    Y *= np.sqrt(val)

    if ndims > N :
        print('too large value for ndims : should be inferior to %d' % N)
        return None
    else :
        return Y[:,np.argsort(-val)[:ndims]]
```

C Parallel Transport Unfolding

This is an implementation in Python of the algorithm given in [4].

C.1 Dijkstra's algorithm modified

```
def ParallelTransportDijkstra(S, G, T) :
    n = S.shape[0]
    D = S.shape[1]
    d = T.shape[2]

    R = np.empty((n,d,d))
    Dgeo = np.empty((n,n))
    D = np.empty((n,n))
    D[:,:] = G[:,:]

    for i in range(n) :
        R[i,:,:] = np.eye(d)
        v = np.zeros((n,d))
        pred = np.zeros((n), dtype=int)
        voisins = np.where(G[i,:]!=np.inf)[1]
        pred[voisins] = i
        notOut = list(range(n))
        q = 0

        while len(notOut) > 0 :
            r = notOut[ np.argmin(D[i,notOut]) ]
            notOut.remove(r)
            q = pred[r]
            U, L, V = np.linalg.svd(T[q,:,:].T@T[r,:,:])
            R[r,:,:] = R[q,:,:]@U@V
            v[r,:] = v[q,:] + R[q,:,:]@T[q,:,:].T@(S[r,:]-S[q,:])
            Dgeo[i,r] = np.linalg.norm(v[r,:])
            voisins = np.where(G[r,:]!=np.inf)[1].tolist()

            for j in voisins :
                dist = D[i,r] + G[r,j]
                if dist < D[i,j] :
                    D[i,j] = dist
                    pred[j] = r

    Dgeo[:,:] = (Dgeo[:,:]+Dgeo.T[:,:])/2
    return D, Dgeo, R
```

C.2 Tangent spaces

```
def Tangent(S, d, K) :
    N = S.shape[0]
    Dim = S.shape[1]
    D = sklearn.neighbors.kneighbors_graph(S, K, mode='connectivity',
                                           n_jobs=-1).todense()
    T = np.zeros((N,Dim,d))

    for i in range(N) :
        vois = np.where(D[i,:]==1)[1]
        M = S[vois,:]
        M -= S[i,:]
        U, L, V = np.linalg.svd(M)
        T[i,:,:] = V.T[:,:d]
```

```
    return T
```

C.3 PTU algorithm

```
def ParallelUnfolding(S, ndims, k, d, K) :
    #Part 1 : proximity graph
    N = S.shape[0]
    D = sklearn.neighbors.kneighbors_graph(S, n_neighbors=k,
                        mode='distance', n_jobs=-1).todense()

    #Part 2 : tangent frames
    T = Tangent(S, d, K)

    #Part 3 : distances using parallel transport
    Dg, Dgeo, R = ParallelTransportDijkstra(S, D, T)

    #Part 4 : generate PTU kernel
    H = np.eye(N)-np.ones((N,N))/N
    Dgeo = Dgeo**2
    G = -0.5*(H@Dgeo@H)

    #Part 5 : kernel decomposition
    val, Y = np.linalg.eig(G)

    #Part 6 : choose coordinates
    Y = Y*np.sqrt(val)
    return Y[:,np.argsort(-val)[:ndims]], R
```

D Eigenmap

D.1 Weighted matrix

```
def weight_matrix(D, t) :
    if t < np.inf :
        return np.exp(-D**2/4/t)
    else :
        S = np.zeros_like(D)
        S[d!=np.inf] = 1
        return S
```

D.2 Eigenmap algorithm

A Python implementation of the method proposed in [1].

```
def eigenmap(X, ndims, mode='k_nearest', vois=1, t=np.inf) :

    #Step 1 : adjacency graph
    if mode == 'epsilon' :
        G = np.sqrt(data_neighborhood_eps(X, vois))
    elif mode == 'k_nearest' :
        G = np.sqrt(data_neighborhood_k(X, vois))
    else :
        print('invalid mode')
        return None
```

```
    #Step 2 : choosing the weight
    n = np.shape(G)[0]
    W = weight_matrix(G,t)
    D = np.sum(W, axis=1)[:,np.newaxis]
    S = np.eye(n) - W/D

    #Step 3 : eigen decomposition
    val, Y = np.linalg.eig(S)

    #Step 4 : return first eigne vectors
    index = np.argsort(val)
    return Y[:,index[1:ndims+1]]
```

E Kernel regression

E.1 Kernel matrix

```
def Kexp(x,y, sigma) :
    return np.exp(-np.linalg.norm(x-y)**2/(2*sigma**2))/(sigma*np.sqrt(2*np.pi))

def kernel_matrix(X, sigma) :
    n = X.shape[1]
    K = np.zeros((n,n))
    for i in range(n) :
        for j in range(n) :
            K[i,j] = Kexp(X[:,i],X[:,j],sigma)
    return K
```

E.2 Regression operator

```
def alpha_opt(X, Y, K, lamb) :
    n = K.shape[0]
    return Y@K.T@np.linalg.inv(K@K.T+lamb*np.eye(n))

def G(x, X, K, alpha, sigma) :
    n = X.shape[1]
    Kx = np.zeros((n))
    for i in range(n) :
        Kx[i] = Kexp(X[:,i],x, sigma)
    return alpha@Kx

def JacG(x, X, K, alpha, sigma) :
    m,n = X.shape
    DK = np.zeros((n,m))
    for i in range(n) :
        DK[i,:] = Kexp(X[:,i],x,sigma)*(X[:,i]-x).T
    return alpha@DK
```