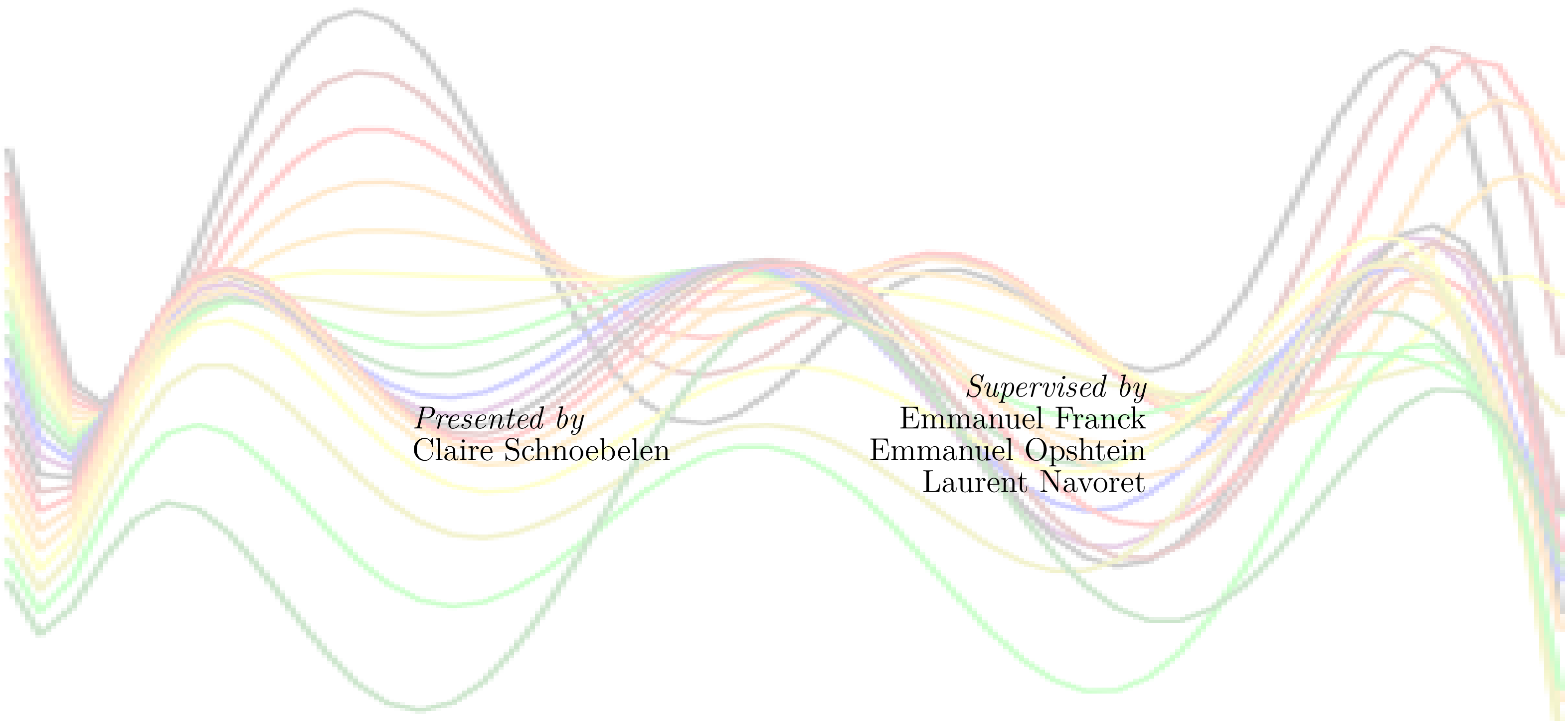
Reduced order models for partial differential equations

Internship report

Master 1 Calcul Scientifique et Mathématiques de l'Information

August 25, 2022



Presented by
Claire Schnoebelen

Supervised by
Emmanuel Franck
Emmanuel Opshtein
Laurent Navoret

Contents

Introduction

INRIA (for Institut National de Recherche en Informatique et Automatique) is a public establishment created in 1967 within the framework of the "plan calcul", a governmental plan intended to develop French knowledge in the field of digital technology and to ensure the digital sovereignty of the country. Today, it has 200 teams spread over 10 research centers, bringing together a total of 3,900 researchers and engineers in mathematics and computer science [9]. The institute works by "équipes-projets", groups of about twenty people working on the same project and for the most part in collaboration with companies [9].

The Nancy research centre was founded in 1986 and today has 20 teams bringing together over 400 people. It has a branch at the University of Strasbourg, where researchers from the TONUS team (for TOkamaks and NUmerical Simulations) work, including my supervisor, Mr. Emmanuel Franck.

I did my internship under the direction of Mr Franck, Mr Opshtein and Mr Navoret between 6 June and 31 July 2022. The aim of the intership was to understand and code (with Python) reduced order models. Given a partial differential equation, the objective of reduced order models is to find a family of functions which alone can explain a large part of the behaviour of the equation solutions. The interest of reduced order modelling is that it allows a faster computation of solutions at any time and for any value of the equation parameters in the interval considered during reduction.

As we saw during last semester project, Proper Orthogonal Decomposition (POD) gives good results for linear equations such as Burgers' one with high viscosity but fail for non-linear one. The aim of last semester project was to study three non-linear reduction methods in the field of manifold learning, Isomap, Parallel Transport Unfolding (PTU) and Eigenmap. Their applications in reduced order modelling in the case of Burgers' equation with no viscosity was the first task of my internship.

The second one was about reduced order modelling in the particular case of linear Hamiltonian systems, where the symplectic structure allows the construction of more robust reduced order models. In short, it consisted of understanding, coding an testing linear reduced order model for Hamiltonian systems. This part required to read and understand theoretical symplectic geometry, particularly about differential forms, symplectic manifolds and generating functions. For that, I mainly used the references [2], [11] and [10]. I also had to read and understand articles dealing with symplectic schemes [7] and symplectic reduced order model ([1], [12] and [13]). To test the models I coded, I had to use three systems modelling a piano string vibration, given in [6]. All these references were given to me by my supervisors, except for [10] and [7], which I found by searching for additional information on my own. Methods and test systems were also given to me.

Among the three problems on which I had to test the reduced order models, two were non-linear. We expected the linear reduced order modelling to fail - and indeed it did. The last objective was then to think about ways to improve those reduced order model in the case of non-linear system.

1 Non-linear reduced order model for Burgers' equation

In that part, we begin by briefly summarize results of the last semester project concerning reduced order-model, non-linear projection and kernel regression. Then, we present results we obtain when we apply those methods to non-linear Burgers' equation.

1.1 Reduced order model

1.1.1 General case

Continuous problem

In what follows, we consider a partial differential equation depending on a parameter $g \in J \subset \mathbb{R}$.

$$\left\{ \begin{array}{cc} \partial_t u = \mathcal{F}_g(u, \partial_z u, \partial_{zz} u, ...) & \forall z \in \Omega \quad \forall t \in I, \\ u(z,0) = u_0(z) & \forall x \in \Omega. \end{array} \right. \tag{1.1}$$

Denote by $\mathcal{M}$ a functional space where lies the solution of (1.1), $\mathcal{C}^p$, $\mathcal{L}^p$ or $\mathcal{H}^p$ for instance. We make the hypothesis that the long-term evolution of the system modelled by this equation depends only on few variables. It means that the solution of (1.1) at any t in I, which is a subset of an infinite dimensional space, can be described using a finite -and low- number of variables, say k.

More precisely, we assume the existence of a k-dimensional manifold $\mathcal{M}$ and of a coordinate map $\mathcal{D} : \mathbb{R}^k \to M$ such that

$$u(t,g) = \mathcal{D}(v(t,g)), \tag{1.2}$$

where $u(t,g)$ is the solution of (1.1) at time t and $v(t,g) \in \mathbb{R}^k$ its coordinates on M.

Discrete problem

After spacial discretisation, the problem (1.1) becomes

$$\left\{ \begin{array}{cc} \frac{d}{dt} Z(t,g) = F_g(Z(t,g)) & \forall t \in I, \\ Z(0) = Z^0 & \forall x \in \Omega. \end{array} \right. \tag{1.3}$$

with $Z \in \mathbb{R}^n$ and $F : \mathbb{R}^n \to \mathbb{R}$ a function that depends on $\mathcal{F}$ and on the scheme we have chosen for space discretisation.

In the discrete space, the assumption that the system (1.1) only depends on k variables leads to the existence of a manifold M embedded in $\mathbb{R}^n$ on which lies solutions $Z(t,g)$ of (1.3).

Let $D : \mathbb{R}^k \to M \subset \mathbb{R}^n$ a discrete coordinate function and $X \in \mathbb{R}^k$ be the coordinates of $Z(t,g)$. The discrete formultion of (1.2) is then given by

$$Z(t,g) = D(X(t,g)). \tag{1.4}$$

Galerkin projection and discrete reduced equation

Applying the chain rule on the previous equation gives

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

We would like to express the discrete system time evolution with a differential equation in $\mathbb{R}^k$ of the form (1.3). If $\mathcal{J}_D$ were everywhere invertible, then it would suffice to multiply the identity (1.2) by $\mathcal{J}_D^{-1}(v(t,g))$ but this can not be true as $\mathcal{J}_D$ is not a square matrix. We will solve this problem by using Galerkin projection.

Let us consider $f : x \in \mathbb{R}^k \mapsto \frac{1}{2}\|z - Jx\|^2$ with $J \in \mathcal{M}_{n,k}(\mathbb{R})$. We have that $f(x+h) = f(x) + \langle h, J^T(Jx-z)\rangle + \langle h, J^TJh\rangle$, so f is twice differentiable with $\nabla f(x) = J^T(Jx-z)$ and $\nabla^2 f(x) = J^TJ$. Let us consider the singular value decomposition of $J = U\Sigma V$, with $U \in \mathcal{O}(n), V \in \mathcal{O}(k)$ and $\Sigma \in \mathcal{M}_{n,k}(\mathbb{R})$ such that

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

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

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

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

$$\frac{d}{dt}X(t) = \mathcal{J}(D)(X(t))^+F(D(X(t))). \tag{1.5}$$

Reduced order modelling consists therefore in finding a *decoding* operator $D : \mathbb{R}^k \to \mathbb{R}^n$ such that (1.4) is verified.

1.1.2 Proper Orthogonal Decomposition

We want to build a numerical model to solve the reduced equation. Assume first that E and D are linear. In that case, (1.5) becomes

$$\frac{d}{dt}X = A^+\tilde{F}(AX), \tag{1.6}$$

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

To build our model, we first consider a model to solve the original equation (1.1). We discretize the spacial domain Ω into n points x_i, the time interval into m points t^l and we note $\hat{Z}^l \in \mathbb{R}^n$ the solution at a time t^l and at the point x_i. In what follows, we use Euler scheme to approximate the time derivative.

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

$$\frac{Z_i^{l+1} - Z_i^l}{\Delta t} = \hat{F}(Z^l).$$

Then, the corresponding scheme in low dimension is

$$\frac{X_i^{l+1} - X_i^l}{\Delta t} = A^+\hat{F}(AX^l), \tag{1.7}$$

where $X^l \in \mathbb{R}^k$ denotes the solution of (1.6) at time $t = l\Delta t$. The initial condition X^0 is given by $A^+ Z_0(x)$.

It then remains to defind A. In the linear case, the fact that the solution of (1.1) can be expressed in k variables means that

$$X(x,t) = \sum_{i=1}^{d} a_i(t)\psi_i(x), \tag{1.8}$$

for ψ_i some functions in $\mathcal{M}$ called *modes* of the equation.

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 (1.8) at any time t. The operator A that will be used in (1.7) 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).

1.1.3 Burgers' equation

Here, we consider Burgers' equation

$$\partial_t Z(z,t) = \tfrac{\partial_z Z^2}{2}(z,t) - \epsilon \partial_{zz} Z(z,t), \qquad \forall z \in J \subset \mathbb{R}, \quad \forall t \in I \subset \mathbb{R}, \tag{1.9}$$

where ϵ denotes the viscosity

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

$$\hat{F}(Z^l) = \frac{\mathcal{F}_{i+\frac{1}{2}} - \mathcal{F}_{i-\frac{1}{2}}}{2} - \epsilon \frac{Z_{i+1}^l - 2Z_i^l + Z_{i-1}^l}{\Delta_z^2},$$

with

$$\mathcal{F}_{i+\frac{1}{2}} = \frac{\frac{(Z_{i+1}^l)^2}{2} + \frac{(Z_i^l)^2}{2}}{2} - c_{i+\frac{1}{2}} \frac{Z_{i+1}^l - Z_i^l}{2},$$

where $c_{i+\frac{1}{2}} = \max\{|Z_{i+1}^l|, |Z_i^l|\}$.

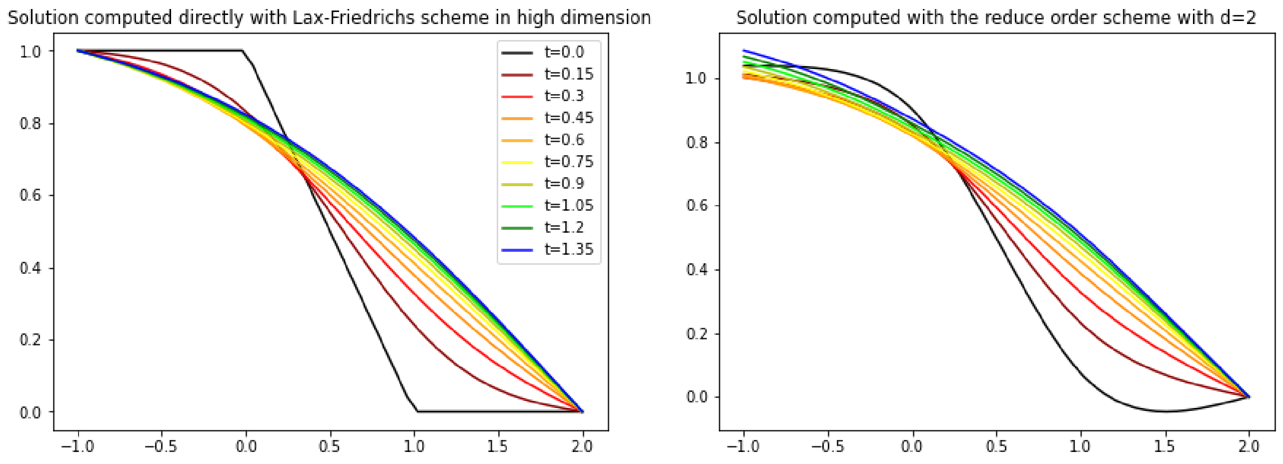


Figure 1.1: Numerical solution of Burgers' equation with $\epsilon = 1$.The reduced order model gives results very similar to what we obtain with Lax-Friedrichs model in low dimension.

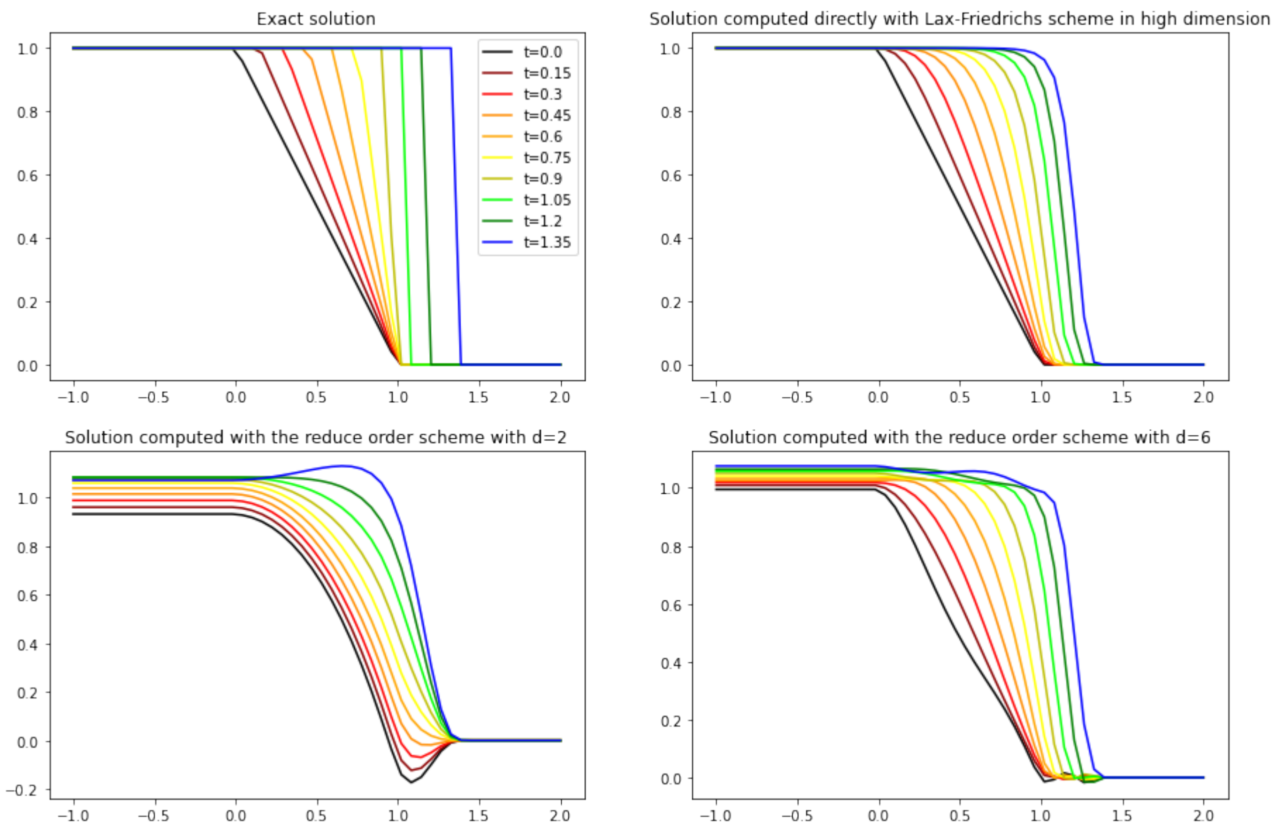


Figure 1.2: Numerical solution of Burgers' equation with $\epsilon = 0$. The solution obtained with the reduced order model with $k = 2$ gives unaccurate results and with $k = 6$, it seems "noisy" for highest values of t next to singular points of the exact solution.

When ϵ is large enough, that is when there is enough diffusion in the equation to regularize the solution, this gives satisfying results, see Figure 1.1. This is not the case anymore when ϵ is too small, as we see in Figure 1.2. This means that the hypothesis that the solution belongs to a linear subspace does not hold anymore.

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

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

1.2 Non-linear reduction methods

During the last semester project, we studied three non-linear reduction methods, two distance-based methods, Isomap [14] and Parallel Transport Unfolding (PTU) [4], and another based on Laplace-Beltrami eigenvalues, Eigenmap [3].

In what follows, the z^i are points in $\mathbb{R}^n$ supposed to live in a k-dimensional manifold M embedded in $\mathbb{R}^n$. We note Z the dataset composed of the z^i. We want to find the coordinates Y of Z in $\mathbb{R}^k$. Using the r-nearest neighboors or δ-neighborhoods, we build a graph $\mathcal{G}$ whose vertices are the z^i.

1.2.1 Isomap

The idea of Isomap, proposed in [14], is to apply MultiDimentional Scaling (MDS) to a distance matrix that approaches distances in the manifold.

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

Then, $Y = \tilde{\phi}(Z)$ minimizes the loss function :

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

where $S_M = (dist_M(z^i, z^j)^2)_{ij}$ is the matrix of the squared distances in M.

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

In practice, D_M is unknown. To estimate it, we use Dijkstra's algorithm, assuming that if $\mathcal{G}$ meshes M sufficiently well, lengths of shortests paths in $\mathcal{G}$ give good approximations of geodesic distances on M, in other words, for $x, y \in \mathcal{G}$:

$$d_M(x,y) = \inf_{\gamma \in \Gamma} \int_0^1 \|\dot{\gamma}\| dt \approx \sum_{i=1}^{s-1} \|\overrightarrow{z^i z^{i+1}}\|,$$

where $\Gamma = \{\gamma : [0,1] \to M \in \mathcal{C}^1_{pm} \quad | \quad \gamma(0) = x \wedge \gamma(1) = y\}$ and $\{z^i\}_{i=1,\dots,s}$ is the set of points where the shortest path in $\mathcal{G}$ between x and y passes through.

1.2.2 Parallel Transport Unfolding

Differences between Parallel Transport Unfolding (PTU), proposed in [4], and Isomap occurs when estimating geodesic distances. In [4], authors propose a more accurate way to esimate geodesic distances on M when the manifold is geodesically convex.

The whole point of calculating distances with Dijkstra's algorithm is to take into account the curvature of M. This process also takes into account the curvature resulting from the particular paths used on M. The aim of PTU is to compute distances by reducing the overestimation induced by the curvature of the paths used while retaining the effect of the curvature of the manifold, in order to have a more accurate estimate of the true distances on the manifold.

Roughly speaking, the idea is to sum vectors $\overrightarrow{z^i z^{i+1}}$ instead of summing there norms. As they do not belong to the same tangent space, we need for that to parallel transport them to the tangent space at z. If we note v_i the parallel transported of $\overrightarrow{z^i z^{i+1}}$ into $T_x M$, we take

$$\|\sum_{i=1}^{n-1} v_i\|$$

as an approximation of $d_M(x,y)$.

If M is k-dimensional manifold, its tangent spaces are vector spaces of dimension k. An approximate basis T_i of $T_{z^i}M$ is given by the k eigenvectors associated to the k largest eigenvalues of the matrix whose columns are $\overrightarrow{z^i z^j}$, with $\{z^j\}$ the set of r-nearest neighbors of z^i in $\mathcal{G}$.

The parallel transport between $T_{z^i}M$ and $T_{z^j}M$ is approached by the transformation R in $\mathcal{O}(k)$ which best aligns bases T_i and T_j, in the sense that R minimizes

$$\|T_i - T_j R\|_F,$$

where $\|\cdot\|_F$ is the Frobenius norm.

1.2.3 Eigenmap

If Isomap and PTU give good results, these two methods have a high computational cost. This is due to Dijkstra's algorithm, whose complexity is in $\mathcal{O}(m^2\log(m))$ [4]. Thus, it can be interesting to see if we could find a "good" projection without computing distances in the manifold.

In the Eigenmap method, proposed in [3], 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 1.2.1. *Let $f : M \to \mathbb{R}$ twice differentiable and x, y two close points in M. Then,*

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

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

$$E(f) = \int_M \|\nabla f(x)\|^2 dx. \tag{1.10}$$

To remove a scaling factor and a translation invariance, we impose $\|f\| = 1$ and $\langle f, \mathbb{1}\rangle = 0$.

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 $f : M \to \mathbb{R}^k$, with $k > 1$. The solution is then given by the k first unit eigenfunctions associated to the k smallest positive eigenvalues of $\mathcal{L}$.

Several discrete operators can be chosen to approach $\mathcal{L}$ [8], one of them is $\Delta = Id - D^{-1}W$, with W the weight matrix of $\mathcal{G}$ and D the diagonal matrix whose coefficients are $D_{ii} = d_i = \sum_{j=1}^m w_{ij}$. It can be shown that Δ converges, in a certain sense and at certain conditions, to $\mathcal{L}$ as the number of points z^i increases (see [8] for exact statement and proof).

1.3 Generalization to out-of-sample points

Non-linear projection methods give a low-dimensional representation of a sample, optimal according to several criteria, but contrary to the linear projection method they do not provide a simple way to compute the projection of a new point in the reduced space.

The final step of the reduced order model construction is to build a map $f : \mathbb{R}^n \to \mathbb{R}^k$ which generalizes the projection found on the sample. An efficient way to proceed is to use a kernel regression.

We recall the kernel and positive semi-definite kernel definitions.

Definition 1.3.1.
1. *A map $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is called a kernel.*
2. *A kernel is said positive definite symmetric (PDS) if for all $\{x^1, ..., x^n\} \in \mathbb{R}^n$ the matrix $(K(x^i, x^j))_{ij}$ is symmetric positive semidefinite.*

In a kernel regression over the $\{z^i\}_{i=1,...,m}$, we look for f of the form

$$f_\theta(x) = \sum_{i=1}^m \theta_i K(z^i, x), \tag{1.11}$$

where $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a symmetric PSD kernel and $\theta_i \in \mathbb{R}^k$ are such that

$$\mathcal{L}(\theta) = \sum_{i=1}^{m} |f_\theta(z^i) - x^i|^2$$

is minimal, with x^i the projections of the z^i. The improve the computation of the optimal values of θ, it can sometimes be useful to consider instead the regularized loss

$$\mathcal{L}_\alpha(\theta) = \sum_{i=1}^{m} |f_\theta(z^i) - x^i|^2 + \alpha \sum_{i=1}^{m} \|f_\theta(z^i)\|_2^2.$$

The choice of the kernel is important. We used Gaussian and Laplacian kernels, whose expressions are respectively

$$K_\sigma(x, y) = \exp(\frac{\|x - y\|_2^2}{2\sigma^2})$$

and

$$K_\gamma(x, y) = \exp(\gamma\|x - y\|_1)$$

with σ and γ scale parameters, and obtain very different results.

As shown on Figure 1.3, the decoder built with Laplacian kernel gives a very satisfying generalization for the projection obtained with Isomap applied to a swiss roll : out-of-samples images are well placed on the swiss rolll. On the reverse, the use of a Gaussian kernel on the same test case gives a poor generalization : out-of-samples images are not on the swiss roll. We have never obtained good results with a Gaussian kernel, regardless of the value we took for γ.

The difference between the two results may be explained by the functional space where live functions of the form (1.11) for both kernels. Gaussian kernel produces functions of $\mathcal{C}^\infty(\mathbb{R}^n, \mathbb{R}^k)$ where as Laplacian kernel produces functions of $\mathcal{H}^1(\mathbb{R}^n, \mathbb{R}^k)$. Then, the assumptions we implicitely make on the regularity of f when we choose a Laplacian kernel rather than a Gaussian one are less restrictive.

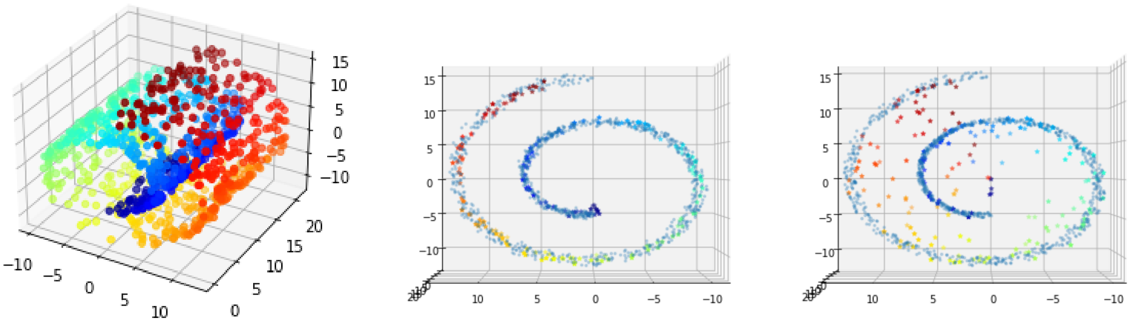


Figure 1.3: Generalized projection built with Laplacian (center) and Gaussian (right) kernels applied to 200 out-of-sample points of the swiss roll. On the left are shown the 2000 samples used to build the projection with Isomap. The Isomap projection was computed from a 11-neighboors graph using sklearn library.

1.4 Application to Burgers' equation

In short, the construction of the reduced order model followed the steps :

1. Computing snapshots with Lax-Friedrichs scheme in high dimension.

2. Find a non-linear projection of these snapshots to a low-dimensional space using Isomap, PTU or Eigenmap.

3. Generalize the previous projection.

4. Solve the equation in the low-dimensional space.

By doing this, we obtain very good results, far better than what we got with POD: results presented in Figure 1.4 show that this non-linear model succeed to reproduce almost perfectly the solution computed in high dimension, regardless of the reduction method used, Isomap or Eigenmap. 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 1.5 and Figure 1.6. This is not surprising as a high value of α leads to a f too far from the projection computed on the samples: the loss $\mathcal{L}$ gives too much importance to the regularization.

The computation using the reduced model is really faster than the computation using Lax-Friedrichs in high dimension (a quasi-instantaneous compared to a few second computation). Projection computation with Isomap and PTU has however a cost of $\mathcal{O}(n^2 \log(n))$ [4] du to singular vlue decomposition and Dijkstra's algorithm. In tests we made, Eigenmap was two or three times faster for the same amount of snapshots.

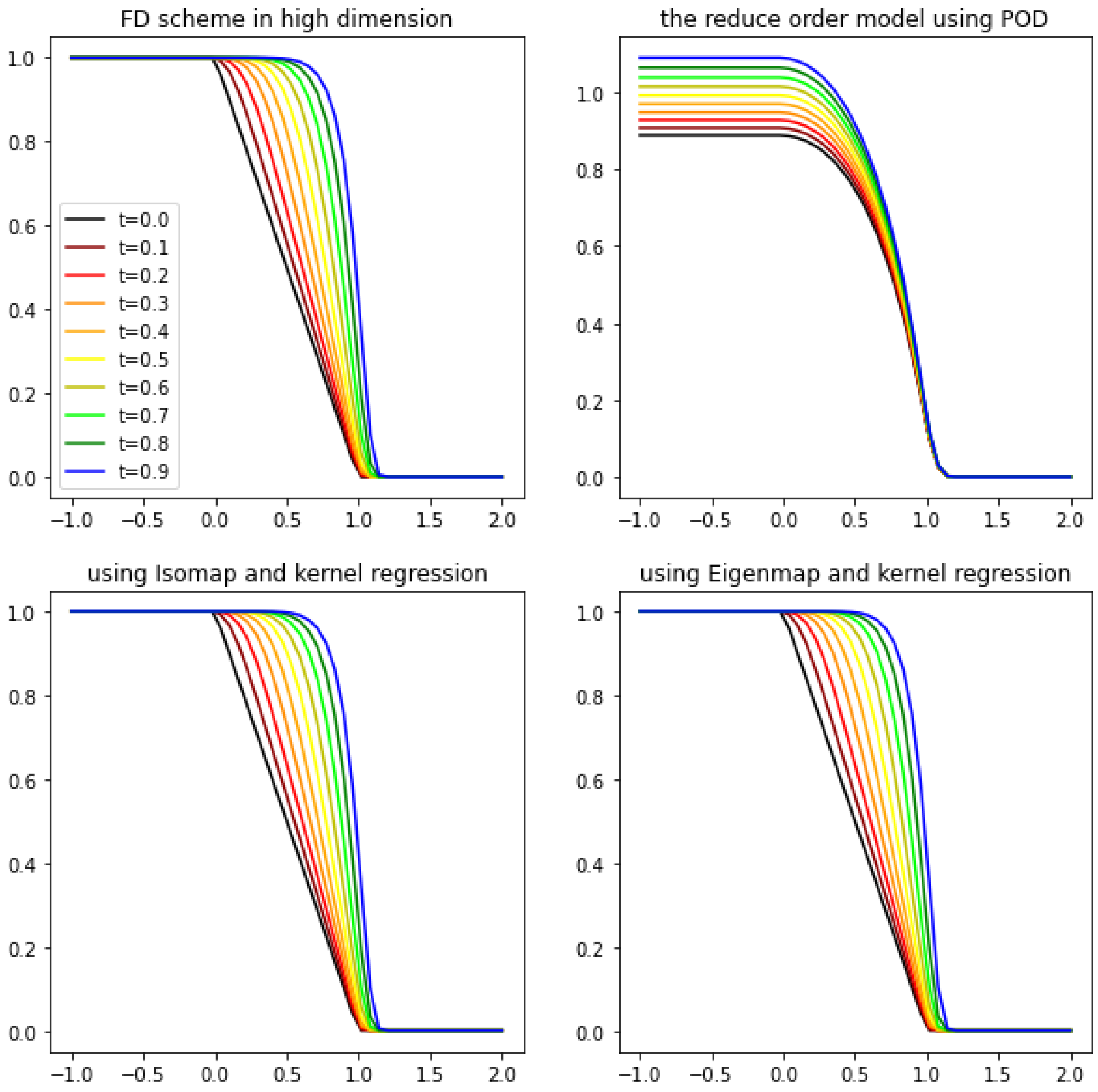


Figure 1.4: Numerical solution of Burgers' equation with $\epsilon = 0$. Samples were computed using Lax-Friedrichs scheme with $m = 1000$ points in time on the interval $[0, 1]$ and $n = 50$ points in space. Parameters for non-linear models are $r = 5$, $k = 1$, $\alpha = 0$ and $\gamma = 0.1$ (we have taken the same parameters values for encoding and decodinng operators). We built $\mathcal{G}$ with the r-nearest neighboors. The solution computed with the non-linear models are closer to the solution computed in high dimension with Lax-Friedrichs schema than the one given by the POD.

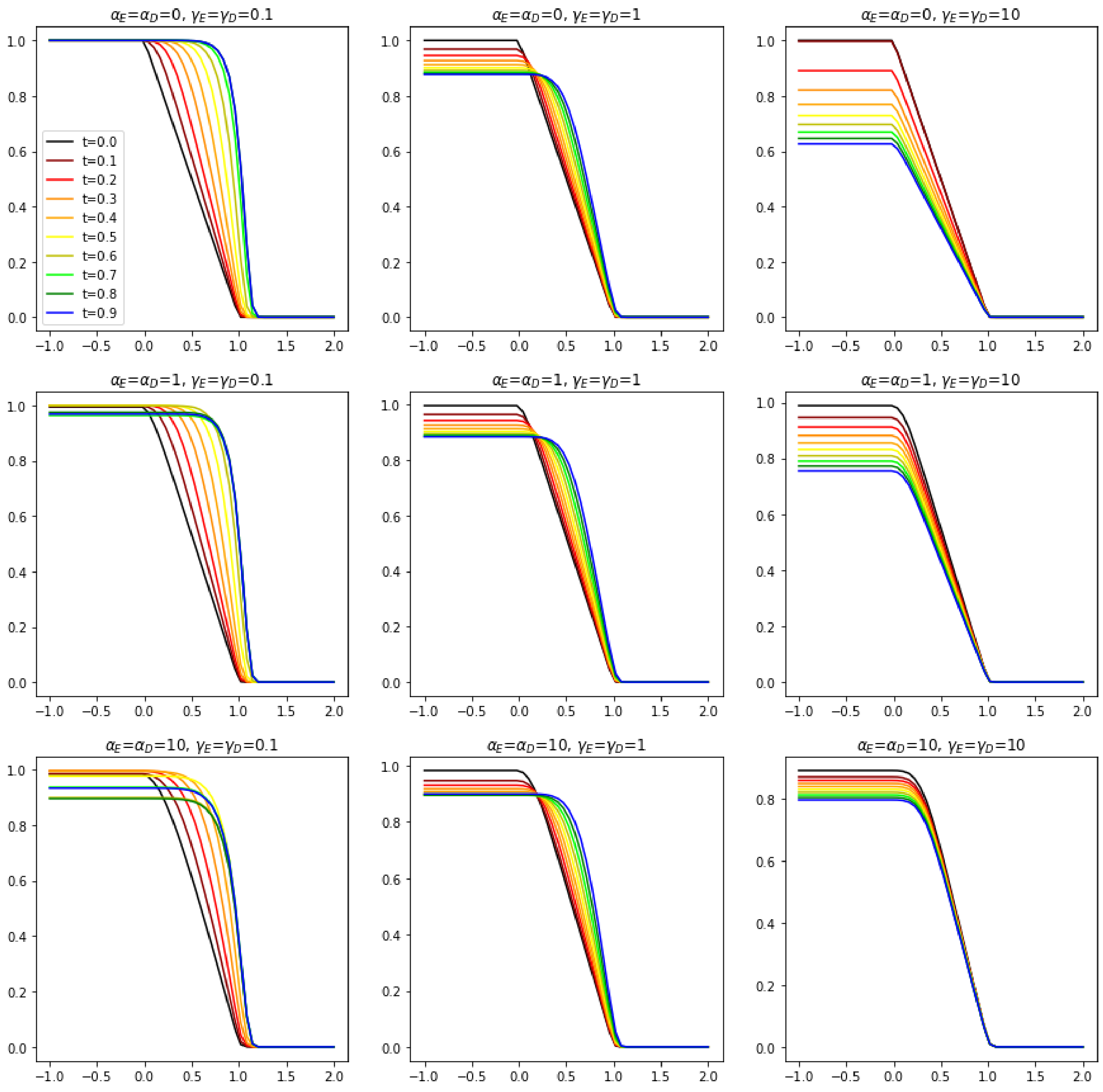


Figure 1.5: Numerical solution of Burgers' equation with $\epsilon = 0$. Samples were computed using Lax-Friedrichs scheme with $m = 1000$ points in time on the interval $[0,1]$ and $n = 50$ points in space. Parameters for the Isomap reduction are $r = 4$, $k = 2$. We have tested threevalues for the kernel regression parameters, α and γ (we have taken the same parameters values for encoding and decodinng operators). We built $\mathcal{G}$ with the r-nearest neighboors. We see that the reduced order model is better when these two parameters are small.

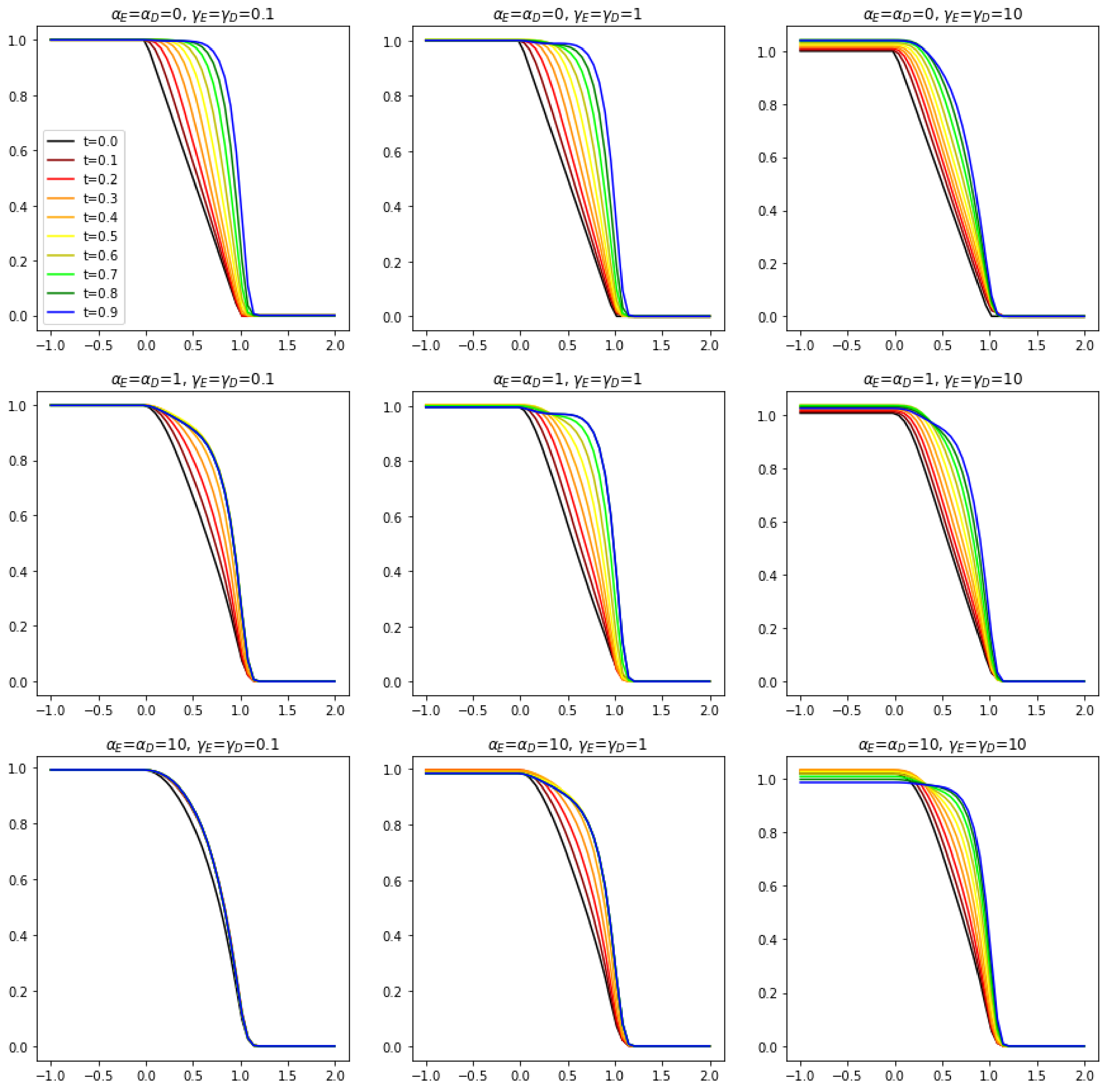


Figure 1.6: Numerical solution of Burgers' equation with $\epsilon = 0$. Samples were computed using Lax-Friedrichs scheme with $m = 1000$ points in time on the interval $[0, 1]$ and $n = 50$ points in space. Parameters for the Eigenmap reduction are $r = 4$, $k = 2$. We have tested threevalues for the kernel regression parameters, α and γ (we have taken the same parameters values for encoding and decodinng operators). We built $\mathcal{G}$ with the r-nearest neighboors. We see that the reduced order model is better when these two parameters are small.

2 Symplectic reduced order model

We have seen that the POD works well on Burgers' equation for large values of ϵ which makes the equation "almost" linear. Nevertheless, the reduction is not always so good, even for linear equations. On Figure 2.1, we show what happens if we use the POD on a one-parameter linear equation modelling a piano string vibration (which will be presented below): even when we take a large value of k, the reduced model fails to capture the shape of the solution.

The system we have chosen has a particular form, this is in fact what we call a Hamiltonian system. The idea we follow from now on is to exploit this structure to build a more robust reduction. We explain below what makes these kinds of systems so particular but we first need to introduce some geometrical concepts.

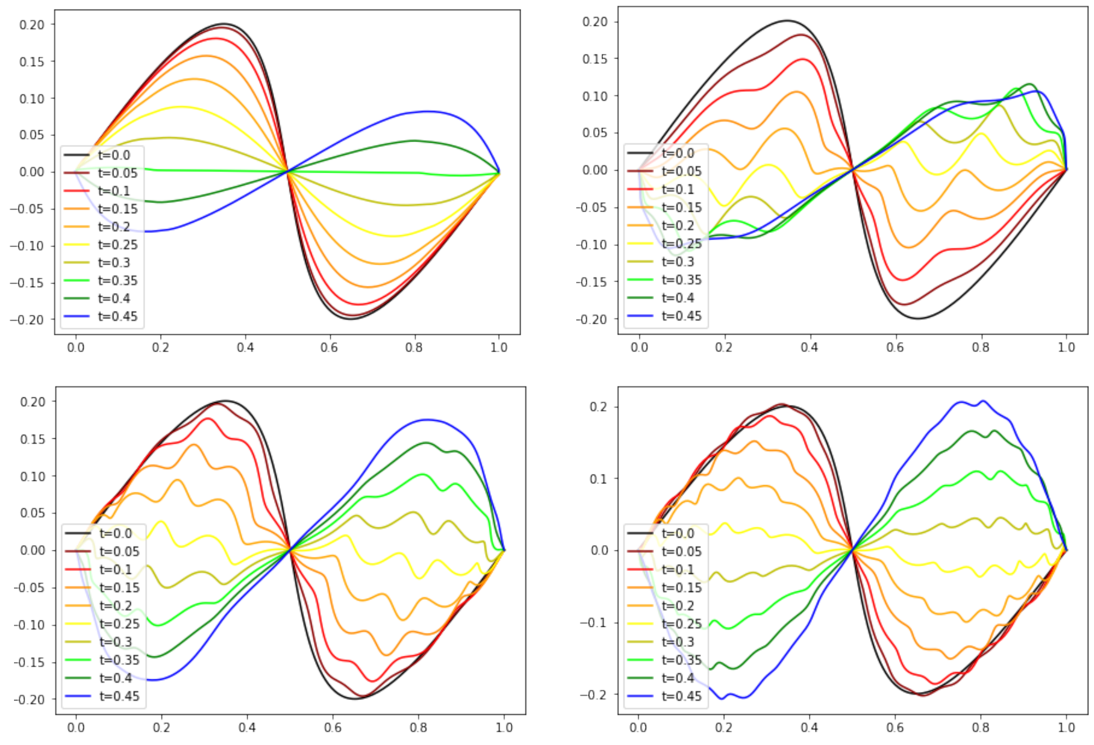


Figure 2.1: Numerical solution of linear piano string equations. The reduction was made using POD with $k = 5$ (top right), 10 (bottom left) and 20 (bottom right).Snapshots were 20 trajectories computed in high dimension with explicit Stormer-Verlet scheme, for values of the equation parameter g taken in $I = [0, 1]$, $n = 200$, $dt = 0.001$ and $m = 500$. Trajectories in low dimension were computed using the reduced model with the same discretisation in time and space and with $g = 0.537$ (this value has been chosen to be different from those used for the projection). Trajectories in high dimension (top left) were also computed with Strörmer-Verlet using the same discretisations.

2.1 Mathematical background

This part is a digest of [2], [11] and [10].

2.1.1 Alternate forms on $\mathbb{R}^n$

Definition 2.1.1. *Let $n, k \in \mathbb{N}$. A k-form ω on $\mathbb{R}^n$ is a k-linear skew-symmetric application of $(\mathbb{R}^n)^k$ in $\mathbb{R}$.*

Remark 2.1.1. Saying that a k-linear form ω on a vector space E is a skew-symmetric means that for all $x_1, ..., x_k$ in E and for all permutation σ,

$$\omega(x_{\sigma(1)}, ..., x_{\sigma(k)}) = \epsilon(\sigma)\omega(x_1, ..., x_k),$$

with $\epsilon(\sigma)$ the signature of σ.

Example 2.1.1.
- The 1-forms on $\mathbb{R}^n$ are the elements of the dual of $\mathbb{R}^n$. For $\{e_i\}_{i=1,...,n}$ a basis of $\mathbb{R}^n$, they can be uniquely written as $\omega = \omega_1 e_1^* + ... + \omega_n e_n^*$, where the forms e_i^* are characterized by $e_i^*(e_j) = \delta_{ij}$ and where the coefficients ω_i are given by $\omega(e_i)$.
- The 2-forms are the bilinear applications verifying $\omega(u, v) = -\omega(v, u)$ for all $u, v \in \mathbb{R}^n$. In $\mathbb{R}^2$, the oriented area $S : (u, v) \to u_1v_2 - u_2v_1$ is a 2-form.

2.1.2 Exterior multiplication

The exterior multiplication gives a way to build a $k + l$-form from a k-form and a l-form.

Definition 2.1.2. *Let ω^k and ω^l be a k-form and a l-form on $\mathbb{R}^n$. The exterior product of ω^k and ω^l is a $(k + l)$-form defined by*

$$\omega^k \wedge \omega^l(u_1, ..., u_{k+l}) = \sum_{\sigma \in S} \epsilon(\sigma)\omega^k(u_{\sigma(1)}, ..., u_{\sigma(k)})\omega^l(u_{\sigma(k+1)}, ..., u_{\sigma(k+l)}),$$

where $S := \{\sigma \in \mathfrak{S}_{k+l} \quad | \quad \sigma(1) < ... < \sigma(k) \quad \wedge \quad \sigma(k + 1) < ... < \sigma(k + l)\}$.

We easily verify that $\omega^k \wedge \omega^l$ is really a $(k+l)$-form. The linearity of $\omega^k \wedge \omega^l$ follows from the linearity of ω^k and ω^l. On another hand, for all permutation μ,

$$\omega^k \wedge \omega^l(u_{\mu(1)}, ..., u_{\mu(k+l)}) = \sum_{\sigma' \in S_\mu} \epsilon(\sigma')\omega^k(u_{\sigma'\circ\mu(1)}, ..., u_{\sigma'\circ\mu(k)})\omega^l(u_{\sigma'\circ(k+1)}, ..., u_{\sigma'\circ\mu(k+l)}),$$

where $S_\mu := \{\sigma' \in \mathfrak{S}_{k+l} \quad | \quad \sigma' \circ \mu \in S\}$. The change of variable $\sigma = \sigma' \circ \mu$ transforms the previous sum into

$$\sum_{\sigma \in S} \epsilon(\sigma \circ \mu^{-1})\omega^k(u_{\sigma(1)}, ..., u_{\sigma(k)})\omega^l(u_{\sigma(k+1)}, ..., u_{\sigma(k+l)}).$$

Noting that $\epsilon(\mu^{-1}) = \epsilon(\mu)$, we find that

$$\omega^k \wedge \omega^l(u_{\mu(1)}, ..., u_{\mu(k+l)}) = \epsilon(\mu)\omega^k \wedge \omega^l(u_1, ..., u_{k+l}).$$

Proposition 2.1.1. *The operation $\wedge$ is distributive, associative and anticommutative.*

Proof. Let ω^k, ω^l and ω^m respectively be m, k and l-forms on $\mathbb{R}^n$.

- Associativity :

Let us start be defining

$$S_\mu^{k_0,...,k_p} := \left\{ \sigma \in \mathfrak{S}_{K_p} \ \middle| \ \begin{cases} \forall r \in [\![0,p-1]\!] & (k_r < i < j \leq k_{r+1}) \implies (\sigma\circ\mu(K_r+i) < \sigma\circ\mu(K_r+j)), \\ \forall i \in [\![0,k_0]\!] & \sigma\circ\mu(i) = \mu(i) \end{cases} \right\},$$

and

$$R_\mu^{k_0,...,k_p} := \left\{ \sigma \in \mathfrak{S}_{K_p} \ \middle| \ \begin{cases} \forall r \in [\![-1,p-2]\!] & (k_r < i < j \leq k_{r+1}) \implies (\sigma\circ\mu(K_r+i) < \sigma\circ\mu(K_r+j)), \\ \forall i \in [\![K_{p-1}+1,K_p]\!] & \sigma\circ\mu(i) = \mu(i) \end{cases} \right\},$$

for all $k_0, ..., k_p \in \mathbb{N}$ and all $\mu \in \mathfrak{S}_{K_p}$, where we have noted $K_r = \sum_{i=0}^p k_i$ and $k_{-1} = 0$.

In particular, $S_{id}^{0,m,k} = R_{id}^{m,k,0}$ is the set of partitions of $[\![0, m+k]\!]$ into two subsets of size m and k whose elements have been arranged in ascending order. More generally, $S_\mu^{0,m,k} = R_\mu^{m,k,0}$ is the set of partitions $[\![0, m+k]\!]$ into two subsets of size m and k whose elements have been arranged in ascending order according to their image by μ. Thus, if μ preserves order, $S_\mu^{0,m,k} = S_{id}^{0,m,k} = R_\mu^{m,k,0} = R_{id}^{m,k,0}$.

Note that for any $\mu \in S_{id}^{0,m,k+l}$ and any $\sigma \in S_\mu^{m,k,l}$, $\sigma \in S_{id}^{0,m,k,l}$. Conversely, any permutation of $S_{id}^{0,m,k,l}$ is uniquely written as a composition of an element of $S_{id}^{0,m,k+l}$ and an element of $S_\mu^{m,k,l}$ (it suffices to see that once given μ, σ is uniquely given).

Similarly, for any $\mu' \in R_{id}^{m+k,l,0}$ and any $\sigma' \in R_\mu^{m,k,l}$, $\sigma'\circ\mu' \in R_{id}^{m,k,l,0} = S_{id}^{0,m,k,l}$. And any element of the last set is uniquely written as a composition of two elements of the first two.

Then,

$$\begin{aligned}
&\omega^m \wedge (\omega^k \wedge \omega^l)(u_1, ..., u_{k+l+m}) \\
=& \sum_{\mu \in S_{id}^{0,m,k+l}} \epsilon(\mu)\omega^m(u_{\mu(1)}, ..., u_{\mu(m)}) \Big(\sum_{\sigma \in S_\mu^{m,k,l}} \epsilon(\sigma)\omega^k(u_{\sigma\circ\mu(m+1)}, ..., u_{\sigma\circ\mu(m+k)})\omega^l(u_{\sigma\circ\mu(m+k+1)}, ..., u_{\sigma\circ\mu(m+k+l)}) \Big) \\
=& \sum_{\mu \in S_{id}^{0,m,k+l}} \sum_{\sigma \in S_\mu^{m,k,l}} \epsilon(\sigma\circ\mu)\omega^m(u_{\sigma\circ\mu(1)}, ..., u_{\sigma\circ\mu(m)})\omega^k(u_{\sigma\circ\mu(m+1)}, ..., u_{\sigma\circ\mu(m+k)})\omega^l(u_{\sigma\circ\mu(m+k+1)}, ..., u_{\sigma\circ\mu(m+k+l)}) \\
=& \sum_{s \in S_{id}^{0,m,k,l}} \epsilon(s)\omega^m(u_{s(1)}, ..., u_{s(m)})\omega^k(u_{s(m+1)}, ..., u_{s(m+k)})\omega^l(u_{s(m+k+1)}, ..., u_{s(m+k+l)}) \\
=& \sum_{\nu \in R_{id}^{m+k,l,0}} \sum_{\eta \in R_\nu^{m,k,l}} \epsilon(\eta\circ\nu)\omega^m(u_{\eta\circ\nu(1)}, ..., u_{\eta\circ\nu(m)})\omega^k(u_{\eta\circ\nu(m+1)}, ..., u_{\eta\circ\nu(m+k)})\omega^l(u_{\eta\circ\nu(m+k+1)}, ..., u_{\eta\circ\nu(m+k+l)}) \\
=&(\omega^m \wedge \omega^k) \wedge \omega^l(u_1, ..., u_{m+k+l}).
\end{aligned}$$

- Distributivity: obvious.
- Skew-commutativity: if we note c the permutation which send the l last elements of $[\![1, k+l]\!]$ on the k first while keeping the order in the two subsets $[\![1, k]\!]$ and $[\![k+1, k+l]\!]$, we have

$$\begin{aligned}
\omega^k \wedge \omega^l(u_1, ..., u_{k+l}) =& \sum_{\sigma \in S^{0,k,l}} \epsilon(\sigma)\omega^k(u_{\sigma(1)}, ..., u_{\sigma(k)})\omega^l(u_{\sigma(k+1)}, ..., u_{\sigma(k+l)}) \\
=& \sum_{\sigma \in S^{0,k,l}} \epsilon(\sigma\circ c^{-1})\omega^k(u_{\sigma(l+1)}, ..., u_{\sigma(l+k)})\omega^l(u_{\sigma(1)}, ..., u_{\sigma(k)}) \\
=&\epsilon(c)(\omega^l \wedge \omega^k)(u_1, ..., u_{k+l}).
\end{aligned}$$

Now, we see that c has $(-1)^{kl}$ for signature. In fact, we just have to notice that we can carry out this permutation by making the last l elements of $(1, ..., k+l)$ "go up" one after the other by means of k transpositions: we bring $(k+i)$ to the desired position by means of k transpositions $(k+i-j, k+i-j-1)$ for j going from 0 to $k-1$ and this for all i going from 1 to l. By doing so, we perform kl transpositions. We therefore deduce that $\epsilon(c) = (-1)^{kl}$.

□

Example 2.1.2.

- If ω_1 and ω_2 are two 1-forms,

$$\omega_1 \wedge \omega_2(u_1, u_2) = \begin{vmatrix} \omega_1(u_1) & \omega_2(u_1) \\ \omega_1(u_2) & \omega_2(u_2) \end{vmatrix}.$$

  In that case, S contains only the identity, with positive signature, and the transposition $(1, 2)$, with negative signature.

- The product of m 1-forms is given by

$$\omega_1 \wedge ... \wedge \omega_m(u_1, ..., u_m) = \begin{vmatrix} \omega_1(u_1) & ... & \omega_m(u_1) \\ \vdots & & \vdots \\ \omega_1(u_m) & ... & \omega_m(u_m) \end{vmatrix}.$$

  Let us show this by induction. The case $m = 2$ has been treated in the previous point. Assume that we have already proved the desired equality for a certain $m \geq 2$ and check that it is still true for $m + 1$. By definition,

$$(\omega_1 \wedge ... \wedge \omega_m) \wedge \omega_{m+1} = \sum_{\sigma \in S} \epsilon(\sigma)(\omega_1 \wedge ... \wedge \omega_m)(u_{\sigma(1)}, ..., u_{\sigma(m)})\omega_{m+1}(u_{\sigma(m+1)}).$$

  Here, $S = \{\sigma \in \mathfrak{S}_{m+1} \quad | \quad \sigma(1) < ... < \sigma(m)\}$ and the sum can in fact be indexed by the $m + 1$ possible choices for $\sigma(m + 1)$, the values of the others $\sigma(i)$ being then fixed. The signature of the corresponding permutation is then $(-1)^{m+1-\sigma(m+1)} = (-1)^{m+1+\sigma(m+1)}$, as $m + 1 - \sigma(m + 1)$ is the number of transposition needed to make the $(m+1)$-th coefficient go to the $\sigma(m + 1)$-th place. Set $i = \sigma(m + 1)$, and write the sum as :

$$\sum_{i=1}^{m+1} (-1)^{i+m+1}(\omega_1 \wedge ... \wedge \omega_m)(u_1, ..., \hat{u}_i, ..., u_{m+1})\omega_{m+1}(u_i).$$

  The induction assumption yields

$$\sum_{i=1}^{m+1} (-1)^{i+m+1} \det([\omega_j(u_k)]_{j \in [\![1,m]\!]; k = [\![1,m]\!] \setminus \{i\}}),$$

  which is exactly Lagrange's developpement of the determinant from the $(m + 1)$-th column. The equality is then true for $m + 1$. We deduce that it is true for all $m \geq 2$.

- On $\mathbb{R}^{2n}$ with coordinates $(p_1, ..., p_n, q_1, ..., q_n)$, we set $\omega^2(u, v) = d\mathbf{p} \wedge d\mathbf{q} = \sum_{i=1}^n dp_i \wedge dq_i$. It is the sum of projected areas of the paralellograms formed by u and v on the subspaces (p_i, q_i).

Proposition 2.1.2. *Let $k \in \mathbb{N}$. The set of k-forms on $\mathbb{R}^n$ forms a vector space of dimension $\binom{n}{k}$.*

Moreover, if $\{e_i\}_{i=1,...,n}$ denotes a basis of $\mathbb{R}^n$ and $\{e_i^*\}_{i=1,...,n}$ the corresponding dual basis, then $\{e_{i_1}^* \wedge ... \wedge e_{i_k}^*\}_{1 \leq i_1 < ... < i_k \leq n}$ gives a basis of the k-forms space.

Proof. Let ω be a k-form on $\mathbb{R}^n$. Let $u_1, ..., u_k$ be points of $\mathbb{R}^n$ and $u_i = \sum_{j=1}^n u_i^j e_j$ their decomposition on the basis $\{e_i\}_{i=1,...,n}$. The linearity of ω and the definition of the dual basis yields

$$\omega(u_1, ..., u_k) = \sum_{j_1,...,j_k=1}^{n} u_1^{j_1} ... u_k^{j_k} \omega(e_{j_1}, ..., e_{j_k}) = \sum_{j_1,...,j_k=1}^{n} e_{j_1}^*(u_1) ... e_{j_k}^*(u_k)\omega(e_{j_1}, ..., e_{j_k}).$$

Set $\omega_{j_1,...,j_k} := \omega(e_{j_1}, ..., e_{j_k})$. By skew-symmetry, if two indices j_l and j_m are equal in the k-uplet $\{j_1, ..., j_k\}$, then $\omega_{j_1,...,j_k} = 0$. Therefore, for each $\{j_1, ..., j_k\} \in \{1, ..., n\}$ such that the corresponding term in the sum is not zero, there exists an unique permutation $\sigma \in \mathfrak{S}_k$ such that $j_{\sigma(1)} < ... < j_{\sigma(k)}\}$. Conversely, to each permutation $\sigma \in \mathfrak{S}_k$ corresponds an unique non-zero term involving $\{j_1, ..., j_k\}$.

Then,

$$\begin{aligned}\omega(u_1,...,u_k) &= \sum_{1\leq j_1<...<j_k\leq n}\sum_{\sigma\in\mathfrak{S}_k} e^*_{j_{\sigma(1)}}(u_1)...e^*_{j_{\sigma(k)}}(u_k)\omega_{j_{\sigma(1)},...,j_{\sigma(k)}}\\ &= \sum_{1\leq j_1<...<j_k\leq n}\omega_{j_1,...,j_k}\sum_{\sigma\in\mathfrak{S}_k}\epsilon(\sigma)e^*_{j_{\sigma(1)}}(u_1)...e^*_{j_{\sigma(k)}}(u_k).\end{aligned}$$

Finally, we recognize the determinant of $[e^*_{j_i}(u_l)]_{i,l=1,...,k}$, and we have

$$\omega(u_1,...,u_k) = \sum_{1\leq j_1<...<j_k\leq n}\omega_{j_1,...,j_k}e^*_{j_1}\wedge...\wedge e^*_{j_k}(u_1,...,u_k).$$

To conclude the proof, it suffices to notice that the set of ordered subsets with k elements of $[\![0,n]\!]$ has a cardinal of $\binom{n}{k}$.

□

2.1.3 Differential forms

Definition 2.1.3. *Let M be a differential manifold. A k-differential form ω is a collection of k-linear and skew-symmetric maps $\omega_x : (T_xM)^k \to \mathbb{R}$ defined at each $x\in M$ and varying differentiably with x (ie. $x\mapsto\omega_x(u_1(x),...,u_k(x))$ is differentiable for all differential vector fields $u_1,...,u_k$ on M).*

We note by $\Omega^k(M)$ the set of all differential k-forms on M and $\Omega(M)$ the set of all differential forms on M.

The exterior product between two differential forms ω^k and ω^l is defined as the form $\omega^k\wedge\omega^l$, which corresponds to $\omega^k_x\wedge\omega^l_x$ on each T_xM.

In a coordinate neighbourhood, we have for any x a differentiable basis $\{e_1,...,e_n\}$ of T_xM. Let us denote $\{dx_1,...,dx_n\}$ the 1-differentiable bases such that in any x, $\{dx_1(x),...,dx_n(x)\}$ is the dual basis of $\{e_1(x),...,e_n(x)\}$. From the previous section, any ω differential form restricted to T_xM can be decomposed into the form

$$\sum_{1\leq j_1<...<j_k\leq n}\omega_{j_1,...,j_k}(x)dx_{j_1}(x)\wedge...\wedge dx_{j_k}(x), \tag{2.1}$$

with the $\omega_{j_1,...,j_k}(x)$ given uniquely.

Defined on this neighbourhood, the applications $x\mapsto\omega_{j_1,...,j_k}(x)$ are differentiable. Conversely, any set of differentiable applications $\omega_{j_1,...,j_k}$ induces a differential form on this neighbourhood by the previous formula.

Indeed, according to the second point of the example 2.1.2, for any x in the considered neighbourhood, $dx_{j_1}(x)\wedge...\wedge dx_{j_k}(x)(e_{i_1},...,e_{i_k}) = 0$ if $\{i_1,...,i_k\}\neq\{j_1,...,j_k\}$ (since $dx_j(x_i)=\delta_{ij}$, if one of the j_q is not in the i_q, the matrix whose determinant is taken has a column of zeros). By evaluating ω in $(e_{j_1},...,e_{j_k})$, we obtain $\omega_{j_1,...,j_k}$ which must be differentiable by definition of a differential form.

Conversely, any application of the form (2.1) defines a differential form on the coordinate neighbourhood considered.

Example 2.1.3.
- A 0-form is a differentiable function and vice-versa. Note that for all 0-form f and all k-form ω, $f\wedge\omega = f\omega$.
- If $M=\mathbb{R}^{2n}$ and if $x\mapsto(p_1,...,p_n,q_1,...,q_n)(x)$ is a coordinate map, $\mathbf{p}d\mathbf{q} := \sum_{i=1}^n p_i dq_i$ and $d\mathbf{p}\wedge d\mathbf{q}$ are respectively is a 1-form and a 2-form on M.

Definition 2.1.4. *We say that a 2-differential form ω on M is non-degenerate if for all $x\in M$ and all $u\in T_xM$ different from zero, $\omega_x(u,\cdot)\neq 0$.*

Example 2.1.4. The form $d\mathbf{p} \wedge d\mathbf{q}$ is non-degenerate. In fact, for all $x \in M$ and $u \in T_xM$, $d\mathbf{p} \wedge d\mathbf{q}(u, e_j) = \sum_{i=1}^n u_i \times 0 - u_{2i}\delta_{ij} = -u_{2i}$. In the same way, $d\mathbf{p} \wedge d\mathbf{q}(u, e_{2j}) = \sum_{i=1}^n u_i\delta_{ij} - u_{2i} \times 0 = u_i$. This shows that if $d\mathbf{q}(u, \cdot)$ is zero, then this is also the case of u, which means that $d\mathbf{p} \wedge d\mathbf{q}$ is non-degenerate.

Definition 2.1.5. *Let M and N two differential manifolds and $f : M \to N$ a differentiable function. Let ω be a differential k-form on N. The pullback of ω by f is the differential k-form on M given by*

$$(f^*\omega)_x(u_1, ..., u_k) = \omega_{f(x)}(d_xf(u_1), ..., d_xf(u_k))$$

for all $x \in M$ and all $u_1, ..., u_k \in T_xM$.

Proposition 2.1.3. *Let M, N and L three differential manifolds and $f : M \to N$, $g : N \to L$ two differentiable functions.*

1. *$f^* : \omega \mapsto f^*\omega$ is a linear map from $\Omega^k(N)$ to $\Omega^k(M)$,*
2. *$(g \circ f)^* = f^* \circ g^*$,*
3. *$f^*(\omega_1 \wedge \omega_2) = f^*\omega_1 \wedge f^*\omega_2$ for all $\omega_1, \omega_2 \in \Omega(M)$.*

Proof.
1. Obvious.
2. Let $\omega \in \Omega(L)$. By definition and by the chain rule,

   $$((g\circ f)^*\omega)_x(u_1, ..., u_k) = \omega_{g(f(x))}(d_x(g\circ f)(u_1), ..., d_x(g\circ f)(u_k)) = \omega_{g(f(x))}(d_{f(x)}g(d_xf(u_1)), ..., d_{f(x)}g(d_xf(u_k))).$$

   By definition of g^* and f^*, we have

   $$((g \circ f)^*\omega)_x(u_1, ..., u_k) = (g^*\omega)_{f(x)}(d_xf(u_1), ..., d_xf(u_k)) = (f^*g^*\omega)_x(u_1, ..., u_k).$$

3. Obvious.

□

2.1.4 Exterior derivative

Proposition 2.1.4. *There exists an unique linear map $d : \Omega(M) \to \Omega(M)$, called exterior derivative verifying*

1. *if $\omega \in \Omega^k(M)$, $d\omega \in \Omega^{k+1}(M) \quad \forall k \in \mathbb{N}$,*
2. *$d(\omega^k \wedge \omega^l) = d(\omega^k) \wedge \omega^l + (-1)^k\omega^k \wedge d(\omega^l) \quad \forall \omega^k, \forall \omega^l \in \Omega(M)$ (with ω^k f degree k),*
3. *$d \circ d(\omega) = 0 \quad \forall \omega \in \Omega(M)$,*
4. *$d(f) = df \quad \forall f \in \mathcal{C}^\infty(M) = \Omega^0(M)$.*

If $\omega \in \Omega(M)$ can be decomposed in

$$\sum_{1\leq j_1<...<j_k\leq n} \omega_{j_1,...,j_k}(x)dx_{j_1}(x) \wedge ... \wedge dx_{j_k}(x)$$

in a certain coordinate system, then its derivative is given by

$$\sum_{1\leq j_1<...<j_k\leq n} d\omega_{j_1,...,j_k}(x) \wedge dx_{j_1}(x) \wedge ... \wedge dx_{j_k}(x)$$

in the same coordinate system, where $d\omega_{j_1,...,j_k}$ denotes the differential of $x \mapsto \omega_{j_1,...,j_k}(x) = \omega_x(e_{j_1}(x), ..., e_{j_k}(x))$.

Proof. Let us first verify that the given expression defines an application $\tilde{d}$ independent of the coordinate system. Consider an open set U included in an unique coordinate neighborhood of a differential manifold M and a form $\omega \in \Omega(U)$. It is obvious that the expression given for $\tilde{d}\omega$ is linear. We can therefore restrict ourselves to the case where ω_x is of the form $\phi(x)dx_{j_1} \wedge ... \wedge dx_{j_k}$ for all $x \in U$, with $\phi : U \to \mathbb{R}$ differentiable and the indices j_i such that $j_1 < ... < j_k$.

We have to write

$$d\phi(x) \wedge dx_{j_1}(x) \wedge ... \wedge dx_{j_k}(x)$$

in a form which no longer involves the coordinates x_i. In fact, we show that

$$(d\phi(x) \wedge dx_{j_1} \wedge ... \wedge dx_{j_k})(u_0, ..., u_k) = \sum_{i=0}^{k}(-1)^i d(x \mapsto \omega_x(u_0, ..., \hat{u}_i, ..., u_k)) \tag{2.2}$$

for all $x \in M$ and all $u_i \in T_xM$.

The developpement of $d\phi(x)$ and the skew-symmetry of alternate forms yield

$$\begin{aligned} d\phi(x) \wedge dx_{j_1} \wedge ... \wedge dx_{j_k}(u_0, ..., u_k) =& \Big(\sum_{p=1}^{n} \frac{\partial \phi}{\partial x_p} dx_p \wedge dx_{j_1} \wedge ... \wedge dx_{j_k}\Big)(u_0, ..., u_k) \\ =& \sum_{p \neq j_1, ..., j_k} \frac{\partial \phi}{\partial x_p} \Big(dx_p \wedge dx_{j_1} \wedge ... \wedge dx_{j_k}\Big)(u_0, ..., u_k). \end{aligned}$$

By linearity of (2.2), we can restrict ourselves to the case where the u_i are vectors e_{l_i} of the basis induced by the coordinates x_l on T_xM.

Then, all the terms in the sum are equal to zero, except maybe one, which would have to be such that $(p, j_1, ..., j_k) = (l_0, ..., l_k)$ up to a permutation σ. Indeed, $\Big(dx_{j_1} \wedge ... \wedge dx_{j_{k+1}}\Big)(u_{l_1}, ..., u_{l_{k+1}}) = \det\Big([dx_{j_s}(u_{l_r})]\Big)$ is equal to $\epsilon(\sigma)$ if this condition is satisfied and is zero otherwise.

If one of the terms is not zero, we get $\frac{\partial \phi}{\partial x_{l_0}}\epsilon(\sigma)$.

On another hand,

$$\sum_{i=0}^{k}(-1)^i d(x \mapsto \omega_x(e_{l_0}, ..., \hat{e}_{l_i}, ..., e_{l_k})) = \sum_{i=0}^{k}(-1)^i d\Big(x \mapsto \phi(x)\big(dx_{j_1} \wedge ... \wedge dx_{j_k}\big)(e_{l_0}, ..., \hat{e}_{l_i}, ..., e_{l_k})\Big),$$

with $\big(dx_{j_1} \wedge ... \wedge dx_{j_k}\big)(e_{l_0}, ..., \hat{e}_{l_i}, ..., e_{l_k})$ which, for the same reasons than previously, is equal to $\epsilon(\mu)$ if $(j_1, ..., j_k) = (l_0, ..., \hat{l}_i, ..., l_k)$ up to a permutation μ and 0 otherwise. The j_i and the l_i are fixed and different (otherwise, the sum would be zero by skew-symmetry of the dx_j and of exterior product), so this can happen for at most one term of the sum. If it exists, it is equal to

$$(-1)^i d\Big(x \mapsto \epsilon(\mu)\phi(x)\Big) = \epsilon(\nu)\frac{\partial \phi}{\partial x_{l_0}},$$

where ν is the permutation given by the composition of μ with the shift which makes l_0 go to the i-th position.

Then, it suffices to notice that, first, if a term is non-zero in the first expression, then it exists a non-zero term in the second one and, secondly, that the permutations σ and ν are the same for they both match $(p, j_1, ..., j_k)$ and $(l_0, ..., l_k)$. We immediately conclude that (2.2) is true and that the expression given in the definition really defines a map independente from the coordinate system where it is expressed.

This map is a differential form, as it can be decomposed in the same form than in (2.1) in all coordinate system :

$$\begin{aligned}
&\sum_{1\leq j_1<\ldots<j_k\leq n} d\omega_{j_1,\ldots,j_k}(x)\wedge dx_{j_1}(x)\wedge\ldots\wedge dx_{j_k}(x)\\
=&\sum_{1\leq j_1<\ldots<j_k\leq n}\sum_{l=1}^{n}\frac{\partial\omega_{j_1,\ldots,j_k}}{\partial x_l}(x)dx_l\wedge dx_{j_1}\wedge\ldots\wedge dx_{j_k}\\
=&\sum_{1\leq j_1<\ldots<j_{k+1}\leq n}\left(\sum_{l=1}^{n}(-1)^{l+1}\frac{\partial\omega_{j_1,\ldots,\hat{j_l},\ldots,j_{k+1}}}{\partial x_{j_l}}(x)\right)dx_{j_1}\wedge\ldots\wedge dx_{j_{k+1}}.
\end{aligned}$$

Let us now verify that this form satisfies axioms (1-4). From the previous paragraph, it is obvious that $\tilde{d}\in\Omega^{k+1}(M)$ and that $\tilde{d}=d$ in $\Omega^0(M)$.

By linearity and skew-symmetry, if (2) is true on the forms $\omega=\phi dx_{j_1}\wedge\ldots\wedge dx_{j_k}$ with $j_1<\ldots<j_k$, then it is also true on $\Omega(M)$. For $\phi,\psi\in\mathcal{C}^\infty(M)$, we easily check (2) from the definition of $\tilde{d}$:

$$\begin{aligned}
&\tilde{d}\big((\phi dx_{j_1}\wedge\ldots\wedge dx_{j_k})\wedge(\psi dx_{i_1}\wedge\ldots\wedge dx_{i_l})\big)\\
=&d(\phi\psi)\wedge dx_{j_1}\wedge\ldots\wedge dx_{j_k}\wedge dx_{i_1}\wedge\ldots\wedge dx_{i_l}\\
=&\psi d\phi\wedge dx_{j_1}\wedge\ldots\wedge dx_{j_k}\wedge dx_{i_1}\wedge\ldots\wedge dx_{i_l}+\phi d\psi\wedge dx_{j_1}\wedge\ldots\wedge dx_{j_k}\wedge dx_{i_1}\wedge\ldots\wedge dx_{i_l}\\
=&(d\phi\wedge dx_{j_1}\wedge\ldots\wedge dx_{j_k})\wedge(\psi dx_{i_1}\wedge\ldots\wedge dx_{i_l})+(-1)^k(\phi dx_{j_1}\wedge\ldots\wedge dx_{j_k})\wedge(d\psi\wedge dx_{i_1}\wedge\ldots\wedge dx_{i_l}).
\end{aligned}$$

Finally, to prove (3), we use the linearity of $\tilde{d}$ to decompose $\tilde{d}(\tilde{d})\omega$ into

$$\begin{aligned}
&\tilde{d}\Big[\sum_{1\leq j_1<\ldots<j_k\leq n}\sum_{l=1}^{n}\frac{\partial\omega_{j_1,\ldots,j_k}}{\partial x_l}(x)dx_l\wedge dx_{j_1}\wedge\ldots\wedge dx_{j_k}\Big]\\
=&\sum_{1\leq j_1<\ldots<j_k\leq n}\sum_{l=1}^{n}\tilde{d}\Big[\frac{\partial\omega_{j_1,\ldots,j_k}}{\partial x_l}(x)dx_l\wedge dx_{j_1}\wedge\ldots\wedge dx_{j_k}\Big]\\
=&\sum_{1\leq j_1<\ldots<j_k\leq n}\sum_{l,m=1}^{n}\frac{\partial^2\omega_{j_1,\ldots,j_k}}{\partial x_l x_m}(x)dx_l\wedge dx_m\wedge dx_{j_1}\wedge\ldots\wedge dx_{j_k}.
\end{aligned}$$

Terms sucht that $l=m$ are zero for $dx_m\wedge dx_l=0$. The others appear exactly two times with opposite signs. The nullity of the sum follows from Schwarz's lemma.

There really exists a linear operator k with properties (1-4) whose expression in local coordinates is as in the proposition. It remains to prove that this operator is unique. We show that if a linear operator $\tilde{d}$ satisfies (1-4), then its expression in local coordinates is given by the formula in the proposition.

Let ω be a differential k-form, for $k\geq 1$, and

$$\sum_{1\leq j_1<\ldots<j_k\leq n}\omega_{j_1,\ldots,j_k}(x)dx_{j_1}(x)\wedge\ldots\wedge dx_{j_k}(x)$$

its decomposition in a certain coordinate system. By (1),

$$\tilde{d}(\omega)=\sum_{1\leq j_1<\ldots<j_k\leq n}\tilde{d}\big(\omega_{j_1,\ldots,j_k}(x)dx_{j_1}(x)\wedge\ldots\wedge dx_{j_k}(x)\big).$$

Using that the product between a 0-form and a k-form corresponds to the multiplication and applying (2), terms of the previous sum can be written as

$$\tilde{d}(\omega_{j_1,\ldots,j_k}(x))\wedge\big(dx_{j_1}(x)\wedge\ldots\wedge dx_{j_k}(x)\big)+\omega_{j_1,\ldots,j_k}(x)\tilde{d}\big(dx_{j_1}(x)\wedge\ldots\wedge dx_{j_k}(x)\big).$$

Noticing that dx_i is the differential of $x\mapsto x_i$ and using (4) on $\omega_{j_1,\ldots,j_k}(x)$ and x_i, we obtain

$$d\omega_{j_1,\ldots,j_k}(x)\wedge\big(dx_{j_1}(x)\wedge\ldots\wedge dx_{j_k}(x)\big)+\omega_{j_1,\ldots,j_k}(x)\tilde{d}\big(\tilde{d}x_{j_1}(x)\wedge\ldots\wedge\tilde{d}x_{j_k}(x)\big).$$

Finally, developing

$$\tilde{d}\big(\tilde{d}x_{j_1}(x) \wedge ... \wedge \tilde{d}x_{j_k}(x)\big)$$

with (2) and applying (3), we find that this term is zero.

At the end,

$$\tilde{d}(\omega) = d\omega_{j_1,...,j_k} \wedge dx_{j_1} \wedge ... \wedge dx_{j_k},$$

which was to be proved. □

Example 2.1.5. On $\mathbb{R}^{2n}$, $d(\mathbf{p}d\mathbf{q}) = d\mathbf{p} \wedge d\mathbf{q}$.

Definition 2.1.6. *We say that a differential k-form ω on a manifold M is closed if $d\omega = 0$.*

Example 2.1.6. If a form is given by the differential of a function $f \in \mathcal{C}^\infty(M)$ (we say that it is *exact*), then it is closed by (3). The converse is false in general.

2.1.5 Symplectic manifolds

Definition 2.1.7 (Symplectic form)**.** *Let M be a differential manifold. We call a symplectic form on M a differential 2-form which is closed and non-degenerate.*

The manifold M endowed with this form ω, (M, ω) is called a symplectic manifold.

Let (M, ω) be a symplectic manifold with even dimension $2n$ (one can think about it as a physical phase space). Take a differentiable function $H : M \to \mathbb{R}$ (one can think about it as an energy function).

If we had a metric g on M, we could have defined the gradient ∇H with $d_xH = g(\nabla H(x), \cdot)$ at each point $x \in M$. Here, ω non-degeneracy allows to define a similar vector field. In fact, at each point x of M, the linear function $u \to \omega_x(u, \cdot)$ between T_xM and T_xM^* is one-to-one. As T_xM and T_xM^* have the same dimension, it is bijective. We define the vector field X_H as the one which verifies $d_xH = \omega(X_H(x), \cdot)$ for all x in M.

Proposition 2.1.5. *1. $X_H(x) \in T\{H = H(x)\}$,*

2. $L_{X_H}\omega = 0$, ie. $(\phi^t_{X_H})^*\omega = \omega$, where $\phi^t_{X_H}$ represents the flow of X_H.

Proof. 1. For $T_q\{H = H(x)\} = \ker(dH(q))$, it suffices to prove that $X_H(q) \in \ker(dH(q))$. By definition, $d_qH(X_H(q)) = \omega(X_H(q), X_H(q))$. This is zero by skew-symmetry of ω.

2. We use Cartan's formula: $L_Z\alpha = i_Zd\alpha + d(i_Z\alpha)$, where $i_Z\alpha(\cdot) = (Z, \cdot)$, for $Z = X_H$ and $\alpha = \omega$. We then have $L_{X_H}\omega = i_{X_H}d\omega + d(i_{X_H}\omega)$. The form ω is closed, so the first term is zero. The second one too as $i_{X_H}\omega = dH$ and $d \circ d = 0$.

□

Definition 2.1.8. *A Hamiltonian equation is a differential equation of the form*

$$\frac{dx(t)}{dt} = X_H(x(t)).$$

The previous proposition has for consequence that if $t \to x(t)$ is a solution of the Hamiltonian equation associated to H, then $x(t) \in \{H = H(x(0))\}$. In other words, the energy of the system is preserved over time.

The second point of the proposition implies that $(\phi^t_{X_H})^*(\overbrace{\omega \wedge ... \wedge \omega}^{ntimes}) = (\phi^t_{X_H})^*\omega \wedge ... \wedge (\phi^t_{X_H})^*\omega = \omega \wedge ... \wedge \omega$. In other words, a Hamiltonian flow preserves the volume of the phase space.

Example 2.1.7. • $\mathbb{R}^{2n}$ endowed with $d\mathbf{p} \wedge d\mathbf{q}$.

- Let M be a differential manifold of dimension n. The cotangent space T^*M is endowed with a natural symplectic structure: $\omega_{can} = d\lambda_{can}$ with $\lambda_{can}(q,p)(v) = p(\Pi_* v)$ for all $q \in M$ and all $p \in (T_qM)^*$. Here, Π represents the canonical projection of TM^* into M and Π_* its differential.

  It is obvious that λ_{can} is a 1-form and so that ω_{can} is a 2-form. It is closed because it is exact. Let us prove its non-degenerecy. Let $v = \sum_{i=1}^n x_i \frac{\partial}{\partial q_i} + y_i \frac{\partial}{\partial p_i}$. We have

  $\lambda_{can}(q,p)(v) = p(\sum_{i=1}^n x_i \frac{\partial}{\partial q_i}) = \sum_{i=1}^n x_i p(\frac{\partial}{\partial q_i}) = \sum_{i=1}^n x_i p_i$, hence $\lambda_{can} = \mathbf{p}d\mathbf{q}$ and $\omega_{can} = d\mathbf{p} \wedge d\mathbf{q}$. By a previous example, this form is non-degenerate.

A symplectic form defines a symplectic structure on the manifold. Maps that preserves this structure are called *symplectic maps.*

Definition 2.1.9. *Let (M,ω) and (N,η) be two symplectic manifolds. A differential map $f : M \to N$ is said to be symplectic if $f^*\eta = \omega$. If f is also a diffeomorphism, then it is a symplectomorphism.*

Example 2.1.8. The flow of a Hamiltonian equation is always symplectic.

2.1.6 Generating functions

In this section, we take $M = \mathbb{R}^{2n}$ endowed with the standard symplectic structure ω^2.

Given a Hamiltonian function H, we want to solve the system (2.1.8). The form of this equation is left unchanged under a symplectic transformation. We then look for such a symplectic change of coordinates $(p,q) \to (P,Q)$ which would lead to an system integrable by quadratures.

Suppose that q and Q are independant coordinates, that is $det\frac{\partial(Q,q)}{\partial(p,q)} = det\frac{\partial Q}{\partial p} \neq 0$. If $(p,q) \to (P,Q)$ is a symplectic transformation, then $pdq - PdQ$ is an exact form $d\tilde{S}$ (see [2], chapter 45 for proof). As q and Q are free, $\tilde{S}$ can be expressed in these coordinates : $S(q,Q) = \tilde{S}(p(q,Q),q)$.

Definition 2.1.10. *The function S is the generating function of the symplectic transformation $(p,q) \to (P,Q)$.*

The identity $pdq - PdQ = dS$ yields

$$\frac{\partial S}{\partial q}(q,Q) = p \qquad \text{and} \qquad \frac{\partial S}{\partial Q}(q,Q) = -P. \tag{2.3}$$

The converse is true, that is

Theorem 2.1.1. *Let $S : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ a function of (q,Q). If $det\frac{\partial^2 S}{\partial Q \partial q} \neq 0$, then S is the generating function of a symplectic transformation $(p,q) \to (P,Q)$.*

Proof. The hypothesis that $det\frac{\partial^2 S}{\partial Q \partial q} \neq 0$ allows us to use the implicit function theorem on $\frac{\partial S}{\partial q}$. We obtain a function Q depending on p and q. Then, we set $P(p,q) = P_1(Q(p,q),q)$, where $P_1(Q,q) = -\frac{\partial S}{\partial Q}(Q,q)$.

The map $(p,q) \to (P,Q)$ we obtained is symplectic with generating function S since

$$pdq - PdQ = \frac{\partial S}{\partial q}(q,Q)dq + \frac{\partial S}{\partial Q}(q,Q)dQ = dS.$$

□

This means that given a generating function S, we can easily built a symplectic transformation. This result will be used later to build numerical integrators.

2.2 Reduced order models for Hamiltonian systems

2.2.1 Hamiltonian systems

In what follows, we consider a symplectic manifold (M, ω) and we study Hamiltonian systems, i.e. systems of the form

$$\begin{cases} \dot{z} = X_H(z), \\ z(0) = z_0, \end{cases}$$

with $X_H \in \Gamma(M)$ the Hamiltionian vector field defined by

$$\omega(X_H, \cdot) = dH,$$

for $H : M \to \mathbb{R}$ a Hamiltonian function.

We also consider, in addition to the symplectic structure induced by ω on M, a Riemannian structure induced by a metric g on this same space. Like any bilinear form, the 2-form ω can be formulated in terms of g : for all $x \in M$, there exists A_{ω_x} such that for all $u, v \in T_xM$

$$\omega_x(u, v) = g(A_{\omega_x} u, v).$$

The matrix A must be skew-symmetric and nondegenerate.

We first consider $M = \mathbb{R}^{2n}$. We consider the standard symplectic form $\omega = d\mathbf{p} \wedge d\mathbf{q}$ and the Euclidean structure induced by the standard scalar product $\langle \cdot, \cdot \rangle$. In this case, for all $u, v \in \mathbb{R}^{2n}$,

$$\omega(u, v) = \langle \mathbb{J}_{2n}^T u, v \rangle,$$

where

$$\mathbb{J}_{2n} = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix}.$$

The previous system is then rewritten as

$$\dot{z} = \mathbb{J}_{2n} \nabla_z H(z).$$

If we write $z(t) = (p(t), q(t))$, this is equivalent to

$$\begin{cases} p_t &= -\frac{\partial H}{\partial q}(p, q), \\ q_t &= -\frac{\partial H}{\partial p}(p, q). \end{cases}$$

We wish to build a reduced model for the equation

$$\dot{z} = X_H(z),$$

with $z \in \mathbb{R}^{2n}$, which is also in a Hamiltonian form:

$$\dot{y} = X_{\tilde{H}}(y),$$

that is

$$\dot{y} = \mathbb{J}_{2k} \nabla_y \tilde{H}(y),$$

with $y \in \mathbb{R}^{2k}$, for a certain Hamiltonian function $\tilde{H}$ of $\mathbb{R}^{2k}$ for $k \ll n$.

2.2.2 Symplectic matrices and symplectic inverse

First, let us assume that the equation depends linearly on the parameters and on time. We then look for a matrix A such that if y verifies $Ay = z$, with z the solution of the initial problem, y is also a solution of a Hamiltonian system.

In particular, we want the decoder A to preserve the Hamiltonian structure, i.e.

$$\omega_{2n}(Au, Av) = \omega_{2k}(u, v)$$

for all $u, v \in \mathbb{R}^{2k}$.

Using the expression of ω_{2n} and ω_{2k} in terms of the scalar products on $\mathbb{R}^{2n}$ and $\mathbb{R}^{2k}$, we immediately find that a necessary and sufficient condition for A to preserve the Hamiltonian structure is given by

$$A^T \mathbb{J}_{2n} A = \mathbb{J}_{2k}.$$

A matrix $A \in \mathcal{M}_{2n,2k}(\mathbb{R})$ is said to be *symplectic* if it satisfies the above condition.

We define the *symplectic inverse* of a matrix $A \in \mathcal{M}_{2n,2k}$ as being

$$A^+ := \mathbb{J}_{2k}^T A^T \mathbb{J}_{2n}.$$

It is easy to check that if A is symplectic, then $A^+A = I_{2k}$ and $(A^+)^T$ is symplectic: if A is symplectic, then using the fact that $\mathbb{J}_{2k}$ is orthogonal,

$$A^+A = \mathbb{J}_{2k}^T A^T \mathbb{J}_{2n} A = \mathbb{J}_{2k}^T \mathbb{J}_{2k} = I_{2k}.$$

Similarly,

$$A^+ \mathbb{J}_{2n} (A^+)^T = (\mathbb{J}_{2k}^T A^T \mathbb{J}_{2n}) \mathbb{J}_{2n} (\mathbb{J}_{2n}^T A \mathbb{J}_{2k}) = \mathbb{J}_{2k}^T (A^T \mathbb{J}_{2n} A) \mathbb{J}_{2k} = \mathbb{J}_{2k}^T \mathbb{J}_{2k} \mathbb{J}_{2k} = \mathbb{J}_{2k}.$$

It is therefore in the set of symplectic matrices that we are now looking for the decoder A. Let us show that this is a sufficient condition for the resulting reduced equation to be Hamiltonian.

By replacing z by Ay in

$$\dot{z} = \mathbb{J}_{2n} \nabla_z H(z),$$

we obtain

$$A\dot{y} = \mathbb{J}_{2n} \nabla_z H(Ay).$$

Multiply the previous equation by A^+ and using the identities $\mathbb{J}_{2k}^T = -\mathbb{J}_{2k}$, $\mathbb{J}_{2n}^2 = -I_{2n}$ and $A^+A = I_{2k}$, we have

$$\dot{y} = \mathbb{J}_{2k} A^T \nabla_z H(Ay).$$

The chain rule then gives

$$\dot{y} = \mathbb{J}_{2k} \nabla_y (H \circ A)(Ay).$$

Finally, if $z = Ay$ with A a symplectic matrix, the initial problem is rewritten

$$\begin{cases} \dot{y} = \mathbb{J}_{2k} \nabla_y \tilde{H}(y), \\ y(0) = A^+ y_0, \end{cases}$$

with $\tilde{H} = H \circ A$.

2.3 Linear symplectic reductions

The idea is to find a symplectic $A \in \mathcal{M}_{2n,2k}(\mathbb{R})$ such that Ay is as close as possible to z, where y is the solution of the reduced problem induced by A and z is the solution of the initial problem.

To do this, we first compute the solution of the initial problem for some values of the parameters and time and we evaluate the difference between these solutions and their images after encoding and decoding, i.e.

$$\|S - AA^+S\|_F. \tag{2.4}$$

We would like to minimize this loss.

However, the set of symplectic matrices is not bounded and this optimization problem does not admit an explicit solution.

Several methods have been proposed to find an optimal A-matrix under additional constraints, such as the cotangent lift method, or the complex SVD. We focus here on two methods, Proper Symplectic Decomposition and another, called *greedy*, proposed in [1].

2.3.1 Greedy algorithm

The idea of the greedy method is to construct an orthosymplectic basis by adding at each iteration the vector most likely to decrease an error indicator E to be specified. As for orthogonal bases, there exists an "orthosymplectization" procedure, called *Gram-Schmidt symplectic algorithm*. We detail it below.

In greedy algorithm, each iteration follows the following pattern:

1. Choice of the parameter g_{k+1} maximising E among all possible values for the parameters.
2. Computation of the high dimensional solution for some times for this parameter: $S := \{z(t_i, g_{k+1})\}_{i=1,...,m}$.
3. Choice of the sample with the "worst" projection for the projection on the space generated by the $2k$ vectors already present in the basis: $s_{k+1} = \text{argmax}_{s\in S}\|s - A_{2k}A_{2k}^+s\|_2$.
4. Add to the basis the vector $\tilde{s}$, obtained by applying the Gramm-Schmidt symplectic orthogonalization procedure.

In point 1. we actually evaluate the maximum among the parameter values in a grid. If the calculation of the E-error is straightforward, one can take this grid very fine.

The error E considered here will be $\Delta H_k(t) := |H(z(t)) - \tilde{H}_{2k}(y_k(t))|$, with $\tilde{H}_{2k}$ and y_k the Hamiltonian function and the solution of the corresponding problem after the k-th iteration. According to the properties of the Hamiltonian flow, this quantity is independent of time and can therefore be rewritten as $\Delta H_{2k} = |H(z_0) - H(A_{2k}A_{2k}^+z_0)|$.

Symplectic Gramm-Schmidt algorithm

Suppose that we have an orthosymplectic family of vectors $A_{2k} = \{e_1, ..., e_k\} \cup \{J_{2n}^T e_1, ..., \mathbb{J}_{2n}^T e_k\}$ and an element $\tilde{v} \in \mathbb{R}^{2n}$ such that $\tilde{v} \notin Span(A_{2k})$. We want to build e_{k+1} such that $A_{2(k+1)} = \{e_1, ..., e_{k+1}\} \cup \{J_{2n}^T e_1, ..., \mathbb{J}_{2n}^T e_{k+1}\}$ is a symplectic family and $Span(A_{2(k+1)}) = Span(A_{2k} \cup \{\tilde{v}\})$. As in the Gramm-Schimdt algorithm, we look for $\bar{v}$ such that $v = \tilde{v} - \bar{v}$ is orthogonal to A_{2k} (for the scalar product). We add the symplectic constraint

$$\omega(v, u) = 0 \quad \forall u \in Span(A_{2k}).$$

Finally, we want these conditions to be also verified for $J_{2n}v$. Note that these last two conditions are in fact satisfied as soon as the first one is satisfied. Indeed,

$$\begin{aligned}\omega(v, e_i) =& \langle \mathbb{J}_{2n}^T v, e_i\rangle = \langle v, \mathbb{J}_{2n} e_i\rangle,\\ \omega(v, \mathbb{J}_{2n} e_i) =& \langle \mathbb{J}_{2n}^T v, \mathbb{J}_{2n} e_i\rangle = \langle v, \mathbb{J}_{2n}^2 e_i\rangle = -\langle v, e_i\rangle,\end{aligned}$$

which ensures that v is $\mathbb{J}_{2n}^T$-orthogonal to A_{2k} if it is already orthogonal to it and vice versa. Similarly,

$$\begin{aligned}\omega(\mathbb{J}_{2n} v, e_i) =& \langle \mathbb{J}_{2n}^T \mathbb{J}_{2n} v, e_i\rangle = \langle v, e_i\rangle,\\ \omega(\mathbb{J}_{2n} v, \mathbb{J}_{2n} e_i) =& \langle \mathbb{J}_{2n}^T \mathbb{J}_{2n} v, \mathbb{J}_{2n} e_i\rangle = \langle v, \mathbb{J}_{2n} e_i\rangle\end{aligned}$$

which ensures that $\mathbb{J}_{2n} v$ is orthogonal, and therefore $\mathbb{J}_{2n}^T$-orthogonal, to A_{2k} if v is.

Finally, since $\omega(v, v) = 0$, $\langle v, \mathbb{J}_{2n} v\rangle = 0$, while $\omega(v, \mathbb{J}_{2n} v) = \langle v, v\rangle = 0$. It is thus enough to construct $\bar{v}$ as in the traditional Gramm-Schmidt algorithm, i.e. to take $\langle v, e_i\rangle$ and $\langle v, \mathbb{J}_{2n} e_i\rangle$ for i-th and $(i+n)$-th coordinates for $\bar{v}$ in the base A_{2k}.

Finally, we set $A_{2(k+1)} = \{e_1, ..., e_{k+1}\} \cup \{J_{2n}^T e_1, ..., \mathbb{J}_{2n}^T e_{k+1}\}$.

Note that the procedure described above leads to particular orthosymplectic bases. The matrix A composed of the vectors of the basis thus constructed is of the form

$$\begin{pmatrix} \phi & -\psi \\ \psi & \phi \end{pmatrix}$$

with ϕ and ψ such that $\phi^T \phi + \psi^T \psi = I_k$ and $\phi^T \psi = \psi^T \phi$. It is thus a unitary matrix, element of a very particular subgroup of the group of symplectic matrices.

Convergence results

We have the following result (see [1] for the proof) :

Theorem 1. *Let S be a compact set of $\mathbb{R}^{2k}$ having a Kolomogorov k-width $d_k \leq c\exp(-\alpha k)$ with $\alpha > 3$. Then, it exsits $\beta > 0$ and C a constant such that the projection P_{2k} on the basis obtain after greedy algorithm verifies*

$$\|s - P_{2k}(s)\|_2 \leq C\exp(-\beta k).$$

2.3.2 Proper Symplectic Decomposition

We can further restrict the space where we look for the minimum of (2.4). The idea of Proper Symplectic Reduction (PSD), also called "cotangent lift", is to look for an operator A of the form

$$\begin{pmatrix} \phi & 0 \\ 0 & \phi \end{pmatrix},$$

with $\phi \in \mathcal{O}(n)$.

As shown in [12], this is in fact equivalent to choose the k columns of ϕ among the $\{p_1, ..., p_m, q_1, ..., q_m\}$ using classical POD.

2.4 Symplectic scheme

We present here the Störmer-Verlet scheme. It is a two-order time scheme which preserves symplectic structure of equations, in a sense which will be given below. Its construction and geometrical interpretation are taken from [7].

2.4.1 Construction

As in [7], we consider first a second order differential equation of the form

$$\ddot{q} = f(q), \tag{2.5}$$

where the right hand side does not depend on $\dot{q}$. This kind of equation is very common in physics where q represents the position of a mass point.

As usual, we discretize the time interval we consider into m points t_l separated by a time step h and look for approximate values of the exact solution at these times, q_l. We approach the second derivative in time with centered finite differences. Replacing $\ddot{q}_l$ by its approached value in (2.5), we get

$$q_{l+1} - 2q_l + q_{l-1} = h^2 f(q_l), \tag{2.6}$$

for all $l = 1, ..., m-1$. This gives q_{l+1} whenever q_l and q_{l-1} are known.

Now, we consider $v = \dot{q}$ the velocity of the mass point and its finite difference approximation

$$v_l = \frac{q_{l+1} - q_{l-1}}{2h}.$$

We also consider the staggered time grid, where

$$v_{l+1/2} = \frac{q_{l+1} - q_l}{h} \qquad \text{and} \qquad q_{l+1/2} = \frac{q_{l+1} + q_l}{2}.$$

We then express q_{l+1} in terms of q_l and $v_{l+1/2}$:

$$q_{l+1} = q_l + \frac{h}{2} v_{l+1/2}.$$

We find $v_{l+1/2}$ by making it appear in (2.6), where

$$\begin{aligned} 2q_{l+1} - 2q_l - (q_{l+1} - q_{l-1}) &= h^2 f(q_l) \\ v_{l+1/2} - v_l &= \frac{h}{2} f(q_l). \end{aligned}$$

Using (2.6) evaluated in $l+1$, we have the expression of v_{l+1} in terms of $v_{l+1/2}$ and q_{l+1} :

$$\begin{aligned} q_{l+2} - q_l - 2q_{l+1} + 2q_l &= h^2 f(q_{l+1}) \\ v_{l+1} - v_{l+1/2} &= \frac{h}{2} f(q_{l+1}). \end{aligned}$$

Finally, we can find v_{l+1} and q_{l+1} whenever we know v_l and q_l : from the two-steps scheme (2.6), we get a one-step one, explicitly given by

$$\begin{cases} v_{l+1/2} = v_l + \frac{h}{2} f(q_l), \\ q_{l+1} = q_l + \frac{h}{2} v_{l+1/2}, \\ v_{l+1} = v_{l+1/2} + \frac{h}{2} f(q_{l+1}). \end{cases} \tag{2.7}$$

Using expressions of q_l in terms of $q_{l+1/2}$ and $q_{l-1/2}$, we obtain the dual version of (2.7) :

$$\begin{cases} q_l = q_{l-1/2} + \frac{h}{2} v_{l-1/2}, \\ v_{l+1/2} = v_{l-1/2} + h f(q_l), \\ q_{l+1/2} = q_l + \frac{h}{2} v_{l+1/2}. \end{cases} \tag{2.8}$$

Both equations leads to the same scheme. Essentially, a step consists in a uniform motion during a half time interval, followed by a correction of the trajectory and another uniform motion for the remaining half time step.

2.4.2 Geometrical properties

In what follows, we consider Hamiltonian systems whose energy is a sum of a kinetic and a potential energy U :

$$H(p,q) = \frac{1}{2}p^T M^{-1} p + U(q), \tag{2.9}$$

where the momentum is $p = M\dot{q}$, with M a symmetric positive definite mass matrix.

We easily find that

$$\nabla_q H(p,q) = \nabla U(q) \qquad \text{and} \qquad \nabla_p H(p,q) = M^{-1}p = v.$$

Then, the canonical equations for this system are

$$\begin{cases} \dot{p} = -\nabla U(q), \\ \dot{q} = M^{-1}p. \end{cases} \tag{2.10}$$

It is a particular case of (2.5) for $f(q) = -M^{-1}\nabla U(q)$. Multiplying equations involving v_l by M in (2.7) and (2.8), we get

$$\begin{cases} p_{l+1/2} = p_l - \dfrac{h}{2}\nabla_q H(q_l), \\ q_{l+1} = q_l + \dfrac{h}{2}\nabla_p H(p_{l+1/2}), \\ p_{l+1} = p_{l+1/2} - \dfrac{h}{2}\nabla_q H(q_{l+1}). \end{cases} \quad \text{and} \quad \begin{cases} q_l = q_{l-1/2} + \dfrac{h}{2}\nabla_p H(p_{l-1/2}), \\ p_{l+1/2} = p_{l-1/2} - h\nabla_q H(q_l), \\ q_{l+1/2} = q_l + \dfrac{h}{2}\nabla_p H(p_{l+1/2}). \end{cases} \tag{2.11}$$

Applied to this kind of systems, Störmer-Verlet scheme is symplectic, in the sense of the following theorem, taken form [7] :

Theorem 2.4.1. *The numerical flow associated to Störmer-Verlet scheme applied to (2.10)* $\Phi : (p_l, q_l) \to (p_{l+1}, q_{l+1})$ *is a symplectic map.*

Proof. To prove this result, we use generating functions. More precisely, we find a function $S_h : (q_l, q_{l+1}) \to (p_l, p_{l+1})$ which satisfies (2.3).

To do that let us forget Störmer-Verlet scheme a moment and consider the least action principle, which states that the trajectories $(p(t), q(t))$ are extremum of

$$\int_\Omega \mathcal{L}(q, \dot{q}), \tag{2.12}$$

with $\mathcal{L}$ is the Lagrangian of the system, given by the Legendre's transform of H, $\mathcal{L}(q,v) = p \cdot v - H(p,q)$ with $p(v) = \nabla_v \mathcal{L}(q, \dot{q})$.

We approach the integral with trapezes formula, which gives

$$\sum_{l=0}^{m-1} S(q_l, q_{l+1}),$$

with $S(q_l, q_{l+1}) = \frac{h}{2}(\mathcal{L}(q_l, v_{l+1/2}) + \mathcal{L}(q_{l+1}, v_{l+1/2}))$. We are looking for $(q_0, ..., q_m)$ for which this discrete version of (2.12) reaches an extremum. The componants of this sum gradient of this sum are zeros if and only if

$$\nabla_q S(q_l, q_{l+1}) + \nabla_Q S(q_{l-1}, q_l) = 0 \tag{2.13}$$

for all $l = 0, ..., m-1$, where q and Q respectively denotes the first and the second variable of S.

We have that

$$\nabla_q S(q_l, q_{l+1}) = \frac{h}{2}\nabla_q \mathcal{L}(q_l, v_{l+1/2}) - \frac{1}{2}(\nabla_v \mathcal{L}(q_l, v_{l+1/2}) + \nabla_v \mathcal{L}(q_{l+1}, v_{l+1/2})$$

and

$$\nabla_Q S(q_l, q_{l+1}) = \frac{h}{2}\nabla_q \mathcal{L}(q_{l+1}, v_{l+1/2}) + \frac{1}{2}(\nabla_v \mathcal{L}(q_l, v_{l+1/2}) + \nabla_v \mathcal{L}(q_{l+1}, v_{l+1/2}).$$

In the case of a Hamiltonian function of the form (2.9),

$$\mathcal{L}(q, v) = \frac{1}{2}p^T M^{-1} p - U(q).$$

Indeed, as $v = \dot{q} = \nabla_p H(p, q) = M^{-1}p$, we have $p \cdot v = p^T M^{-1} p$, from which we immediately get $\mathcal{L}(q, v) = \frac{1}{2}p^T M^{-1} p - U(q) = \frac{1}{2}v^T M v - U(q)$. Thus, $\nabla_q \mathcal{L}(q, v) = -\nabla U(q)$ and $\nabla_v \mathcal{L}(q, v) = Mv$.

For such systems, (2.13) is equivalent to

$$\left(\frac{h}{2}\nabla U(q_l) - Mv_{l+1/2}\right) + \left(\frac{h}{2}\nabla U(q_l) + Mv_{l-1/2}\right) = 0,$$

that is, after multiplication by $-hM^{-1}$,

$$-h^2 f(q_l) + (q_{l+1} - 2q_l + q_{l-1}) = 0,$$

which is exactly the equation from which we set the q_l ad the p_l.

On another hand, using the first line of (2.7) we have

$$\begin{cases} \nabla_q S(q_l, q_{l+1}) = \dfrac{h}{2}\nabla U(q_l) - Mv_{l+1/2} = \dfrac{h}{2}M\dfrac{2}{h}(v_{n+1/2} - v_n) - Mv_{l+1/2} = -Mv_l = -p_l, \\ \nabla_Q S(q_l, q_{l+1}) = \dfrac{h}{2}\nabla U(q_{l+1}) + Mv_{l+1/2} = M(v_{l+1+1/2} - v_{l+1}) + Mv_{l+1/2} = Mv_{l+1} = p_{n+1}, \end{cases}$$

which means that S is the generating function of the transformation $(p_l, q_l) \to (p_{l+1}, q_{l+1})$. We deduce that the numerical flow is a symplectic map. □

2.4.3 Generalisation to general Hamiltonian systems

The schemes (2.11) can be built for all Hamiltonian systems, even if there are not of the form (2.10). The following generalisation is also taken from [7].

Taking the first and the last equations of (2.7) and (2.8) for $l + 1/2$ and $l + 1$, we get the map $\Phi_A : (p_l, q_l) \to (p_{l+1/2}, q_{l+1/2})$ and its adjoint $\Phi_B : (p_{l+1/2}, q_{l+1/2}) \to (p_{l+1}, q_{l+1})$:

$$\begin{cases} v_{l+1/2} &= v_l + \frac{h}{2}f(q_l), \\ q_{l+1/2} &= q_l + \frac{h}{2}v_{l+1/2}, \end{cases} \qquad \text{and} \qquad \begin{cases} q_{l+1} &= q_{l+1/2} + \frac{h}{2}v_{l+1/2}, \\ v_{l+1} &= v_{l+1/2} + \frac{h}{2}f(q_{l+1}). \end{cases} \tag{2.14}$$

Numerical flows of schemes (2.7) and (2.8) are respectively the compositions $\Phi_A \circ \Phi_B$ and $\Phi_B \circ \Phi_A$.

This interpretation, given in [7], allows generalisation of (2.7) and (2.8) to most general systems of differential equations, where

$$\dot{q} = g(p, q) \qquad \text{and} \qquad \dot{v} = f(p, q).$$

So far, we worked with $g(p, q) = M^{-1}p$.

In [7], authors extend (2.14) with

$$\begin{cases} v_{l+1/2} &= v_l + \frac{h}{2}f(q_l, v_{l+1/2}), \\ q_{l+1/2} &= q_l + \frac{h}{2}g(q_l, v_{l+1/2}), \end{cases} \qquad \text{and} \qquad \begin{cases} q_{l+1} &= q_{l+1/2} + \frac{h}{2}f(q_{l+1}, v_{l+1/2}), \\ v_{l+1} &= v_{l+1/2} + \frac{h}{2}g(q_{l+1}, v_{l+1/2}). \end{cases} \tag{2.15}$$

After composing the two numerical flows, we get

$$\begin{cases} p_{l+1/2} = p_l - \dfrac{\Delta t}{2}\nabla_q H(q_l, p_{l+1/2}) \\ q_{l+1} = q_l + \dfrac{\Delta t}{2}\big(\nabla_p H(q_l, p_{l+1/2}) + \nabla_p H(q_{l+1}, p_{l+1/2})\big), \\ p_{l+1} = p_{l+1/2} - \dfrac{\Delta t}{2}\nabla_q H(q_{l+1}, p_{l+1/2}), \end{cases} \tag{2.16}$$

and

$$\begin{cases} q_{l+1/2} = q_l + \dfrac{\Delta t}{2}\nabla_p H(q_{l+1/2}, p_l), \\ p_{l+1} = p_l - \dfrac{\Delta t}{2}\big(\nabla_q H(q_{l+1/2}, p_l) + \nabla_q H(q_{l+1/2}, p_{l+1})\big), \\ q_{l+1} = q_{l+1/2} + \dfrac{\Delta t}{2}\nabla_p H(q_{l+1/2}, p_{l+1}). \end{cases} \tag{2.17}$$

Note that the resulting schemes are implicit when the Hamiltonian function is not separable.

2.4.4 Comparison with Euler scheme

We have tested Störmer-Verlet model on the simple pendulum system, whose Hamiltonian function is

$$H(p,q) = \frac{1}{2}p^2 + 10(1-\cos(q)).$$

We see on Figure 2.4.4 that the numerical motion computed with Störmer-Verlet scheme is periodic, as expected, and that the energy oscillates between two fixed values. On the contrary, classical models as explicit or implicit Euler schemes do not conserve energy : it decreases (implicit) or increases (explicit). The pendulum makes bigger (explicit) or shorter (implicit) oscillations with time and the motion is not periodic.

2.5 Application to a piano vibrating string

We test the reduction on a set of equations modelling a piano string vibration, proposed in [6].

We consider the following problem:

$$\begin{cases} \partial_{tt}^2 U(x,t) = \partial_x \Big[\nabla V(\partial_x U(x,t))\Big] & \forall (x,t) \in \Omega \times \mathbb{R}_+ \\ U(x,0) = U_0(x) & \forall x \in \Omega, \\ \partial_t U(x,0) = U_1(x) & \forall x \in \Omega, \\ U(x,t) = 0 & \forall (x,t) \in \partial\Omega \times \mathbb{R}_+. \end{cases}$$

In what follows, $U(x,t) = (v(x,t), u(x,t))$ represents the longitudinal and transverse variations of the position of the point x in a piano string on the oscillation plane. The domain Ω is the interval $[0,1]$.

Let $q = (u,v)$ and $p = (\partial_t u, \partial_t v)$. The previous equation is rewritten

$$\begin{cases} \dfrac{\partial q}{\partial t} = p, \\ \dfrac{\partial p}{\partial t} = \partial_x \Big[\nabla V(\partial_x q)\Big]. \end{cases}$$

The has an Hamiltonian formulation with the energy function

$$H(p,q,t) = \int_\Omega \frac{1}{2}|p|^2 + V(\partial_x q)dx.$$

Indeed, on the one hand

$$H(p+p',q,t) = \int_\Omega \frac{1}{2}|p|^2 + V(\partial_x q)dx + \int_\Omega p \cdot p' dx + \int_\Omega \frac{1}{2}|p'|^2 dx = H(p,q,t) + \langle p, p' \rangle + o(|p'|),$$

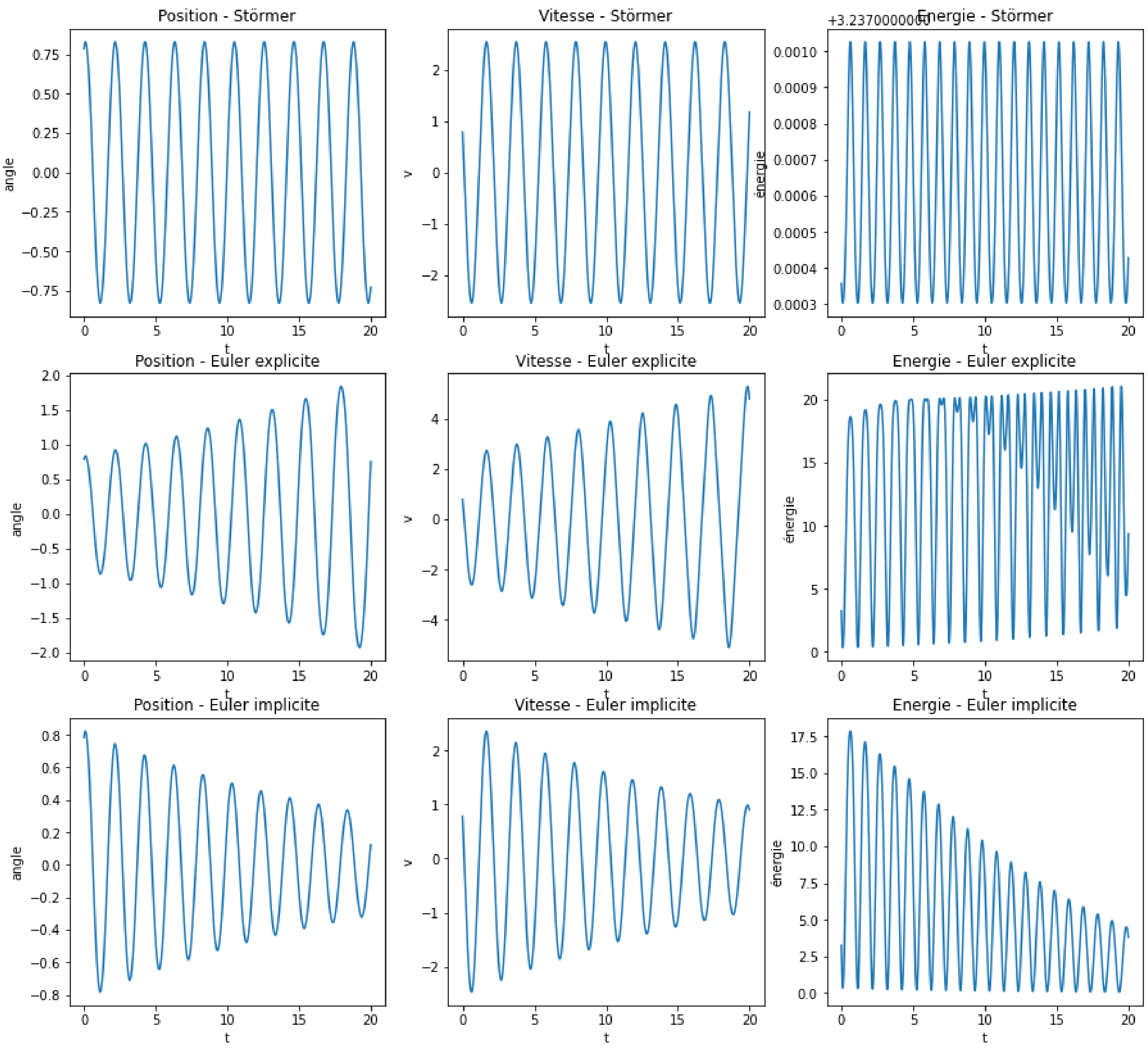


Figure 2.2: Simple pendulum motion numerically computed with Störmer-Verlet, Euler explicit and Euler implicit schemes. The mass of the pendulum and its length were respectively equal to 1 and 10. The initial angle and speed were set to $\frac{\pi}{4}$. The computation was made with a time step $dt = 0.01$ during $m = 2000$ steps.

from which

$$\nabla_p H(p,q,t) = p.$$

On the other hand,

$$H(p,q+q',t) = \int_\Omega \frac{1}{2}|p|^2 + V(\partial_x q)dx + \int_\Omega \nabla V(\partial_x q)\cdot\partial_x q' dx + \int_\Omega o(|\partial_x q'|) = H(p,q,t) + \int_\Omega \nabla V(\partial_x q)\cdot \partial_x q' dx + o(|q'|).$$

After an integration by parts, since by hypothesis q' is null on $\partial\Omega$, we find

$$\int_\Omega \nabla V(\partial_x q)\cdot \partial_x q' dx = -\int_\Omega \partial_x \nabla V(\partial_x q)\cdot q' dx,$$

from which

$$\nabla_q H(p,q,t) = -\partial_x \nabla V(\partial_x q).$$

The Hamiltonian is thus separated. The term in p represents the kinetic energy and that in q the potential one. We study different expressions for V, all given in [6].

2.5.1 Linear model

The model

We consider the case where the potential energy is given by $V(u,v) = \frac{(1-\alpha)}{2}u^2 + \frac{1}{2}v^2$, with α a parameter depending on the string caracteristics.

This yields the following system :

$$\begin{cases} \partial^2_{tt} u = \partial_x\Big[(1-\alpha)\partial_x u\Big] = (1-\alpha)\partial^2_{xx} u, \\ \partial^2_{tt} v = \partial_x\Big[\partial_x v\Big] = \partial^2_{xx} v, \end{cases}$$

or, in Hamiltonian formulation

$$\begin{cases} \nabla_p H(p_u,p_v,q_u,q_v) = (p_u,p_v), \\ \nabla_q H(p_u,p_v,q_u,q_v) = -\big((1-\alpha)\partial^2_{xx} q_u, \partial^2_{xx} q_v\big). \end{cases}$$

In a similar way than proposed in [1], we build a discrete approxiation of ∇H by approximating second derivatives with finite differences. We obtain

$$\nabla H \left(\begin{smallmatrix} p \\ q \end{smallmatrix}\right) \approx M_l \left(\begin{smallmatrix} p \\ q \end{smallmatrix}\right),$$

with

$$M_l = \begin{pmatrix} I_{2n} & 0_{2n} \\ 0_{2n} & B_n \end{pmatrix},$$

where

$$B_n = \begin{pmatrix} (1-\alpha)L & 0_n \\ 0_n & L \end{pmatrix}$$

and

$$L = \frac{1}{\Delta_x^2}\begin{pmatrix} 2 & -1 & & -1 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ -1 & & -1 & 2 \end{pmatrix}.$$

Results

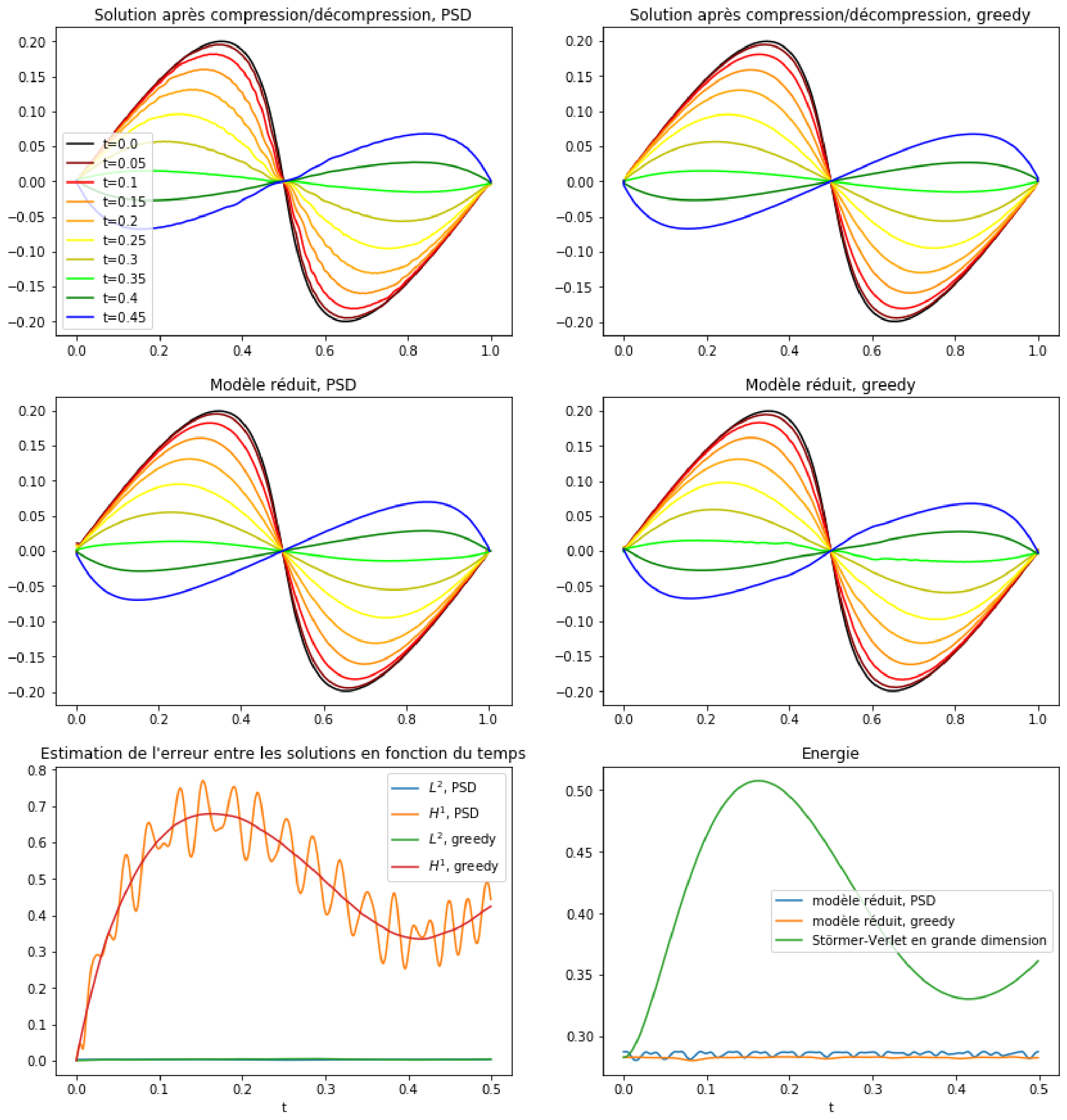


Figure 2.3: Numerical solution of piano string equation with $V = \frac{1}{2}((1-g)u^2+v^2)$, where $g = 0.537$. The reductions were made using greedy algorithm and PSD with $k = 5$. Trajectories were computed in high dimension with explicit Stormer-Verlet scheme, for values of g taken in $I = [0, 1]$, $n = 200$, $dt = 0.001$ and $m = 500$. For PSD, we have taken 5 values of the parameter g uniformly sampled in I. Trajectories in low dimension were computed using the reduced model with the same discretisation in time and space and with $g = 0.537$ (this value has been chosen to be different from those used for the projection). the two first lines represent the piano string position in the oscillation plane for different times. the first line represents the solution computed in high dimension with Stormer-Verlet scheme after being compressed in low dimension and decompressed (that is AA^+z). The second line represents the solution computed in low dimension using reduced models. At bottom right are the errors, in L^2 and H^1 norms, between the solutions computed with the reduced model and the solution computed in high dimension. At bottom left is the variation of energy of those solutions with time.

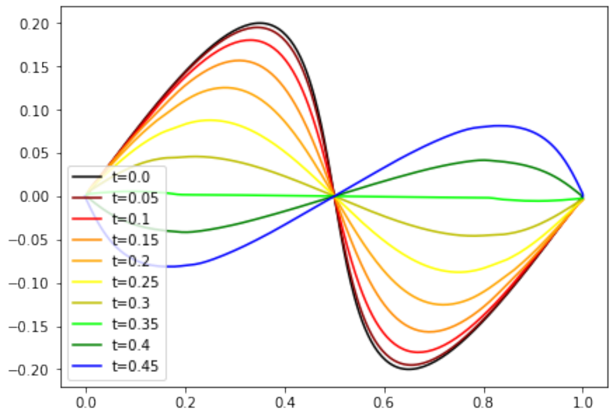


Figure 2.4: Numerical solution of piano string equation with $V = \frac{1}{2}((1-g)u^2 + v^2)$, where $g = 0.537$ computed in high dimension with explicit Stormer-Verlet scheme, with $n = 200$, $dt = 0.0005$ and $m = 500$.

We see on Figure 2.3 that both PSD and greedy algorithm lead to satisfying models, which gives a solution $\mathcal{C}^0$-close from the solution computed in high dimension (see Figure 2.4 for comparison). The energy is constant and close to the initial energy of the solution computed in high dimension. From this point of view, the reduced model is even better than the initial one, in which energy undergoes more important variations.

As for Burgers' equation, computation using the reduced order model is faster than the one using Störmer-Verlet in high dimension (a quasi-instantaneous computation compared to a few seconds computation).

2.5.2 Non-linear model

The model

We now consider antoher model, where V is non-linear: $V(u,v) = \frac{1-\alpha}{2}u^2 + \frac{1}{2}v^2 + \frac{\alpha}{2}(u^2 v + \frac{1}{4}u^4)$.

This yields the following system :

$$\begin{cases} \partial_{tt}^2 u = \partial_x \Big[(1-\alpha)\partial_x u + \alpha(\partial_x u \partial_x v + \dfrac{1}{2}(\partial_x u)^3)\Big], \\ \partial_{tt}^2 v = \partial_x \Big[\partial_x v + \dfrac{\alpha}{2}(\partial_x u)^2\Big]. \end{cases}$$

As previously, we approach first and second derivatives with finite differences :

$$\partial_x u \approx \frac{u(x^{i+1}) - u(x^i)}{\Delta x}$$

and

$$\partial_x^2 u \approx \frac{u(x^{i+1}) - 2u(x^i) + u(x^{i-1})}{\Delta x^2}.$$

Results

As we see on Figure 2.5, reduction using PSD as well as greedy algorithm gives inaccurate reduced models for $k = 5$. The reduced model built with greedy algorithm does not produce the sharp bumps which can be seen on the solution computed in high dimension Figure 2.8. On the contrary, the "bumps" are too sharp on the solution computed with PSD and one can see some oscillations. Looking at H^1 errors and energy, the model built with PSD produces an unstable solution.

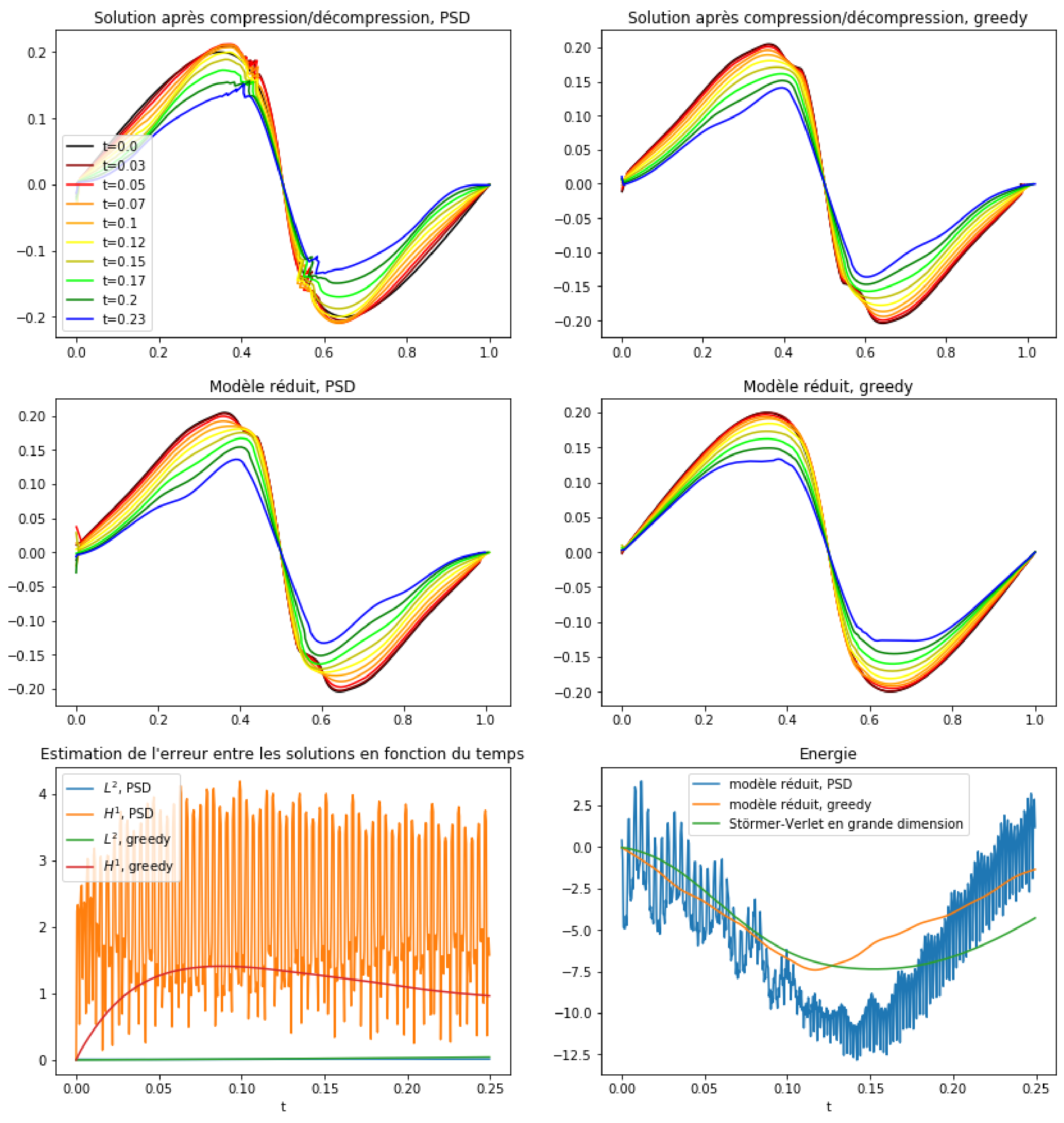


Figure 2.5: Numerical solution of piano string equation with $V = \frac{1}{2}((1-g)u^2 + v^2) + (\frac{g}{2}(u^2v + \frac{1}{4}u^4)$, where $g = 0.8$. The reductions were made using greedy algorithm and PSD with $k = 5$. Trajectories were computed in high dimension with explicit Stormer-Verlet scheme, for values of g taken in $I = [0, 0.8]$, $n = 200$, $dt = 0.0005$ and $m = 500$. For PSD, we took 20 trajectories, with values of g uniformly sampled in I. Trajectories in low dimension were computed using the reduced model with the same discretisation in time and space and with $g = 0.8$. the two first lines represent the piano string position in the oscillation plane for different times. the first line represents the solution computed in high dimension with Stormer-Verlet scheme after being compressed in low dimension and decompressed (that is AA^+z). The second line represents the solution computed in low dimension using reduced models.

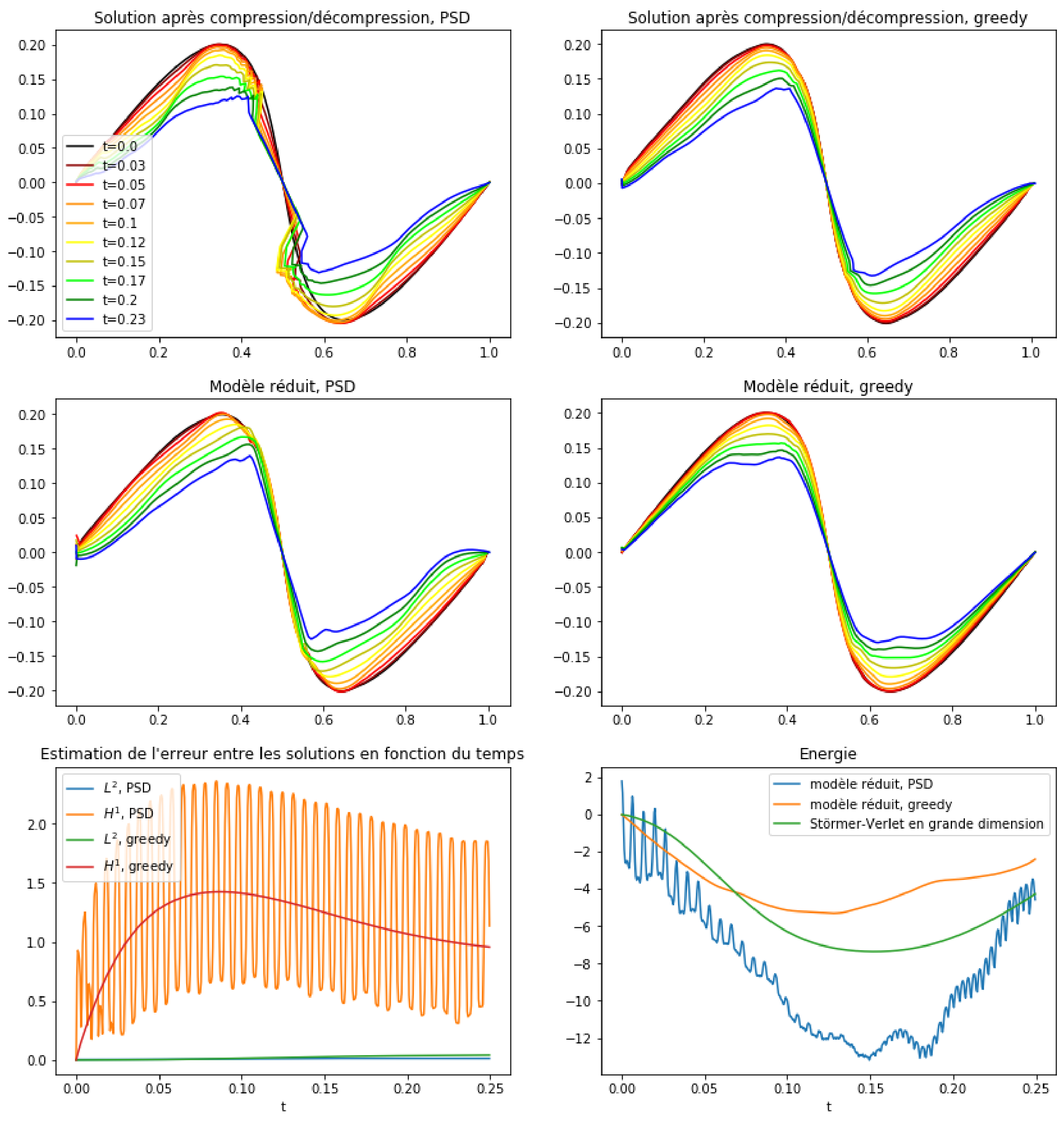


Figure 2.6: Numerical solution of piano string equation with $V = \frac{1}{2}((1-g)u^2+v^2)+(\frac{g}{2}(u^2v+\frac{1}{4}u^4)$, where $g = 0.8$. The reductions were made using greedy algorithm and PSD with $k = 10$. Trajectories were computed in high dimension with explicit Stormer-Verlet scheme, for values of g taken in $I = [0, 0.8]$, $n = 200$, $dt = 0.0005$ and $m = 500$. For PSD, we took 20 trajectories, with values of g uniformly sampled in I. Trajectories in low dimension were computed using the reduced model with the same discretisation in time and space and with $g = 0.8$. the two first lines represent the piano string position in the oscillation plane for different times. the first line represents the solution computed in high dimension with Stormer-Verlet scheme after being compressed in low dimension and decompressed (that is AA^+z). The second line represents the solution computed in low dimension using reduced models.

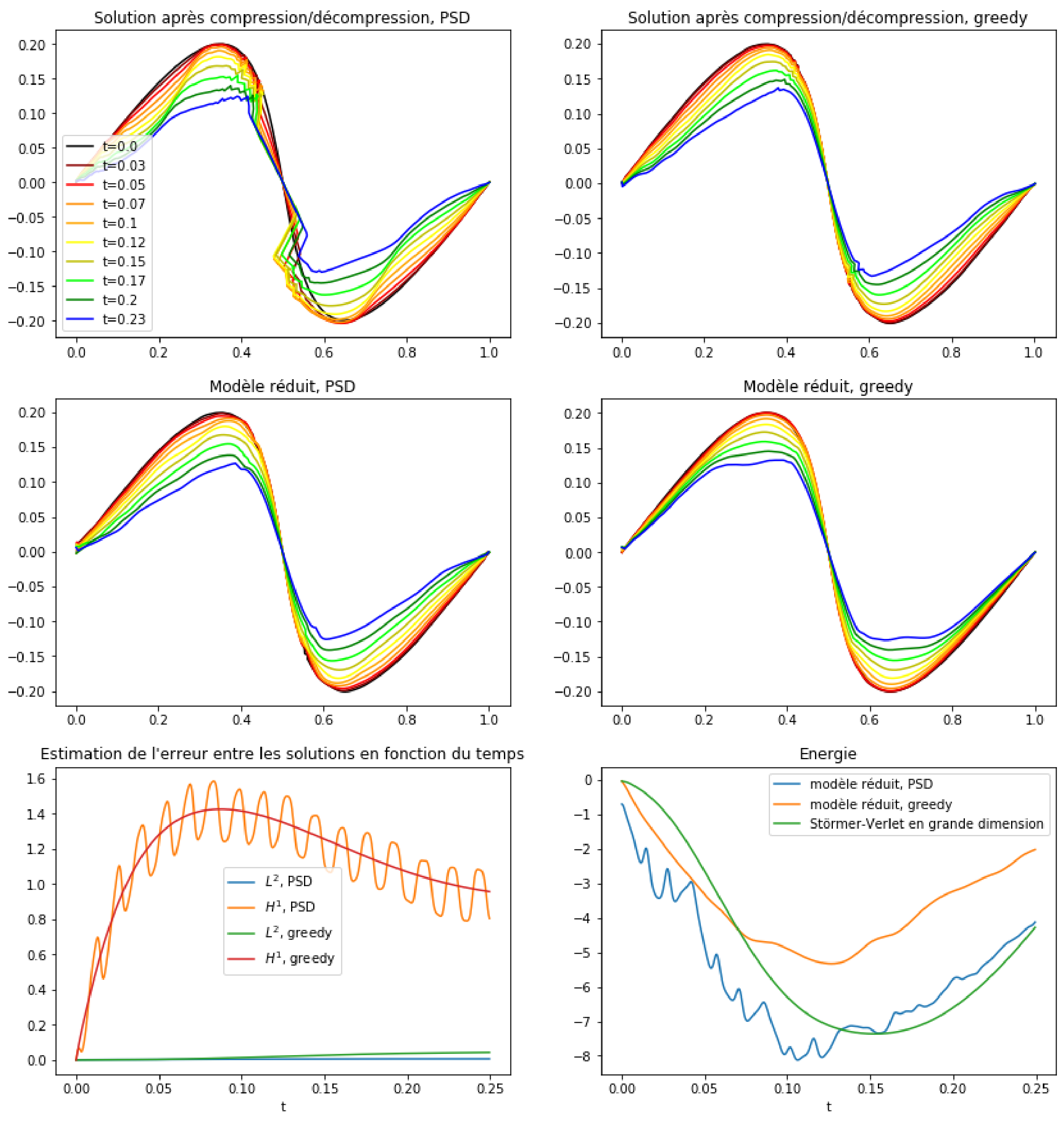


Figure 2.7: Numerical solution of piano string equation with $V = \frac{1}{2}((1-g)u^2+v^2)+(\frac{g}{2}(u^2v+\frac{1}{4}u^4)$, where $g = 0.8$. The reductions were made using greedy algorithm and PSD with $k = 20$. Trajectories were computed in high dimension with explicit Stormer-Verlet scheme, for values of g taken in $I = [0, 0.8]$, $n = 200$, $dt = 0.0005$ and $m = 500$. For PSD, we took 20 trajectories, with values of g uniformly sampled in I. Trajectories in low dimension were computed using the reduced model with the same discretisation in time and space and with $g = 0.8$. the two first lines represent the piano string position in the oscillation plane for different times. the first line represents the solution computed in high dimension with Stormer-Verlet scheme after being compressed in low dimension and decompressed (that is AA^+z). The second line represents the solution computed in low dimension using reduced models.

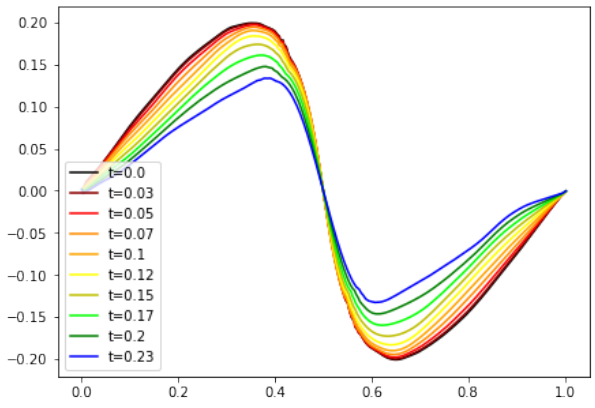


Figure 2.8: Numerical solution of piano string equation with $V = \frac{1}{2}((1-g)u^2+v^2)+(\frac{g}{2}(u^2v+\frac{1}{4}u^4)$, where $g = 0.8$ computed in high dimension with explicit Stormer-Verlet scheme, with $n = 200$, $dt = 0.0005$ and $m = 500$.

When we take $k = 10$, these problems are still visible on both models, as one can see on Figure 2.7. To obtain a satisfactoring solution with the reduced model, one should increase the reduced space dimension and take $k = 20$. This in fact means that the reduction failed because we did not succeed in capturing the low dimensional structure of the problem. As for Burgers' equation when the parameter ϵ is close to 1, we can deduce that the problem we want to solve here is too far from being linear to admit a linear reduction. If we still want to use greedy algorithm or PSD, we thus have to look further to improve the reduction.

2.6 Ideas to look into for non-linear cases

2.6.1 Hyper-reduction

In practice, even in linear case, solving the system with $\tilde{H} = H \circ A$ requires a lot of computation time, as the evaluation of H and ∇H requires the evaluation of H and a multiplication by A. The idea is therefore to find a continuous operator $\bar{H} : \mathbb{R}^{2k} \to \mathbb{R}$ close to $\tilde{H}$ whose expression does not depend on A and H.

A simple way to construct $\bar{H}$ is to proceed by kernel regression. For a certain kernel K, the Hamiltonian takes the form

$$\bar{H}(x) = \sum_{i=1}^{m} \theta_i K(x^i, x)$$

with the θ_i being real coefficients such that the difference between $\bar{H}$ and $H \circ A$ is minimal. The $\{x^i\}_{i=1,\dots,m}$ are the interpolation points. Here, they will be Az^i with $z^i = (p_u^i, p_v^i, q_u^i, q_v^i)$ the solution of the equation computed in high dimension at different times and for different values of the parameter g.

This is in fact the difference between these two functions evaluated in the samples that we seek to minimise, more precisely

$$\mathcal{L}(\theta) = \sum_{i=1}^{m} |\bar{H}(x_i) - H(Ax_i)|^2 = \|y - \mathbb{K}\theta\|_2^2,$$

where $y \in \mathbb{R}^m$ is such that $y_i = H(x_i)$ and $\mathbb{K}_{ij} = K(x_i, x_j)$.

If K is not singular, it is clear that $\mathcal{L}$ is coercive, which implies that it admits at least one minimizer, which verifies $\nabla\mathcal{L}(\theta) = 0$. This happens if and only if $\theta = (\mathbb{K}^T\mathbb{K})^{-1}\mathbb{K}^T y$, which is thus the unique minimizer.

When we apply this on $H \circ A$, we obtain a function $H_\theta : \mathbb{R}^{2k} \times \mathbb{R}^{2k} \to \mathbb{R}$ with fits almost perfectly $H \circ A$. However, trajectories obtained with Störmer-Verlet scheme in low dimension are stationary.

In fact, by doing so we build $\bar{H}$ to be $\mathcal{C}^0$ close to $H \circ A$. Nevertheless, in the Hamiltonian system we are looking to solve, $\tilde{H}$ appears only through its gradient.

Maybe, looking for a $\bar{H}$ which would be not only $\mathcal{C}^0$ but also $\mathcal{C}^1$ close to $H \circ A$ would improve the reduce order model.

A more precise idea is to consider the loss

$$\mathcal{L}(\theta) = \alpha \sum_{i=1}^{m} \| \frac{A^+ z^{i+1} - A^+ z^i}{\Delta t} - \mathbb{J}_{2k} \nabla \bar{H}(x^i) \|^2 + \beta \sum_{i=1}^{m} |\bar{H}(x^i) - H(Ax^i)|^2.$$

Note that the case where $\alpha = 0$ and $\beta = 1$ corresponds to the previous loss. Let $\Delta^i = \frac{A^+ z^{i+1} - A^+ z^i}{\Delta t}$ and $d_x K^i = \left(\frac{\partial K}{\partial y_j}(x^k, x^i)\right)_{kj}$.

Each term which composes $\mathcal{L}$ is of the form $\|Ax - a\|$, whose gradient is $-2A^T(A - a)$. Thus, $\mathcal{L}$ is differentiable and its gradient is given by

$$\begin{aligned}
\nabla_\theta \mathcal{L}(\theta) =& -2 \left[\alpha \left(\sum_{i=1}^{m} (d_x K^i)^T \left(\mathbb{J}_{2k} \Delta^i - d_x K^i \theta \right) \right) + \beta \mathbb{K}^T (y - \mathbb{K}\theta) \right] \\
=& 2 \left[\left(\beta \mathbb{K}^T y + \alpha \sum_{i=1}^{m} (d_x K^i)^T \mathbb{J}_{2k} \Delta^i \right) - \left(\beta \mathbb{K}^T \mathbb{K} + \alpha \sum_{i=1}^{m} (d_x K^i)^T (d_x K^i) \right) \theta \right].
\end{aligned}$$

Then, $\nabla \mathcal{L}$ is zero if and only if

$$\theta = (\alpha B + \beta \mathbb{K}^T \mathbb{K})^{-1} (\beta \mathbb{K}^T y + \alpha b),$$

with $b \in \mathbb{R}^m$ and $B \in \mathcal{M}_{m,m}(\mathbb{R})$ such that

$$b = \sum_{i=1}^{m} (d_x K^i)^T \mathbb{J}_{2k} \Delta^i \qquad \text{and} \qquad B_{ij} = \sum_{i=1}^{m} (d_x K^i)^T (d_x K^i).$$

In what follows, we use the Matérn kernel, whose expression is

$$K(x, y) = \frac{1}{\Gamma(\nu) 2^{\nu-1}} \left(\frac{\sqrt{2\nu}}{\sigma} \right)^{\nu} \|x - y\|_2^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{\sigma} \|x - y\|_2 \right),$$

with K_{nu} the Bessel function with parameter ν. Here, sigma is the variance of the kernel.

When $\nu = 0.5$, K is the exponential kernel and at the limit when $\nu \to \infty$, K corresponds to the gaussian kernel. As ν increases, K becomes more regular. More precisely, K is d times differentiable if and only if $\nu > d$ [13].

In the case of Matérn's kernel, we have

$$(d_x K^i)_{kj} = \frac{\partial K}{\partial x_k}(x^i, x^j) = \frac{2\nu c_1}{\sigma^2} (x_k^i - x_k^j) \left(\nu K_\nu(c_1) + c_1 K'_\nu(c_1) \right),$$

where $c_1 = \frac{\sqrt{2\nu}}{\sigma} \|x^i - x^j\|_2$ and $c_2 = \frac{1}{\Gamma(\nu) 2^{\nu-1}} c_1^{\nu-2}$.

Results

Here, we put this method into practice by taking for the x^i snapshots of 20 trajectories computed in high dimension for different values of g. To reduce the computational cost of the method, we take only the solution at one time t^n every ten, so we have 20×100 samples. On Figure 2.10, we see that including the derivatives of H_θ in the loss leads to a better reduced model: for the value of parameter chosen

($g = 0.783$, a value different from those used during th construction of A) the trajectory associated to H_θ is no longer stationary and is close to the one associated to $H \circ A$. The difference with the trajectory computed in high dimension is not smaller but is of the same order.

However, the energy of the solution computed with the reduced model decreases linearly with time, as we can better see on Figure 2.12. This shows that we didn't succed to built a symplectic reduced model, as energy is not conserved. Yet, as shown on Figure 2.9, if H_θ no longer matches exactly $H \circ A$, what we expected because we have taken $\alpha > 0$, it is still close to it.

Moreover, we see on Figure 2.11 that a little change of α yields a completetly different solution, close to stationary solutions. This means that the hyper-reduction is very sensitive to the value of its parameters, which may be problematic to find the good parameters.

For example, we have tried to use hyper-reduction method on the linear model, for which the reduced oder model give very good results, in order to see if this method could be generalized. In that case too, we experimented a strong dependance on the parameters, especially α. On Figure 2.13, we show the better solution we obtained. Visually, it is far from the solution we expected (compare with Figure 2.3) but this does not mean that it is impossible to obtain a good result with this method. Maybe we just didn't find the good parameters.

At least, these tests show that it is imperative to consider ∇H and not only H to construct $\bar{H}$, in other words to approach H in a $\mathcal{C}^1$ way and not only $\mathcal{C}^0$.

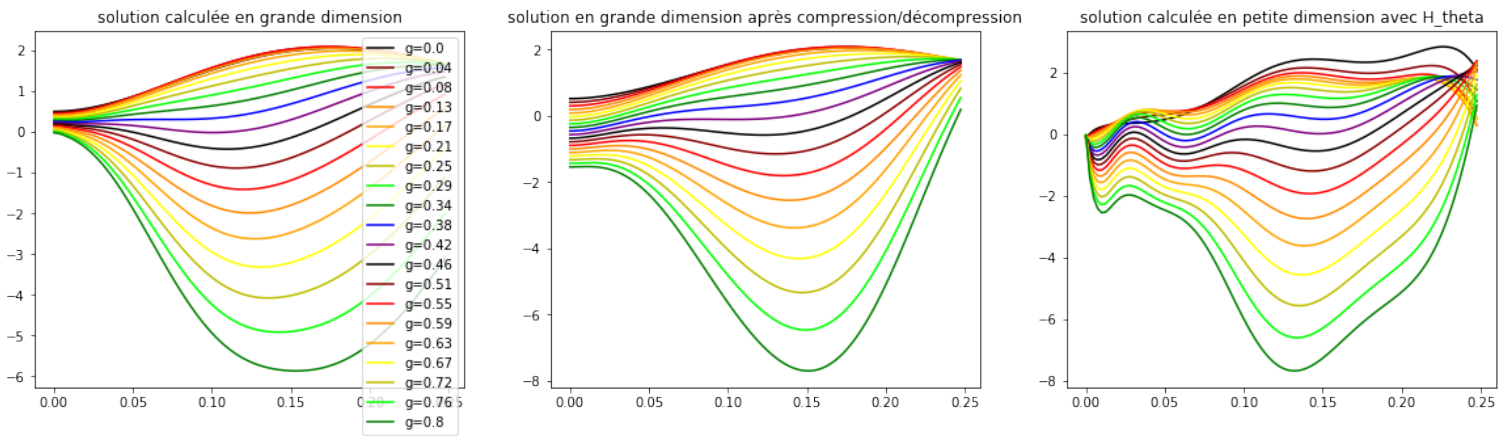


Figure 2.9: Variation of energie during time for different values of the parameter g. On the left are the variations of H for solutions computed in high dimension, on the middle the variations of $H \circ A$ for solutions computed in high dimension and projected in low dimension with A^+, on the right, the variations of H_θ for solutions computed in high dimension and projected in low dimension with A^+. The projection and the hyper-reduction are those of Figure 2.10. If the regression had produced the exact $H \circ A$, the two graphs on the right would have been the same.

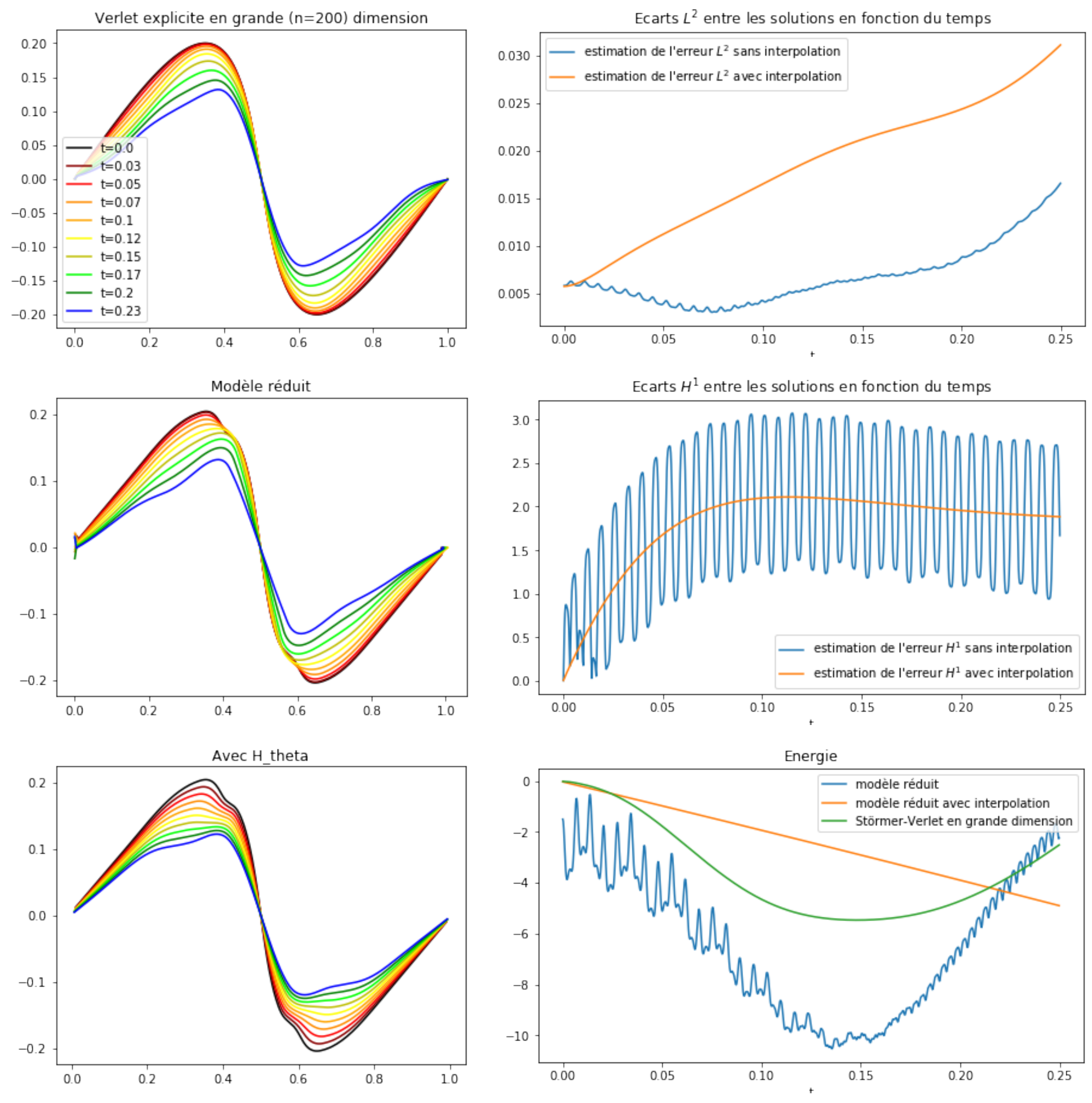


Figure 2.10: Numerical solution of piano string equation with $V = \frac{1}{2}((1-g)u^2 + v^2) + (\frac{g}{2}(u^2 v + \frac{1}{4}u^4)$, where $g = 0.783$. The first column represents the piano string position in the oscillation plane for different times computed in high dimention (top), in low dimension with $H \circ A$ (middle) and in low dimension with H_θ (bottom). The second column represents an estimation of the errors made on the string position through time on L^2 (top) and H^1 (middle) norms and the variation of energy durng time for the three different solutions (bottom). Blues curves represent the solution computed with the reduce model without hyper-regression, orange curves the solution computed wih the reduce model with hyper-regression and the green curve the solution in high dimension. The reduction was made using PSD with $k = 5$ using 20 trajectories, computed in high dimension with explicit Stormer-Verlet scheme, for values of g uniformly sampled in $I = [0, 0.8]$, $n = 200$, $dt = 0.0005$ and $m = 500$. Trajectories in low dimension were computed using the reduced model with the same discretisation in time and space and with $g = 0.783$. The construction of H_θ was made with the hyper-regression using Matern kernel with $\nu = 6.5$, $\sigma = 50$ and $\alpha = 0.001$.

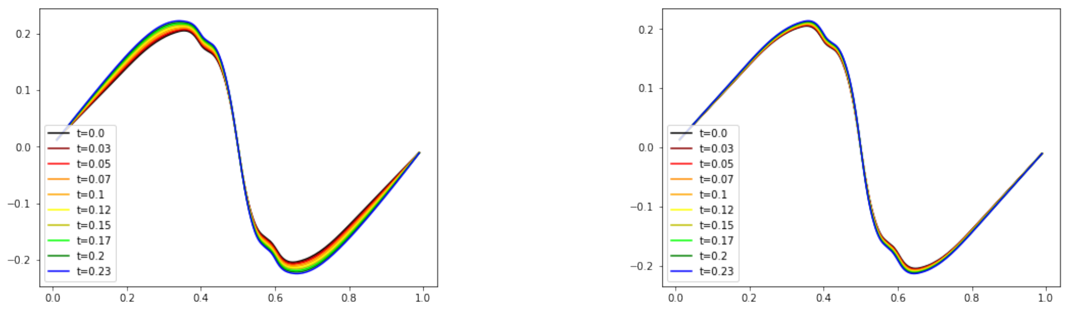


Figure 2.11: Numerical solution of piano string equation with $V = \frac{1}{2}((1-g)u^2 + v^2) + (\frac{g}{2}(u^2v + \frac{1}{4}u^4)$, where $g = 0.783$. Both pictures represent the piano string position in oscillation plane computed with reduce model and hyper-reduction built as in Figure 2.10. The parameter α of the hyper-reduction is 0.00099 (left) and 0.00101 (right).

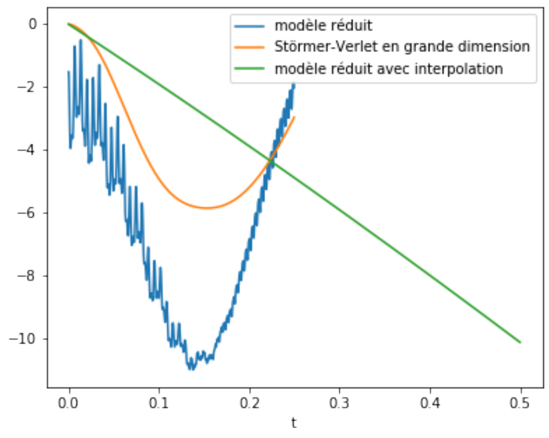


Figure 2.12: Variations of energy during time for solutions computed as in Figure 2.10. Solution of reduce model with hyper-reduction was computed for a longer time.

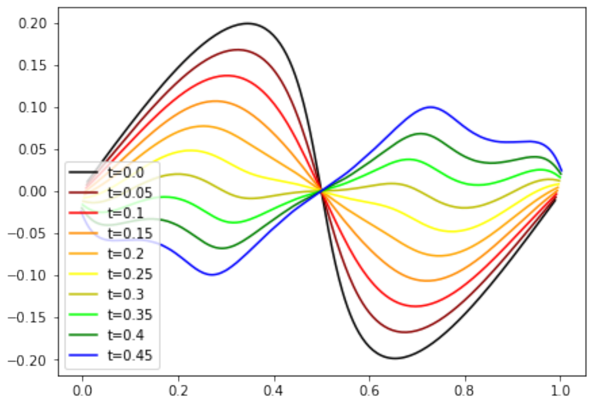


Figure 2.13: Numerical solution of piano string equation with $V = \frac{1}{2}((1-g)u^2 + v^2)$, where $g = 0.537$. The reduction was made with PSD with 20 trajectories and with the same parameters as in Figure 2.3. The hyper-reduction parameters were $\alpha = 0.00799$, $\nu = 4.5$ and $\sigma = 50$.

Conclusion

We have seen different methods to build reduced order model. The first one, the POD, fails when the equation is non-linear or has the particular form of a Hamiltonian equation. We have seen that combining a non-liear projection using geodesics distances or Laplacian eigenvectors with a kernel regression gives good results for Burgers' equation and allows to save time when computing the solution. When the problem is a linear Hamiltonian system, cotangent lift or greedy algorithm provides a good reduction with a noticeable gain of time during computation too.

In the case of non-linear Hamiltonian systems however, these two last methods do not work. As a symplectic structure is less constraining than a Riemannian one, we have less theoretical tools available to build relevant reduced order models respecting this structure. For these reasons, it seems to be more delicate to build a reduced order models for Hamiltonian systems. Nevertheless, hyper-regression may be an idea to develop.

To conclude, I think that the two first of the internship objectives were reached. The results of the tests were in line with the theorical expectations, so there were no unforeseen problems or changes of objectives during the intership. I could surely have been expected to code more quickly, especially the hyper-reduction, and this is certainly the reason why the last part is less developed than originally planed and the third objective is partiatially reached. It could have been interesting to try a non-linear reduction method for Hamiltonian systems.

During this intership, I worked on some skills I acquired during the year of M1. I used Python to implement methods I learned and this gave me the opportunity to discover Scikit-learn tools for manifold learning and kernel regression. From the numerical analysis point of view, I discovered new kinds of schemes to solve PDEs - symplectic schemes. I also worked on my English skills as almost all the references I had to read were in English.

I also developed new skills during this internship. In mathematics, I discovered a new way to deal with PDEs, namely the reduced order modelling. The part I found the most interesting in the internship was the theoretical one, during which I discovered symplectic geometry, which I would otherwise never have seen in the master's programme.

In the field of "soft" skills, I often had to take a step back and think about the global mechanisms rather than the technical details, but without losing sight of the geometric rigour which is the only way to properly explain the validity of the methods studied. Because I had a geometrician and two numerical analysts for supervisors, I sometimes had to switch from a point of view to another to understand what there were saying about the same subject. I found these exercises difficult and enriching and I still have a lot of room for improvement.

Bibliography

[1] B.M. Afkham and J.S. Hesthaven. Structure preserving model reduction of parametric hamiltonian systems. Preprint at https://arxiv.org/abs/1703.08345, 2017.

[2] V.I. Arnold. *Mathematical Methods of Classical Mechanics*, chapter 7, 8, 9, 10, pages 163–200. Springer, 1989.

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

[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] J. Chabassier and P. Joly. Energy preserving schemes for nonlinear hamiltonian systems of wave equations: Application to the vibrating piano string. *Computer Methods in Applied Mechanics and Engineering*, 199(45-48):2779–2795, 2010.

[7] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration illustrated by the Störmer-Verlet method.* Cambridge University Press, 2003.

[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] INRIA. Inria et son écosystème. url: https://www.inria.fr/fr/innovation-numerique-ecosysteme, november 2020. Accessed: 2022-08-20.

[10] P. Massot. *Topologie différentielle, chapitre 6, lecture notes, Ecole Polytechnique*, 2016.

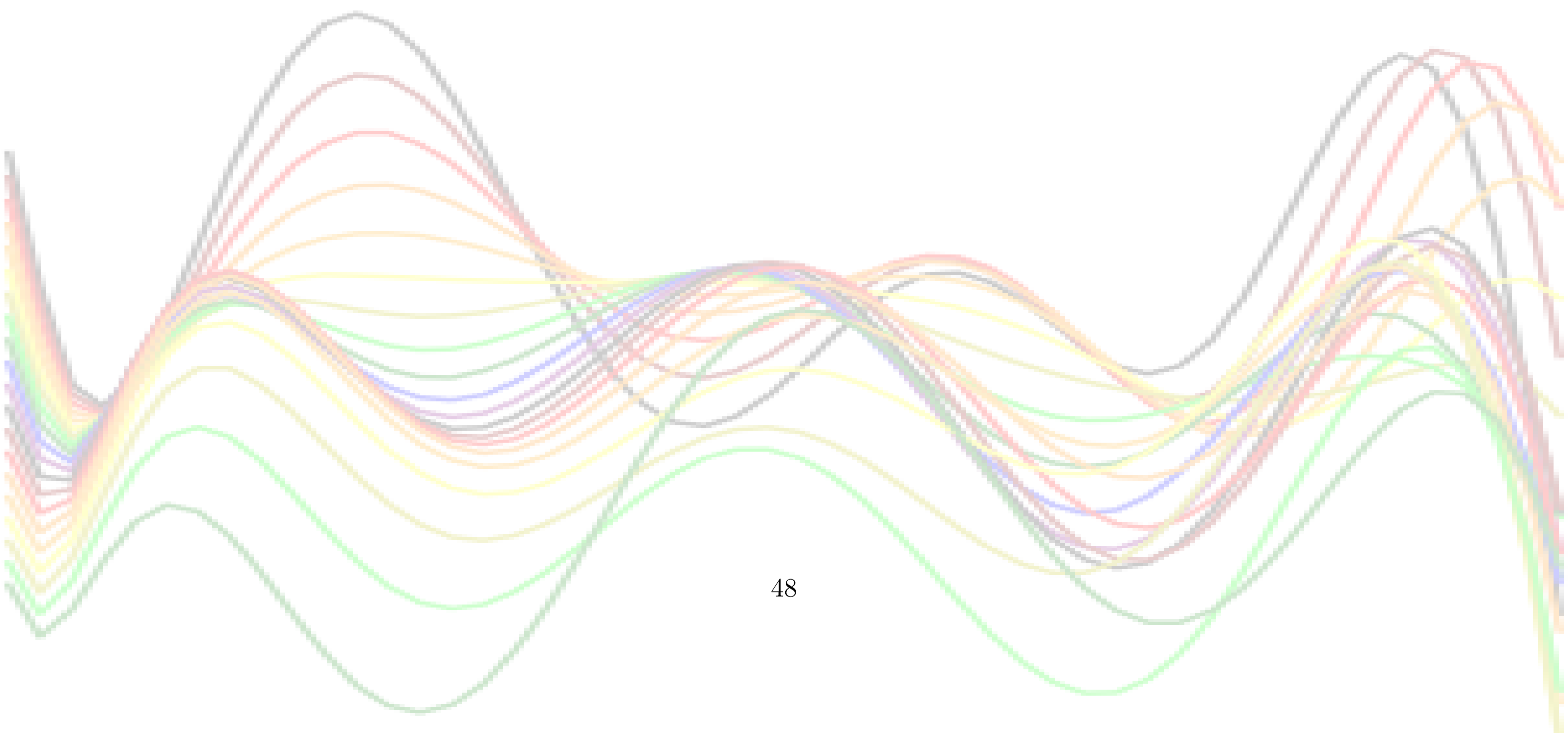
[11] E. Opshtein. *Formes différentielles, lecture notes, Université de Strasbourg*, 2022.

[12] L. Peng and K. Mohseni. Symplectic model reduction of hamiltonian systems. Preprint at https://arxiv.org/abs/1407.6118v2, 2015.

[13] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*, chapter Covariance Functions, pages 84–85. MIT Press, 2006.

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

Appendices



Here, we gives Python implementation we used to test schemes and methods used in the report.

.1 Störmer-Verlet

Here are the implicit version of the algorithm and the explicit one, for the particular case where H is separated.

```python
def Stormer_Verlet_explicit(DpH, DqH, v0, n, m, dt) :
    '''DpH, DqH : functions, gradients of H by respect to p and q
                  (it depend only on X)
    v0 : vector of size n, initial solution
    g : parameters of the Hamiltonian
    n : integer,  number of points in spatial discretization
    m : integer, number of points in temporal discretization
    dt : float, time step'''

    #initialisation
    X = np.zeros((2*n,m))
    X[:,0] = v0
    demi = np.zeros((n))

    #boucle en temps
    for i in range(1,m) :
        demi[:] = X[n:,i-1] + (dt/2)*DpH(X[:n,i-1])
        X[:n,i] = X[:n,i-1] - dt*DqH(demi)
        X[n:,i] = demi[:] + (dt/2)*DpH(X[:n,i])

        '''dual version :
        demi[:] = X[:n,i-1] - (dt/2)*DqH(X[n:,i-1])
        X[n:,i] = X[n:,i-1] + dt*DpH(demi)
        X[:n,i] = P_2[:] - (dt/2)*DqH(X[n:,i])'''

    return X
```

```python
def Stormer_Verlet_implicit(DpH, DqH, v0, n, m, dt) :
    '''DpH, DqH : functions, H gradient by respect to p and q
    v0 : vector of size n, initial solution
    g : parameters of the Hamiltonian
    n : integer,  number of points in spatial discretization
    m : integer, number of points in temporal discretization
    dt : float, time step'''

    #initialisation
    X = np.zeros((2*n,m))
    X[:,0] = v0
    demi = np.zeros((n))

    #time loop
    for i in range(1,m) :
        Fq = lambda q: X[n:,i-1]+(dt/2)*DpH(X[:n,i-1],q) -q
        demi[:] = sp.optimize.newton(Fq, X[n:,i-1]+(dt/2)*DpH(X[:n,i-1],X[n:,i-1]),
                                     maxiter=1000, tol=1e-3, disp=False)

        Fp = lambda p: X[:n,i-1]-(dt/2)*( DqH(X[:n,i-1],demi) + DqH(p,demi) ) -p
        X[:n,i] = sp.optimize.newton(Fp, X[:n,i-1]-dt*DqH(X[:n,i-1],demi),
                                     maxiter=1000, tol=1e-3, disp=False)

        X[n:,i] = demi[:] + (dt/2)*DpH(X[:n,i],demi)

    return X
```

.2 Reduced order models

We give here the cotangent lift and the greedy algorithm implementations.

```python
def J(n) :
    '''returns the matrix of the standard symplectic matrix for the standard
       scalar product in R^2n'''
    return np.diag(np.ones((n)), k=n) + np.diag(-np.ones((n)), k=-n)
```

```python
def greedy(H, DpH, DqH, z0, n, k, F, m, G, dt, a, dx, N=1) :
    '''H : function, Hamiltonian function of the high dimensional system
    DpH : function, gradient of the previous by respect to p
    DqH : function, gradient of H by respect to q
    z0 : function, initial condition of the high dimensional system, depends on
         the parameters
    n : integer, initial (high) dimension (=number of points in spatial
        discretisation)
    k : integer, 0.5*size of the reduced model
    F : function, model to compute solutions in high dimensional space
    m : integer, number of snapshots taken at each iteration
    G : array of size (d,l) with l the number of parameters and d the number of
     combinaison, parameters grid
    dt : float, time step when taking snapshots
    a : float, lower bound of spatial interval considered
    dx : float, space step
    plot : boolean, if, or not, the solution should be plot after computation
    err : maximal error between the Hamiltonian functions
    A : reduced orthosymplectic basis (size : (n,k))
    Ap : its symplectic inverse'''

    #discretization
    x = np.linspace(a,a+dx*n,n)

    #useful constants
    J_2n = J(N*n)
    v0 = z0(x) #de taille N*n : par ex, pour N=2 (p_1, p_2, q_1, q_2)
    H0 = H(v0,G)

    #initialisation
    g = G[0]
    v = v0.copy()
    v /= np.linalg.norm(v)
    J_2k = J(1)
    A = np.stack([v,J_2n.T@v]).T
    Ap = J_2k.T@A.T@J_2n
    stock = np.zeros((2*N*n,2*k*m))

    for i in range(1,k) :
        #choice of the worst parameter
        dH = np.abs(H0-H(A@Ap@v0,G))
        g = G[np.argmax(dH)]

        #computation of the solution in high dimension for this parameter
        S = F(lambda p : DpH(p,g), lambda q : DqH(q,g), v0, n*N, m, dt)
        stock[:,i*m:(i+1)*m] = S[:,:]

        #choice of the worst projected snapshot
        dP = np.linalg.norm(S-A@Ap@S, axis=0)
        s = S[:,np.argmax(dP)]

        #computation of the two new vectors in the basis using symplectic
        Gramm-Schmidt process
        v = s - A@(s.T@A).T
        v /= np.linalg.norm(v)
        J_2k = J(i+1)
        A = np.concatenate([A[:,:i], v[:,np.newaxis], A[:,i:],
                            (J_2n.T@v)[:,np.newaxis]], axis=1)
        Ap = J_2k.T@A.T@J_2n

    #computation of final error on the Hamiltonian
    dH = np.abs(H0-H(A@Ap@v0,G))
    err = max(dH)

    return err, A, Ap, stock
```

```python
def cotlift(DpH, DqH, g_list, v0, n, k, m, dt, a, dx) :

    l = len(g_list)
    X = np.empty((2*n+1,l*m))

    for i in range(l):
        g = g_list[i]
        X[:-1,i*m:(i+1)*m] = Stormer_Verlet(lambda p : DpH(p,g),
                                            lambda q : DqH(q,g),
                                            v0, n, m, dt, True)
        X[-1,i*m:(i+1)*m] = g

    U, S, V = np.linalg.svd(np.concatenate([X[:n,:],X[n:-1]], axis=1))
    A = U[:,:k]

    return A, X
```

.3 Hyper-reduction

Here are the functions that compute the optimal coefficient θ for the hyper-reduction, the resulting energy function and its gradient.

```python
def theta(X, H_vec, alpha, nu, sigma, dt, R) :
    '''X : snapshots with corresponding parameter (in low dimension 2k) :
           array of size (2k+1, l=nbr of snapshots),
    H_vec : value of H \circ A at this snapshots : vector of size l,
    alpha : proportion of the loss which controls the gradient of H_theta :
            float between 0 and 1,
    nu : parameter of the Matern kernel : positive float,
    sigma : kernel variance,
    dt : time step used for snapshots computation,
    R : nbr of trajectoiries in X.'''

    (k,l) = X.shape
    k = (k-1)//2
    m = l//R

    JD = J(k).T @ (X[:-1,1:]-X[:-1,:-1]) /dt
    kv = Matern(length_scale=sigma, length_scale_bounds='fixed', nu=nu)
    K = kv(X.T)
    C = np.zeros((l,l))
    c = np.zeros((l))

    for r in range(R) :
        for s in range(m-1) :
            i = r*m + s

            Dx = X[:-1,i][:,np.newaxis] - X[:-1,:]
            nDx = np.sqrt( np.sum((X[:-1,i][:,np.newaxis] - X[:-1,:])**2, axis=0)
                          + (X[-1,i]*np.ones((X.shape[1]))-X[-1,:])**2 )
            c1 = np.sqrt(2*nu) * nDx / sigma
            c2 = 1 / (spsp.gamma(nu) * (2**(nu-1)))
            dxKi = c2 * c1**(nu-2) * (np.sqrt(2*nu)/sigma)**2 * Dx *
                   (nu*spsp.kv(c1,nu) + c1*spsp.kvp(c1,nu))

            C += dxKi.T@dxKi
            c += dxKi.T@JD[:,i]

    B = alpha*C + (1-alpha)*(K.T@K)
    b = alpha*c + (1-alpha)*(K.T@H_vec)

    return K, C, c, np.linalg.solve(B,b)
```

```python
def H_theta(X, x, nu, sigma, theta) :
    '''X : trajectories used to compute theta :
           array of size (2k+1, nbr de snapshots),
    x : snapshots at which we want the energy :
        array of size (2k+1, nbr of snapshots,
     each snapshot is of the form (p,q,g),
    nu : Matern kernel parameter,
    sigma : Matern kernel variance,
    theta : optimal coefficients for H_theta.'''

    k = Matern(nu=nu, length_scale_bounds='fixed', length_scale=sigma)
    K = k(X.T, x)
    return K.T@theta
```

```
def dH_theta(X, x, nu, sigma, theta, g) :
    '''X : trajectories used to compute theta :
           array of size (2k+1, nbr de snapshots),
    x : snapshots at which we want the energy :
        array of size (2k, nbr of snapshots),
    each snapshot is of the form (p,q),
    nu : Matern kernel parameter,
    sigma : Matern kernel variance,
    theta : optimal coefficients for H_theta,
    g : parameter of H used during resolution.'''

    Dx = x[:,np.newaxis] - X[:-1,:]
    nDx = np.sqrt( np.sum((x[:,np.newaxis] - X[:-1,:])**2, axis=0)
          + (g*np.ones((X.shape[1]))-X[-1,:])**2 )
    c1 = np.sqrt(2*nu) * nDx / sigma
    c2 = 1 / (spsp.gamma(nu) * (2**(nu-1)) * nDx)
    dxK = c2 * (c1**nu) * Dx * (nu*spsp.kv(c1,nu) + c1*spsp.kvp(c1,nu))
    return dxK@theta
```