Thèse

INSTITUT DE RECHERCHE MATHÉMATIQUE AVANCÉE

UMR 7501

Strasbourg

présentée pour obtenir le grade de docteur de l'Université de Strasbourg
Spécialité MATHÉMATIQUES APPLIQUÉES

Guillaume Steimer

Méthode de réduction d'ordre pour des dynamiques hamiltoniennes utilisant l'apprentissage profond

Soutenue le 12 septembre 2025
devant la commission d'examen

Raphaël Côte, directeur de thèse
Virginie Ehrlacher, rapporteure
Tommaso Taddei, rapporteur
Stéphane Lanteri, examinateur
Nicolas Crouseilles, examinateur
Christophe, Prud'homme, examinateur
Emmanuel Franck, co-encadrant
Laurent Navoret, co-encadrant
Vincent Vigon, co-encadrant

https://irma.math.unistra.fr

cnrs

IRMA
Institut de Recherche Mathématique Avancée

Université de Strasbourg

Remerciements

Au terme de ce travail de thèse, le moment des remerciements est pour moi un instant tout particulier. Il me permet de prendre du recul et de mesurer à quel point ce périple, à la fois scientifique et humain, n'aurait été possible sans le soutien et la bienveillance de nombreuses personnes. Ces remerciements vous sont dédiés.

Je tiens tout d'abord à exprimer ma profonde gratitude à mon directeur de thèse, Raphaël Côte, ainsi qu'à mes co-encadrants Emmanuel Franck, Laurent Navoret et Vincent Vigon, pour leur accompagnement, leurs conseils avisés et leur disponibilité tout au long de ces années. Merci pour votre confiance et pour tout le temps que vous m'avez consacré. Ces remerciements ne concernent pas seulement votre suivi scientifique, à travers les nombreuses et enrichissantes réunions que nous avons eues autour de nos travaux de recherche et les multiples relectures attentives de mes écrits mathématiques, mais aussi votre accompagnement humain, votre soutien et votre écoute, qui ont beaucoup compté pour moi.

Je remercie également Virginie Ehrlacher et Tommaso Taddei d'avoir accepté de rapporter ma thèse, ainsi que les membres du jury Stéphane Lanteri, Nicolas Crouseilles — avec qui j'ai eu de nombreuses discussions autour de la dynamique des plasmas — et Christophe Prud'homme, dont la présence revêt une signification particulière puisqu'il est également le directeur du Master qui a marqué mes premiers pas dans le domaine de la recherche.

Je salue aussi les membres des équipes TONUS et MACARON pour tous ces séminaires qui m'ont permis de découvrir plus en détail l'analyse numérique, les EDP ou encore le machine learning scientifique. Je pense en particulier à Victor Michel-Dansac, pour sa bonne humeur, sa gentillesse et nos nombreux échanges scientifiques. Je souhaite également remercier le personnel administratif de l'IRMA, de l'UFR, de l'ED MSII et de l'INRIA, notamment Ouiza Herbi, qui a fait preuve d'une grande gentillesse et d'une grande patience lors de la saisie de mes missions pour diverses conférences.

Ces années de doctorat n'auraient pas eu la même saveur sans mes camarades — amis doctorants et anciens doctorants de l'IRMA. Je vous remercie d'avoir rendu ces années plus agréables, que ce soit lors des séminaires doctorants, à l'occasion d'un café, d'un goûter ou, soyons honnêtes, autour d'une bière au Comptoir puis au Télégraphe. J'aimerais saluer nommément Thomas et Tom, qui me supportent depuis le master, ainsi que Romane, Brieuc et Léopold pour nos escapades marseillaises du CEMRACS 2022, sans oublier Adam, Roméo, Killian, Frédérique, Jean-Pierre, Clarence, Clément, Victoria, Céline, Thomas avec des lunettes, Paul, Ludovic, Mickaël, Léo, Claire, Roxana, Vincent, Robin, Colin, Nicolas, Victor, Florian, Thibaut, Yohann, et al.

J'exprime aussi un remerciement spécial à mes amis les plus proches, les «belettes », ces petites créatures rusées et agiles qui ont toujours su apporter leur soutien et leur sourire, même dans les moments difficiles. J'ai une pensée particulière pour Océane, mon fidèle acolyte de course à pied et de musculation, et pour Daphné, qui sont elles aussi en train de terminer leur thèse de doctorat. En poursuivant dans la famille des mustélidés, je remercie chaleureusement Claudia, Tom, Rémi et Louis, mais aussi Alex et Dimitri, piliers en amitié depuis notre première

année de licence en MPA. Je tiens également à remercier Marie et Cédric pour leur amitié et leur soutien précieux depuis longtemps.

À ma fiancée, Céline, merci du fond du cœur. Ta bienveillance, ta force, ton soutien indéfectible, tes encouragements et ta présence constante m'ont porté plus loin que je ne saurais le dire. Merci d'avoir survécu à ces quatre années de thèse, à ces semaines de travail interminables et à mes humeurs de problème mal posé. Je te promets de redevenir rapidement un fiancé normal ! Merci d'avoir cru en moi sans relâche. Tu es mon roc, et cette thèse est aussi un peu la tienne.

Merci à ma famille : sans les racines solides que vous m'avez données depuis mon plus jeune âge, rien n'aurait été possible. Un immense merci à ma mère, qui m'a toujours soutenu et rassuré dans mes études et au-delà, à mon frère, pilier discret et solide, une présence sur laquelle je peux toujours compter, et à ma grand-mère, dont la bienveillance et la sagesse tranquille m'ont toujours accompagné. Merci aussi à mon père, parti trop tôt, mais dont la mémoire, les valeurs et la force continuent de m'accompagner chaque jour. Papa, tu n'es plus là, mais je t'ai senti à mes côtés tout au long du chemin. Enfin, pour reprendre les mots de Monsieur Fernand dans *Les Tontons Flingueurs* : «Hé ben dis donc ! C'est du brutal ! » — une formule qui résume assez bien ces quelques années de thèse.

Contents

1 Introduction **7**
- 1.1 Contexte & problématique . . . 7
- 1.2 Description du manuscrit et contributions . . . 9
- References . . . 12

2 Hamiltonian systems: numerical methods, reduction and deep learning tools **13**
- 2.1 A primer on Hamiltonian dynamics . . . 14
- 2.2 Hamiltonian systems & numerical methods . . . 19
  - 2.2.1 A formal introduction to Hamiltonian PDEs . . . 19
  - 2.2.2 Hamiltonian ODEs: definition & properties . . . 22
  - 2.2.3 Numerical methods for time integration . . . 27
- 2.3 Linear model order reduction for Hamiltonian ODEs . . . 30
  - 2.3.1 Proper Symplectic Decomposition . . . 31
  - 2.3.2 Building the projection matrix . . . 33
  - 2.3.3 Hyper-reduction with symplectic DEIM . . . 35
  - 2.3.4 A practical example: the shallow-water system . . . 37
  - 2.3.5 Other reduction methods . . . 42
- 2.4 Deep learning tools for Hamiltonian model order reduction . . . 43
  - 2.4.1 Outline of neural networks . . . 43
  - 2.4.2 Learning symplectic flows with Hamiltonian Neural Networks . . . 45
  - 2.4.3 Convolutional AutoEncoder for low-dimensional representations . . . 46
- References . . . 48

3 Hamiltonian reduction using a convolutional autoencoder coupled to an Hamiltonian neural network **52**
- 3.1 Introduction . . . 53
- 3.2 Parameterized Hamiltonian systems and reduction . . . 55
  - 3.2.1 Parameterized Hamiltonian dynamics . . . 55
  - 3.2.2 Hamiltonian reduced order modeling . . . 57
- 3.3 A nonlinear Hamiltonian reduction method . . . 58
  - 3.3.1 Reduction with an Auto Encoder (AE) . . . 58
  - 3.3.2 Reduced model with a Hamiltonian Neural Network (HNN) . . . 59
  - 3.3.3 Strong coupling of the neural networks . . . 61
  - 3.3.4 Training hyper-parameters . . . 61
  - 3.3.5 Numerical complexity . . . 63
- 3.4 Numerical results . . . 64
  - 3.4.1 Wave equations . . . 64
  - 3.4.2 1D shallow water system . . . 72
  - 3.4.3 2D shallow water system . . . 77

3.5 Conclusion . . . 83
References . . . 84

4 Reduced Particle in Cell method for the Vlasov-Poisson system using autoencoder and Hamiltonian neural networks 87
4.1 Introduction . . . 88
4.2 Particle discretization of the Vlasov-Poisson equation . . . 91
4.2.1 The Vlasov-Poisson equation . . . 91
4.2.2 Hamiltonian particle-based discretization . . . 91
4.2.3 Time discretization and initialization . . . 93
4.3 A Hamiltonian reduction with Proper Symplectic Decompostion prereduction . 94
4.3.1 PSD-AE-HNN reduction method . . . 95
4.3.2 PSD reduction . . . 97
4.3.3 AE-HNN reduction . . . 98
4.3.4 Hyperparameters tuning . . . 99
4.4 Numerical results . . . 101
4.4.1 Linear Landau damping . . . 102
4.4.2 Nonlinear Landau damping . . . 105
4.4.3 Two-stream instability . . . 111
4.4.4 Computation gain . . . 114
4.5 Conclusion . . . 118
References . . . 119

5 Hyperbolic reduced model for Vlasov-Poisson equation with Fokker-Planck collision 122
5.1 Introduction . . . 123
5.2 Reduced model in velocity for 1D Vlasov-Poisson-Fokker-Planck . . . 124
5.2.1 Vlasov-Poisson-Fokker-Planck model . . . 124
5.2.2 Semi-discretized model in velocity . . . 125
5.2.3 Reduced model . . . 127
5.3 Hyper-reduction and corrections of the reduced collision operator . . . 128
5.3.1 DEIM hyper-reduction . . . 128
5.3.2 Preservation of reduced moments . . . 128
5.3.3 Reduced Maxwellian distributions . . . 129
5.3.4 Reduced moment equations . . . 130
5.4 Numerical results . . . 131
5.4.1 Reduction for given parameters . . . 131
5.4.2 Preservation of Maxwellian distributions . . . 133
5.4.3 Generalization to other parameters . . . 134
5.5 Conclusion . . . 136
References . . . 136
A Numerical discretization details . . . 137
A.1 Discretization of the full model . . . 137
A.2 Discretization of the reduced model . . . 138
B Solution to the minimization problem . . . 138

6 Conclusion & perspectives 139
References . . . 141

Chapitre 1

Introduction

1.1 Contexte & problématique

Les systèmes rencontrés en sciences et en ingénierie – de la mécanique céleste à la dynamique moléculaire – sont régis par des lois physiques fondamentales qui décrivent leur évolution au cours du temps. Dans de nombreux cas, ces lois impliquent la conservation de certaines grandeurs telles que la masse, la charge ou le moment angulaire. Notre analyse se concentre plus particulièrement sur le cas, souvent rencontré, où l'énergie est conservée. Ces systèmes dits conservatifs sont naturellement décrits dans le cadre formel hamiltonien, nommé d'après le célèbre physicien Sir William R. Hamilton.

Les systèmes hamiltoniens constituent une classe de modèles puissants et efficaces, apparaissant naturellement dans de nombreux domaines : mécanique céleste, dynamique moléculaire, physique quantique, dynamique des plasmas, théorie du contrôle, etc. Ces systèmes décrivent comment l'énergie totale, répartie entre énergie cinétique et énergie potentielle, guide l'évolution de l'état du système. La notion de système hamiltonien est centrale non seulement en raison de son omniprésence dans la nature, mais aussi parce qu'elle encode des lois fondamentales de conservation, profondément liées à la structure géométrique de ces systèmes d'une manière remarquablement élégante.

Par exemple, dans le cas de l'équation des ondes, la structure hamiltonienne sous-tend le comportement d'une corde ou d'une membrane qui s'étire ou se plie sous l'effet de forces. Le modèle de Saint-Venant (dynamique des eaux peu profondes, shallow water) décrit l'évolution de la surface libre d'un fluide sous l'influence de la gravité, illustrant notamment le mouvement des vagues dans l'océan. Les systèmes hamiltoniens sont également omniprésents en physique des plasmas, où ils gouvernent le mouvement rapide des particules chargées dans des champs électromagnétiques jusqu'à la propagation des ondes de plasma.

Bien que le formalisme hamiltonien ne nécessite essentiellement que la connaissance de l'énergie totale du système et de sa structure pour prédire son comportement complet, l'utilisation pratique de tels modèles est souvent limitée par leur taille et leur complexité. En effet, lorsqu'on modélise un système physique complexe, on commence généralement par formuler des Équations aux Dérivées Partielles (EDP). Ces équations sont ensuite discrétisées à l'aide de techniques telles que les éléments finis, les différences finies, les discrétisations spectrales ou les méthodes de type Particle-In-Cell (PIC), conduisant à des Équations Différentielles Ordinaires (EDO) souvent de très grande dimension ou de nombreux degrés de liberté, et potentiellement des non-linéarités.

Ces modèles complets (Full Order Models, FOMs) capturent des dynamiques très précises, mais au prix d'un coût de simulation numérique / computationnel élevé. L'exécution d'une seule simulation à haute résolution peut nécessiter plusieurs heures de calcul, voire davantage, même

sur un cluster de calcul haute performance. Dans les domaines du contrôle, de l'optimisation de la conception et de la quantification d'incertitudes, il est souvent nécessaire d'effectuer de nombreuses simulations de haute fidélité pour différents jeux de paramètres. Cela devient alors difficilement réalisable, en particulier lorsque le temps ou la puissance de calcul sont limités, par exemple dans des applications temps réel ou embarquées. Ces difficultés ne se limitent pas au processus de simulation lui-même : elles compliquent également l'analyse, les études de sensibilité ou tout processus décisionnel s'appuyant sur de tels modèles. Cela conduit à une demande croissante pour des représentations réduites beaucoup moins coûteuses à évaluer, tout en conservant les dynamiques essentielles.

Dans ce contexte, la réduction de modèle (Model Order Reduction, MOR) offre une solution particulièrement intéressante. L'idée principale de la MOR est relativement claire : trouver un modèle beaucoup plus petit qui imite le comportement du modèle complet. Ce modèle réduit doit être capable de prédire la dynamique du modèle complet sous différents paramètres tout en préservant des caractéristiques essentielles telles que la précision, la stabilité et la fidélité physique. En pratique, la MOR isole les motifs de faible dimension présents dans la dynamique et filtre le reste. Par exemple, lorsqu'on cherche à résoudre la propagation d'ondes dans un milieu sur une grille très fine afin d'obtenir des solutions numériques précises, cela conduit à la résolution d'un système d'EDO de très grande dimension sur ordinateur. La réduction de modèle permet d'identifier que la partie la plus significative de la dynamique peut être gouvernée par quelques modes de Fourier dominants.

Cependant, une approximation satisfaisante de la dynamique du système ne suffit pas dans le cas des systèmes hamiltoniens. Les techniques de réduction doivent également préserver la structure hamiltonienne dans le modèle réduit. Un modèle réduit qui ne conserve pas la structure hamiltonienne, notamment la préservation de l'énergie, conduit la plupart du temps à des instabilités numériques et à des états non physiques lors de simulations à long terme. Or les approches classiques de réduction détruisent souvent cette structure. Pour remédier à ce problème, on met en œuvre des techniques de réduction de modèle préservant la structure. À titre d'exemple, la décomposition symplectique aux valeurs propres (Proper Symplectic Decomposition, PSD) projette le système complet sur un sous-espace vectoriel de faible dimension tout en conservant la structure hamiltonienne grâce à une base de vecteurs adaptée.

Parallèlement au développement de la réduction hamiltonienne de modèle, l'apprentissage automatique (machine learning, ML) s'est imposé comme un outil puissant pour enrichir l'analyse numérique. De nombreuses approches hybrides combinant analyse numérique classique et techniques d'apprentissage ont émergé, avec dans certains cas des résultats prometteurs. L'apprentissage automatique propose une approche fondée sur les données pour découvrir des motifs et des structures, permettant parfois de construire des modèles approchés quand les formules analytiques explicites sont difficiles à obtenir ou bien que les coûts de calculs sont trop élevés.

Néanmoins, les approches standards d'apprentissage automatique ne respectent pas la structure hamiltonienne des modèles considérés. Cela a conduit à un intérêt croissant pour les méthodes d'apprentissage informé par la physique et de préservation de la structure (structure preserving machine learning) au sein de la communauté scientifique. Par exemple, les réseaux de neurones informés par la physique (Physics-Informed Neural Networks, PINNs) intègrent directement les lois physiques dans l'entraînement du réseau de neurones (Neural Networks, NNs) afin de prédire les états du modèle complet, tandis que les réseaux de neurones hamiltoniens (Hamiltonian Neural Networks, HNNs) sont capables d'apprendre directement l'énergie totale d'un système à partir des données.

À la lumière de l'engouement récent pour le développement de techniques de réduction préservant la structure, combinant méthodes classiques de réduction et apprentissage automatique, cette thèse vise à développer de nouvelles méthodes de *réduction de modèle hamiltonienne, assistées par apprentissage profond* (deep learning), combinant réseaux de neurones, techniques de préservation de la structure et réduction de modèle. Les méthodes développées dans ce manuscrit sont testées sur des équations de type ondulatoire, en dynamique des fluides (Saint-Venant), ainsi qu'en dynamique des plasmas avec l'équation de Vlasov-Poisson, par exemple.

1.2 Description du manuscrit et contributions

Dans ce manuscrit, nous présentons le développement de nouvelles méthodes de réduction hamiltonienne, assistées par apprentissage profond. Il est structuré en cinq chapitres. Le premier chapitre est une introduction fournissant une mise en contexte ainsi qu'une présentation du manuscrit en français. Le deuxième chapitre fait office de boîte à outils présentant les notions et méthodes utilisées dans les trois chapitres suivants. Chacun de ces chapitres correspond à un article — l'un est déjà publié, le suivant est proche d'une soumission pour publication, et le dernier est une communication (proceedings) de l'école d'été du CEMRACS 2022. Ils sont présentés sous la forme d'articles scientifiques et peuvent être lus indépendamment les uns des autres. Ils sont suivis d'une conclusion générale ainsi que des perspectives. Présentons plus en détail le contenu des chapitres un à cinq.

Chapitre 1 - Introduction

Ce premier chapitre introduit les concepts de systèmes hamiltoniens, d'apprentissage automatique et de réduction de modèle, afin de poser le cadre du sujet traité dans cette thèse. Il se conclut par une présentation générale du contenu des chapitres du manuscrit.

Chapitre 2 - Hamiltonian systems: numerical methods, reduction and deep learning tools

Le second chapitre regroupe l'ensemble des ressources théoriques et méthodologiques nécessaires à ce manuscrit. Il aborde les questions clés ainsi que les défis inhérents au développement de modèles réduits hamiltoniens, fondés sur l'apprentissage profond. Nous introduisons dans un premier temps quelques notions de base afin de présenter les systèmes hamiltoniens à travers des exemples d'EDO simples. En pratique, l'intérêt se porte sur des EDO de grande dimension issues de la semi-discrétisation d'EDP hamiltoniennes. Ainsi, nous définissons d'abord les modèles hamiltoniens sous la forme d'EDP, avant de procéder à leur semi-discrétisation en EDO hamiltoniennes, pour enfin aboutir à des modèles numériques entièrement discrets, comme détaillé dans les Sec. 2.2.1 et Sec. 2.2.2. Le principal défi réside dans la préservation de la structure hamiltonienne à chaque étape du processus, ainsi que dans la compréhension des implications de cette préservation en termes de difficultés techniques et de coût de calcul. Ensuite, pour répondre à la problématique du coût numérique élevé et du temps de calcul important des simulations de modèles complets, nous introduisons les principes fondamentaux de la réduction de modèle dans la Sec. 2.3. L'hypothèse principale de travail est que l'espace des solutions peut être décrit par un sous-espace linéaire. En utilisant des projections symplectiques, nous dérivons un premier modèle réduit. Toutefois, en raison de l'hypothèse de linéarité, ce modèle réduit rencontre des obstacles pour réduire efficacement la complexité de calcul dans les problèmes non-linéaires, bien que plusieurs techniques existent pour atténuer ce coût. En conséquence, nous introduisons les outils d'apprentissage profond nécessaires pour la réduction hamiltonienne non-linéaire dans la

Sec. 2.4, à savoir les réseaux de neurones. Le couplage entre réduction hamiltonienne et réseaux de neurones est réalisé dans les chapitres suivants.

Chapitre 3 - Hamiltonian reduction using a convolutional autoencoder coupled to an Hamiltonian neural network

Ce chapitre correspond à un premier article [1]. Nous considérons une famille d'EDO hamiltoniennes paramétrées et proposons une approche de réduction basée sur des auto-encodeurs convolutifs (Convolution AutoEncoder, AE) et des réseaux de neurones hamiltoniens (Hamiltonian Neural Network, HNN). Les auto-encodeurs sont utilisés pour apprendre des opérateurs d'encodage et de décodage, tandis que le réseau de neurones hamiltonien est employé pour apprendre la dynamique réduite de manière à préserver la structure hamiltonienne. Le processus d'entraînement est couplé, ce qui signifie que les trois composantes — l'encodeur, le décodeur et la dynamique réduite — sont apprises simultanément. L'utilisation du réseau de neurones hamiltonien garantit que la dynamique réduite conserve la structure hamiltonienne sous-jacente.

Cette approche repose sur trois éléments clés :

(i) l'auto-encodeur apprend une projection non-linéaire adaptée aux équations considérées entre les espaces complet et réduit, offrant de meilleures performances que les techniques de projection linéaire classiques;

(ii) l'apprentissage conjoint avec un réseau de neurones hamiltonien compense l'absence de contrainte symplectique sur l'auto-encodeur, en construisant un modèle réduit qui est hamiltonien par construction et proche du modèle complet grâce au processus d'entraînement;

(iii) l'utilisation de réseaux de neurones permet une forte parallélisation sur GPU, ce qui accélère considérablement l'évaluation numérique du modèle réduit.

Nous présentons plusieurs cas tests pour illustrer les bonnes propriétés de réduction obtenues. Deux ensembles d'équations sont considérés. Nous examinons d'abord une équation d'onde 1D non-linéaire paramétrée

$$\partial_{tt} v(x,t;\mu) - \mu_a \partial_x \left[w' \left(\partial_x v(x,t;\mu), \mu_b \right) \right] + g'(v(x,t;\mu), \mu_c) = 0, \tag{1.1}$$

sur un domaine périodique de longueur $L > 0$, avec une solution $v : \mathbb{R}/(L\mathbb{Z}) \times [0,T] \times \Gamma \to \mathbb{R}$ représentant le déplacement orthogonal à une corde vibrante, et $\mu = (\mu_a \ \mu_b \ \mu_c)^T \in \Gamma \subset \mathbb{R}^3$ désigne des paramètres; w et g sont des fonctions données. Le cas linéaire est également traité, il est obtenu avec $w(x) = \frac{1}{2}x^2$ et $g(x) = 0$. Ensuite, nous testons notre méthode sur les équations de Saint-Venant 1D et 2D sans bathymétrie, sur un domaine carré et périodique de longueur $L > 0$. En 2D, elles s'écrivent

$$\begin{cases} \partial_t \chi(\mathbf{x},t;\mu) + \nabla \cdot \left((1 + \chi(\mathbf{x},t;\mu)) \nabla \phi(\mathbf{x},t;\mu) \right) = 0, \\ \partial_t \phi(\mathbf{x},t;\mu) + \dfrac{1}{2} |\nabla \phi(\mathbf{x},t;\mu)|^2 + \chi(\mathbf{x},t;\mu) = 0, \end{cases} \tag{1.2}$$

où $\chi, \phi : \mathbb{R}^2/(L\mathbb{Z}^2) \times [0,T] \times \Gamma \to \mathbb{R}$ représentent respectivement la perturbation par rapport à l'équilibre et le potentiel scalaire. Dans ce cas test, la condition initiale est paramétrée par $\mu \in \Gamma \subset \mathbb{R}^2$ deux paramètres scalaires correspondant à l'amplitude et l'écart-type d'une gaussienne.

Chapitre 4 - Reduced Particle in Cell method for the Vlasov-Poisson system using autoencoder and Hamiltonian neural networks

Le Chap. 4 correspond à un article qui sera soumis pour publication dans un avenir proche. Il est consacré à l'équation de Vlasov-Poisson 1D-1V de la forme

$$\begin{cases} \partial_t f(t,x,v;\mu) + v\partial_x f(t,x,v;\mu) + \dfrac{q}{m} E(t,x;\mu)\partial_v f(t,x,v;\mu) = 0, \\ \partial_x E(t,x;\mu) = q \displaystyle\int f(t,x,v;\mu)\, dv - 1, \end{cases} \tag{1.3}$$

avec pour solution une distribution de particules $f : [0,T] \times \mathbb{R}/(2\pi\mathbb{Z}) \times \mathbb{R} \to \mathbb{R}$ de charge q et masse m. Elle est discrétisée par une méthode Particle-In-Cell (PIC) : f est approchée par une distribution discrète de particules numériques et le champ électrique $E(t,x;\mu)$ est résolu à l'aide d'un maillage du domaine spatial. L'une des principales difficultés réside dans le fait que le nombre de particules — dont dépend la dimension du modèle — est très élevé. De plus, la dynamique est hautement non-linéaire et multi-échelles en raison du champ électrique auto-induit, c'est-à-dire dépendant de la fonction de distribution f.

L'idée que nous présentons consiste à combiner les approches introduites dans l'article [1] et au Chap. 3 avec une première projection linéaire et symplectique. Cela aboutit à une architecture encodeur–décodeur à deux étages, couplée à une approximation de la dynamique réduite fondée sur des réseaux de neurones.

Précisément, notre méthode précédente ne peut pas être appliquée directement à ce problème, car les données en entrée sont constituées de particules : elles sont à la fois très nombreuses et, aussi, non structurées (non définies sur une grille régulière). Une projection symplectique linéaire préliminaire permet de réduire drastiquement la dimension du système tout en préservant les propriétés symplectiques essentielles dans un espace intermédiaire désormais bien structuré. Toutefois, en raison de la dynamique non-linéaire sous-jacente, cette projection ne suffit pas à elle seule pour obtenir un modèle réduit suffisamment précis. Pour pallier à cette limite, nous la combinons avec notre méthode introduite dans [1]. Le processus d'apprentissage global est facilité par la qualité de la représentation intermédiaire, qui capture les caractéristiques essentielles de la dynamique d'origine sans filtrer excessivement les comportements non-linéaires.

Nous démontrons l'efficacité de notre méthode sur trois problèmes tests classiques : l'amortissement Landau linéaire, l'amortissement Landau non-linéaire et l'instabilité double faisceau.

Chapitre 5 - Hyperbolic reduced model for Vlasov-Poisson equation with Fokker-Planck collision

Le dernier Chap. 5 présente le travail réalisé lors de l'école d'été du CEMRACS 2022, qui a donné lieu à une communication [2]. Comme dans le Chap. 4 précédent, nous considérons l'équation de Vlasov-Poisson 1D-1V, cette fois avec un terme de collision de type Fokker-Planck $Q(f)$ défini par

$$\begin{cases} \partial_t f(t,x,v;\mu) + v\partial_x f(t,x,v;\mu) + \dfrac{q}{m} E(t,x;\mu)\partial_v f(t,x,v;\mu) = \dfrac{1}{\varepsilon} Q(f)(t,x,v;\mu), \\ \partial_x E(t,x;\mu) = q \displaystyle\int f(t,x,v;\mu)\, dv - 1, \end{cases} \tag{1.4}$$

où f représente à nouveau la distribution solution sur un domaine périodique en espace, E est le champ électrique qu'elle génère via l'équation de Poisson, et Q désigne l'opérateur de collision non-linéaire de Fokker-Planck, donné par

$$Q(f) = \partial_v \left[(v - u_f) f + T_f \partial_v f \right]$$

avec $u_f(t,x), T_f(t,x)$ désignent respectivement la vitesse et la température de la distribution. Le paramètre est $\mu = (\alpha\ \varepsilon)^T \in \Gamma \subset \mathbb{R}^2$ où α paramètre la condition initiale et $\varepsilon > 0$ sert de facteur d'échelle, permettant de distinguer les régimes fortement colisionnels ($\varepsilon \ll 1$) de ceux avec peu de collisions.

Une part importante du coût calcul provient de la variable vitesse. Pour y remédier, nous construisons un modèle réduit dans l'espace des vitesses en appliquant une compression sur l'équation semi-discrétisée. Afin d'améliorer davantage l'efficacité numérique, nous appliquons la méthode d'interpolation empirique discrète (Discrete Empirical Interpolation Method, DEIM) à l'opérateur non-linéaire de Fokker–Planck. La taille du modèle réduit est déterminée en fonction du nombre de Knudsen. Par ailleurs, nous introduisons une correction de l'opérateur de collision réduit afin que les moments réduits satisfassent un système de type Euler : les trois premiers moments réduits sont préservé par l'opérateur de collision réduit ainsi que la propriété qu'il s'annule sur des maxwelliennes discrètes. Les expériences numériques montrent que le modèle réduit reproduit fidèlement la dynamique de Vlasov–Poisson dans divers régimes collisionnels, pour différentes conditions initiales et au delà du temps de simulation exploré par le modèle réduit à sa construction.

Ce travail diffère des autres (Chaps. 3 et 4), en ce que le modèle réduit n'est pas hamiltonien. Il est conçu de manière à ce que d'autres quantités / propriétés soient conservées comme décrit plus haut.

References

[1] R. Côte et al. "Hamiltonian reduction using a convolutional auto-encoder coupled to a Hamiltonian neural network". In: *Commun. Comput. Phys.* 37.2 (2025), pp. 315–352. ISSN: 1815-2406,1991-7120. DOI: 10.4208/cicp.OA-2023-0300.

[2] E. Franck et al. "Hyperbolic reduced model for Vlasov-Poisson equation with Fokker-Planck collision". In: *ESAIM: ProcS* 77 (2024), pp. 213–228. DOI: 10.1051/proc/202477213.

Chapter 2

Hamiltonian systems: numerical methods, reduction and deep learning tools

Chapter's contents

2.1 A primer on Hamiltonian dynamics 14
2.2 Hamiltonian systems & numerical methods 19
2.2.1 A formal introduction to Hamiltonian PDEs 19
2.2.2 Hamiltonian ODEs: definition & properties 22
2.2.3 Numerical methods for time integration 27
2.3 Linear model order reduction for Hamiltonian ODEs 30
2.3.1 Proper Symplectic Decomposition 31
2.3.2 Building the projection matrix 33
2.3.3 Hyper-reduction with symplectic DEIM 35
2.3.4 A practical example: the shallow-water system 37
2.3.5 Other reduction methods 42
2.4 Deep learning tools for Hamiltonian model order reduction 43
2.4.1 Outline of neural networks 43
2.4.2 Learning symplectic flows with Hamiltonian Neural Networks 45
2.4.3 Convolutional AutoEncoder for low-dimensional representations . . . 46
References 48

2.1 A primer on Hamiltonian dynamics

The objective of this section is to introduce a few basics of Hamiltonian mechanics. We provide a gentle introduction to these systems, beginning with a handful of elementary concepts from Newtonian mechanics, and proceed to describe the corresponding equations of motion. These are then illustrated through several toy examples.

Basics of Newton's mechanics One of the earliest major developments in mechanics with a mathematical formalism comes from I. Newton in his treatise *Philosophiæ Naturalis Principia Mathematica* [1] published in 1687. Some of these principles can be expressed as follows. In a $d > 0$ dimensional real space $\mathbb{R}^d$, a point-like particle's motion at time $t \in \mathbb{R}_+$ is described with its position $\mathbf{x} \in C^2(\mathbb{R}, \mathbb{R}^d)$ and its velocity $\mathbf{v} \in C^1(\mathbb{R}, \mathbb{R}^d)$. The velocity is given by the rate of change of the position

$$\mathbf{v}(t) := \frac{d\,\mathbf{x}(t)}{d\,t},$$

and so that the acceleration is the rate of change of the velocity $\frac{d\,\mathbf{v}(t)}{d\,t} = \frac{d^2\,\mathbf{x}(t)}{d\,t^2}$. Newton's Second Law states that the force $\mathbf{F}$ that drives a particle's motion is equal to its mass $m > 0$ multiplied by its acceleration

$$\mathbf{F}(t) := \frac{d\,m\mathbf{v}(t)}{d\,t} = m\frac{d\,\mathbf{v}(t)}{d\,t} = m\frac{d^2\,\mathbf{x}(t)}{d\,t^2},$$

which can be written as

$$\begin{cases} \dfrac{d\,\mathbf{x}(t)}{d\,t} = \mathbf{v}(t), \\ \dfrac{d\,\mathbf{v}(t)}{d\,t} = \dfrac{1}{m}\mathbf{F}(t). \end{cases} \tag{2.1}$$

By adding a few principles, such as the law of inertia and the action-reaction principle, we obtain the rudiments of classical mechanics. This theory accurately describes the motion of objects that have a reasonable mass and size, and are not moving at too high a speed.

Classical mechanics argues that once we know the force $\mathbf{F}$ applied to an object (and some initial / boundary conditions), we can deduce its dynamics. This force can take a variety of forms. Here, we focus on conservative forces. A force $\mathbf{F} : \mathbb{R}^d \to \mathbb{R}^d$ is said conservative if and only if there exists a potential $\varphi : \mathbb{R}^d \to \mathbb{R}$ such that

$$\mathbf{F}(\mathbf{x}) = -\nabla\varphi(\mathbf{x}),$$

where ∇ is the gradient operator $\left(\nabla\varphi(\mathbf{x})\right)_i = \partial_{x_i}\varphi(\mathbf{x})$. The system (2.1) rewrites

$$\begin{cases} \dfrac{d\,\mathbf{x}(t)}{d\,t} = \mathbf{v}(t), \\ \dfrac{d\,\mathbf{v}(t)}{d\,t} = -\dfrac{1}{m}\nabla\varphi(\mathbf{x}(t)). \end{cases} \tag{2.2}$$

Subsequently, φ is commonly referred to as the potential energy of the system. Potential energy represents the energy an object possesses due to its position relative to other objects. For instance, conservative forces include the restoring force of a spring $\mathbf{F}(\mathbf{x}) = k\mathbf{x}$ with k the spring stiffness constant and the gravitational force near the Earth $\mathbf{F} = m\mathbf{g}$ with $\mathbf{g}$ the gravity field. Similarly, there are many non-conservative forces. For example, there are dissipative forces such as the drag force $\mathbf{F}(\mathbf{v}) = b\mathbf{v}$ with b a drag coefficient or the rolling friction $\mathbf{F} = c\,\mathbf{n}$ with c a friction coefficient and $\mathbf{n}$ a normal vector.

There is one other type of energy present in our system: the kinetic energy $\mathcal{K} : \mathbb{R}^d \to \mathbb{R}$ is the form of energy resulting to the object's motion $\mathbf{v}$. In our setting, it is given by

$$\mathcal{K}(\mathbf{v}) = \frac{1}{2} m \|\mathbf{v}\|^2.$$

Ultimately, the total energy $\mathscr{H}$ is defined as the sum of the kinetic energy $\mathcal{K}$ and the potential energy φ

$$\mathscr{H}(\mathbf{x}, \mathbf{v}) := \mathcal{K}(\mathbf{v}) + \varphi(\mathbf{x}).$$

A system is called conservative when all the forces acting on it are conservative, the total energy of the system remains conserved over time. Indeed

$$\begin{aligned} \frac{d\,\mathscr{H}(\mathbf{x}(t), \mathbf{v}(t))}{d\,t} &= \nabla_{\mathbf{v}} \mathscr{H}(\mathbf{x}(t), \mathbf{v}(t)) \cdot \frac{d\,\mathbf{v}(t)}{d\,t} + \nabla_{\mathbf{x}} \mathscr{H}(\mathbf{x}(t), \mathbf{v}(t)) \cdot \frac{d\,\mathbf{x}(t)}{d\,t} \\ &= m\mathbf{v}(t) \cdot \frac{d\,\mathbf{v}(t)}{d\,t} + \nabla\varphi(\mathbf{x}) \cdot \mathbf{v}(t) \\ &= -\nabla\varphi(\mathbf{x}(t)) \cdot \mathbf{v}(t) + \nabla\varphi(\mathbf{x}(t)) \cdot \mathbf{v}(t) = 0. \end{aligned}$$

These conservatives systems are called Hamiltonian systems. The conservation of energy is only one aspect of these systems. As we will see, they also encode the structure of the phase space, and it is the combination of these two features that allows us to predict their dynamics.

A first look at Hamiltonian mechanics In this part, we present a sketch of Hamiltonian formalism in classical mechanics, which was first introduced in 1833 by W. R. Hamilton in [2]. It focuses on the total energy $\mathscr{H}$ of the system. First of all, we must generalize the concept of position to accommodate a system of multiple bodies, where the particles may undergo some constrained motion. We define the generalized coordinates

$$\mathbf{q}(t) \in \mathbb{R}^N.$$

It is a vector of $N > 0$ parameters used to uniquely describe the state of the system, they are also called the degrees of freedom. For example, the motion of a pendulum in the plane is constrained by the string that holds it. It is sufficient to know the angle formed by the string to describe the position of the pendulum, rather than using the complete 2D Cartesian coordinates. In comparison, a n particles system moving freely in 3 dimension would require $N = 3n$ parameters to be described. In addition to the generalized coordinates, the system is described with the generalized momentum

$$\mathbf{p}(t) \in \mathbb{R}^N.$$

For instance, the generalized momentum of a distribution of N particles of mass m moving freely is $\mathbf{p}(t) = m\dot{\mathbf{q}}(t)$. Additional terms, such as angular or electromagnetic momenta, can also be introduced depending on the system considered. In our case $\mathbf{p} = m\dot{\mathbf{q}}$, the kinetic energy becomes $\mathcal{K}(\mathbf{p}(t)) = \|\mathbf{p}(t)\|^2/(2m)$. We can then rewrite the Hamiltonian $\mathscr{H}$ as

$$\mathscr{H}(\mathbf{q}(t), \mathbf{p}(t)) = \frac{1}{2m} \|\mathbf{p}(t)\|^2 + \varphi(\mathbf{q}(t)).$$

Arguing Eq. (2.2), we derive Hamilton's equations

$$\begin{cases} \dot{\mathbf{q}}(t) = \nabla_{\mathbf{p}} \mathscr{H}(\mathbf{q}(t), \mathbf{p}(t)), \\ \dot{\mathbf{p}}(t) = -\nabla_{\mathbf{q}} \mathscr{H}(\mathbf{q}(t), \mathbf{p}(t)). \end{cases} \tag{2.3}$$

In fact, these systems exhibit a much richer structure, involving more general Hamiltonian functionals and vector fields with significant geometric structure as we will explore in Sec. 2.2. We finish this section with three toy examples of Hamiltonian systems: the harmonic oscillator, the simple gravity pendulum and the 2-body problem.

Harmonic oscillator We consider a 1D system where a spring with a stiffness constant $k > 0$ is attached at one end to a fixed wall and at the other end to a mobile mass $m > 0$. The quantity $x \in \mathbb{R}$ is the displacement of the mass relative to its resting position. We denote $\omega = \sqrt{k/m}$ the angular frequency. We scale the displacement $q = \sqrt{m\omega}x$ and the momentum $p = \omega^{-1}\dot{q}$. Arguing Hooke's law for ideal springs, we derive the potential energy $\varphi(q) = \omega q^2/2$. With the kinetic energy $\mathcal{K}(p) = \omega p^2/2$, we then write the Hamiltonian

$$\mathscr{H}(q,p) = \frac{\omega}{2}(q^2 + p^2).$$

We derive Hamilton's equation

$$\begin{cases} \dfrac{dq}{dt} = \omega p, \\ \dfrac{dp}{dt} = -\omega q. \end{cases}$$

The solution of the system is

$$\begin{cases} q(t) = q_{\text{init}}\cos(\omega t) + p_{\text{init}}\sin(\omega t), \\ p(t) = p_{\text{init}}\cos(\omega t) - q_{\text{init}}\sin(\omega t). \end{cases}$$

We remark that

$$\forall t,\ q^2(t) + p^2(t) = r^2,$$

meaning the trajectory in the phase space is a circle of constant radius $r = \sqrt{q_{\text{init}}^2 + p_{\text{init}}^2}$. Furthermore, $\mathscr{H}(q(t),p(t)) = \omega r^2/2$ is constant and we verify that the total energy is indeed conserved.

We observe the streamline plot of this system for $\omega = 1$ in Fig. 2.1. Such a plot shows curves (streamlines) that remain tangent to the vector field at every point. Put differently, each streamline represents a trajectory that a point (q,p) in the phase-space would follow if it were initially placed on that curve. We verify that trajectories are circles.

Simple gravity pendulum We consider a mass m tied to a rigid rope of length l, with the other end fixed to the ceiling. The quantity $q \in [-\pi, \pi[$ is the angle between the rope and the downward vertical equilibrium and its corresponding momentum $p = ml^2\dot{q}$. Arguing a constant and downward gravity force of constant $g > 0$, the potential energy is $\varphi(q) = mgl(1 - \cos q)$ hence the Hamiltonian of the system reads

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

Then, we derive Hamilton's equations

$$\begin{cases} \dfrac{dq}{dt} = \dfrac{1}{ml^2}p, \\ \dfrac{dp}{dt} = mgl\sin q. \end{cases}$$

Beyond the small angle approximation $\sin q = q + \mathcal{O}(q^2)$, no closed-form solution is known. Indeed, looking at the streamline plot of the system in Fig. 2.2, we observe that, for small angles i.e. $q \ll 1$, the trajectory (q,p) is close to a circle. For larger angles, the motion near the maximum amplitude deviates from the circular trajectory, displaying a rotational motion until reaching an unstable equilibrium point at $(q,p) = (\pi, 0)$.

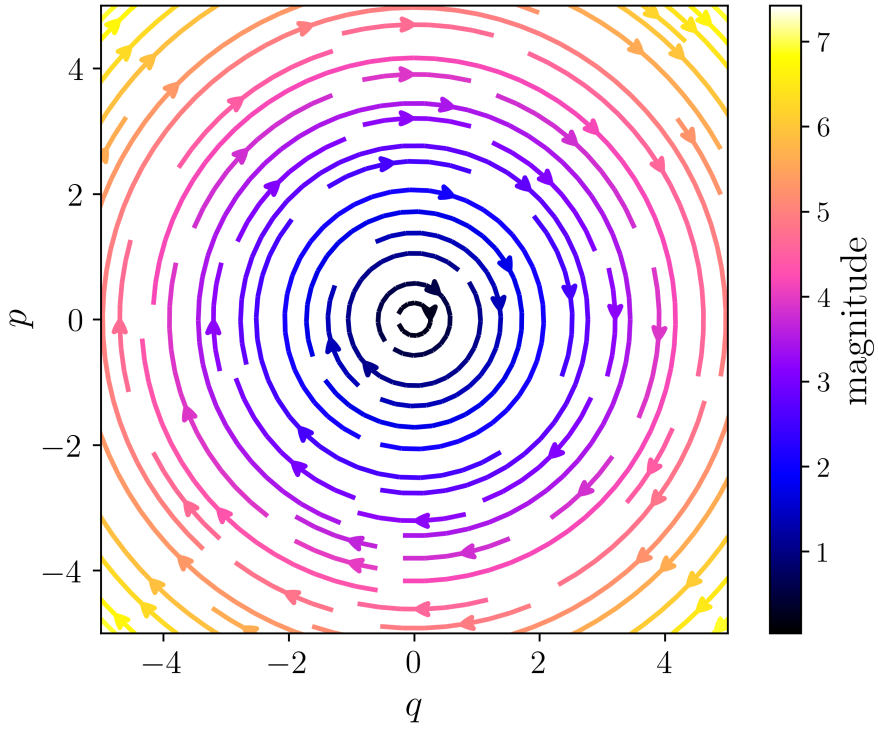


Figure 2.1: (Harmonic oscillator) Streamline plot for $\omega = 1$. The color-map represents the values of the magnitude $\|\dot{u}\|_2$.

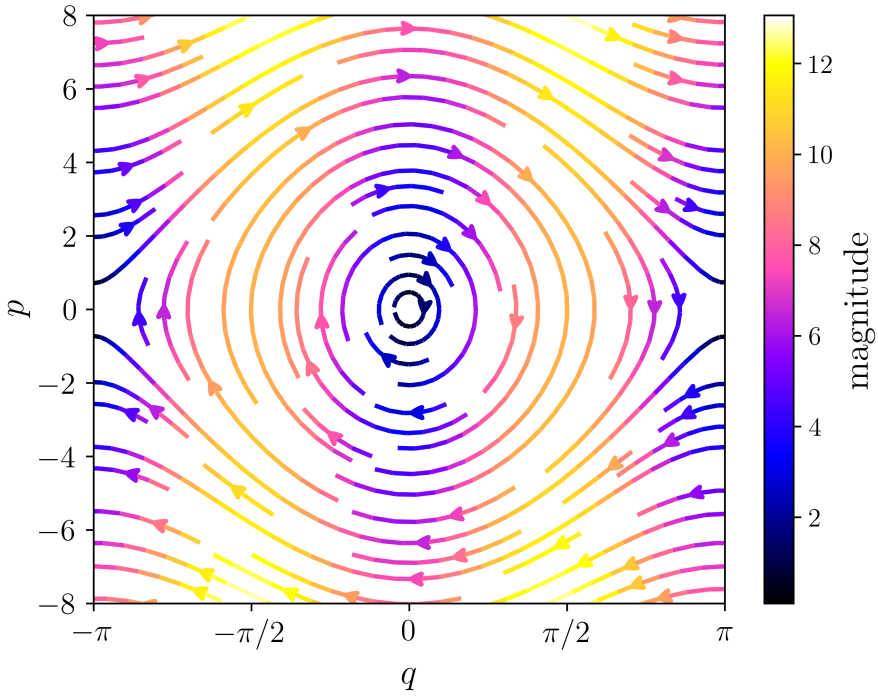


Figure 2.2: (Simple gravity pendulum) Streamline plot for $m = l = 1, g = 10$. The color-map represents the values of the magnitude $\|\dot{u}\|_2$.

2-body problem We consider a simplest version of the well-known n-body problem [3], asking whether it is possible to solve the individual trajectories of a group of celestial objects interacting

gravitationally. We set $n = 2$ e.g. the Earth-Sun system or a twin stars system. We admit the motion is planar. For all $i \in \{1, 2\}$, we denote $\mathbf{q}_i$ the position of the i-th body of mass m_i and momentum $\mathbf{p}_i = m_i \dot{\mathbf{q}}_i$. Arguing Newton's law of universal gravitation, the potential energy is $\varphi(\mathbf{q}_1, \mathbf{q}_2) = -Gm_1m_2/\|\mathbf{q}_1 - \mathbf{q}_2\|$ and the Hamiltonian of the system is

$$\mathscr{H}(\mathbf{q}, \mathbf{p}) = \frac{1}{2m_1}\|\mathbf{p}_1\|^2 + \frac{1}{2m_2}\|\mathbf{p}_2\|^2 - G\frac{m_1m_2}{\|\mathbf{q}_1 - \mathbf{q}_2\|},$$

with $G > 0$, the universal gravity constant. We derive Hamilton's equations

$$\begin{cases} \dfrac{d\,\mathbf{q}_i}{d\,t} = \dfrac{1}{m_i}\mathbf{p}_i, & \forall i \in \{1, 2\}, \\ \dfrac{d\,\mathbf{p}_i}{d\,t} = G\dfrac{m_1m_2}{\|\mathbf{q}_j - \mathbf{q}_i\|^3}(\mathbf{q}_j - \mathbf{q}_i), & \forall i, j \in \{1, 2\}, j \neq i. \end{cases}$$

For instance, setting $m_1 = m_2 = G = 1$, an initial condition $\mathbf{q}_{\text{init},1} = -\mathbf{q}_{\text{init},2} = (-1\ 0)^T$ and $\mathbf{p}_{\text{init},1} = -\mathbf{p}_{\text{init},2} = (0\ 1/11)^T$ results in trajectories that are two ellipses with a common focal point, the latter being the unmoving barycenter of the system, as shown in Fig. 2.3.

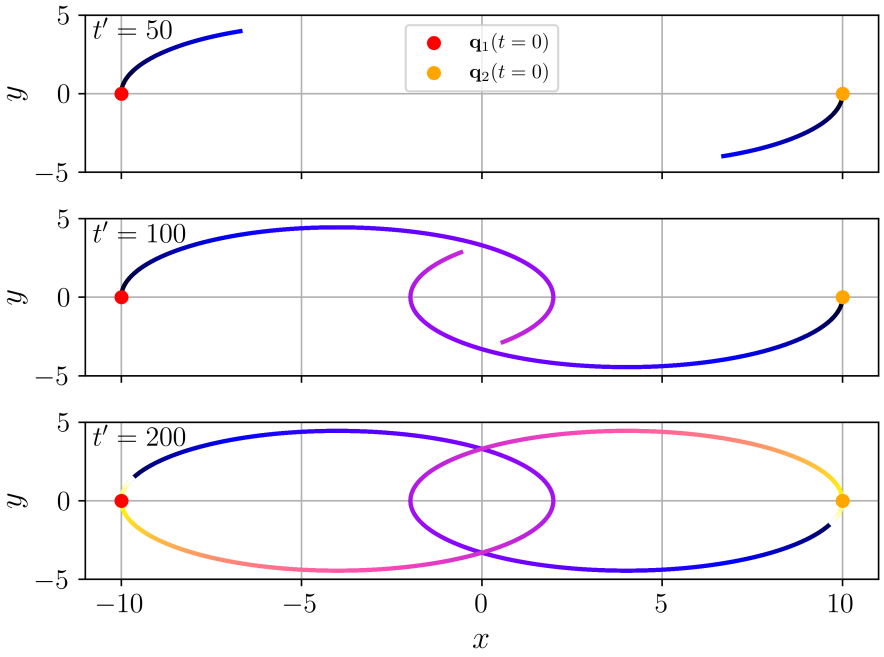


Figure 2.3: (2-body problem) Trajectories of $\mathbf{q}_1(t)$ and $\mathbf{q}_2(t)$. Each sub-figure portrays the trajectories for all times $t \in [0, t']$. The color map represents the time evolution for $t = 0$ until $t = 200$, from darker to brighter shades.

For more details on Hamiltonian systems, see [4, 5, 6]. In the remainder of this manuscript, we focus on high-dimensional Hamiltonian Ordinary Differential Equations (ODEs) arising from the semi-discretization of Hamiltonian Partial Differential Equations (PDEs).

2.2 Hamiltonian systems & numerical methods

To begin, we provide a formal introduction to Hamiltonian PDEs, outlining their various formulations and key properties in Sec. 2.2.1 as well as numerical methods for semi-discretizing Hamiltonian PDEs into Hamiltonian ODEs, employing techniques such as finite elements or particle-based methods. We then explore the mathematical properties of Hamiltonian systems in the ODE framework in Sec. 2.2.2, as it offers a broader, well-posed and more accessible perspective. Finally, we conclude with the terminal discretization process described in Sec. 2.2.3, leading to numerical models and including a section on time integrators.

2.2.1 A formal introduction to Hamiltonian PDEs

In a general setting, a Hamiltonian system describes the evolution of a field $u \in V$ with V some functional space. Typically, this field $u(x,t;\mu)$ depends on space $x \in \Omega \subset \mathbb{R}^d$, time $t \in \mathcal{T} = [0,T]$ and some parameters $\mu \in \Gamma \subset \mathbb{R}^p$. Given a smooth Hamiltonian functional $\mathscr{H} : V \to \mathbb{R}$ of the system, the evolution of the state u is determined by the following equation,

$$\frac{\partial u}{\partial t} = \mathcal{J}(u)\frac{\delta \mathscr{H}}{\delta u}(u), \tag{2.4}$$

It is called a Poisson system. This is generalization of Hamiltonian ones, where $\mathcal{J}$ depends on state u. More precisely, $\mathcal{J}(u) : V \to V$ is a skew-adjoint operator

$$\int_\Omega \mathcal{J}(u)v \cdot w\, dx = -\int_\Omega v \cdot \mathcal{J}(u)w\, dx,\ \forall v, w \in V,$$

which satisfies a Jacobi identity defined as, for any smooth functionals $\mathcal{F}, \mathcal{G}, \mathcal{H} : V \to \mathbb{R}$

$$\{\mathcal{F}, \{\mathcal{G}, \mathcal{H}\}\} + \{\mathcal{H}, \{\mathcal{F}, \mathcal{G}\}\} + \{\mathcal{G}, \{\mathcal{H}, \mathcal{F}\}\} = 0,$$

where $\{\cdot,\cdot\}$ denotes the so-called Poisson bracket. The Poisson bracket of two functionals $\mathcal{F}(u)$ and $\mathcal{G}(u)$ is formally defined as

$$\{\mathcal{F}, \mathcal{G}\}(u) = \int_\Omega \frac{\delta \mathcal{F}(u)}{\delta u} \cdot \mathcal{J}(u)\frac{\delta \mathcal{G}(u)}{\delta u}\, dx,$$

where $\cdot$ denotes the inner product of V, and for any functional $\mathcal{F} : V \to \mathbb{R}$, $\delta\mathcal{F}/\delta u$ is the functional derivative of $\mathcal{F}$.

The operator $\mathcal{J}(u)$ is referred to as the Poisson structure operator, and it encapsulates the intrinsic geometric framework of the phase space. In most settings, the Hamiltonian functional $\mathscr{H}$ represents the total energy of the system, whereas $\mathcal{J}(u)$ prescribes the geometric structure through which the dynamics is driven. When the Poisson structure is nontrivial or exhibits dependence on the state variable u, the verification of the Jacobi identity becomes a highly delicate and nontrivial task, often requiring additional compatibility conditions.

A strength of this formulation is that the Hamiltonian structure is naturally encoded in it. For instance, the Hamiltonian $\mathscr{H}$ is formally preserved throughout time

$$\frac{d}{dt}\mathscr{H}(u(.,t)) = 0.$$

Indeed, using the chain rule and arguing that $\mathcal{J}(u)$ is skew-adjoint

$$\frac{d}{dt}\mathscr{H}(u(.,t)) = \int_\Omega \frac{\delta \mathscr{H}}{\delta u}(u(.,t)) \cdot \frac{\partial u}{\partial t}(.,t) = \int_\Omega \frac{\delta \mathscr{H}}{\delta u}(u(.,t)) \cdot \mathcal{J}(u(.,t))\frac{\delta \mathscr{H}}{\delta u}(u(.,t)) = 0.$$

More generally, the evolution of any smooth observable $\mathcal{F} : V \to \mathbb{R}$ is given by the following equation:

$$\frac{d}{dt}\mathcal{F}(u) = \{\mathcal{F}, \mathscr{H}\}(u).$$

In particular, we can recover formally the evolution Eq. (2.4) by considering the functional $u \mapsto \delta_x u = u(x,t)$

$$\begin{aligned}\frac{\partial u}{\partial t}(x,t) = \frac{\partial\, \delta_x u(.,t)}{\partial t} &= \{\delta_x, \mathscr{H}\}(u) \\ &= \int_\Omega \delta_x \mathcal{J}(u) \frac{\delta \mathscr{H}}{\delta u}(u)\, dx' = \mathcal{J}(u)\frac{\delta \mathcal{H}}{\delta u}(u),\end{aligned}$$

noting that $u \mapsto \delta_x u$ has functional derivative equal to δ_x. Thus, Eq. (2.4) can sometimes be written as

$$\frac{\partial u}{\partial t}(x,t) = \{\delta_x, \mathscr{H}\}(u). \tag{2.5}$$

More details on the finite and infinite-dimensional Hamiltonian structure can be found in [7, Chapters 6 & 7],[8, Chapters 8 & 9] and [9]. These systems are described under standard Poisson operator in [7] and in the field dependent case in [10] in which the Jacobi identity must be carefully verified.

We now restrict our attention to Hamiltonian systems that can be expressed in canonical coordinates. In this canonical setting, the function $u(x,t;\mu)$ takes values in $\mathbb{R}^{2N}$ and the system writes:

$$\frac{\partial \mathbf{u}}{\partial t} = \mathcal{J}\frac{\delta \mathscr{H}}{\delta \mathbf{u}}(\mathbf{u}), \quad \text{with } \mathcal{J}(\mathbf{u}) = \begin{pmatrix} 0 & \mathrm{Id} \\ -\mathrm{Id} & 0 \end{pmatrix}. \tag{2.6}$$

Equivalently, by introducing the canonical coordinates $(q(x,t;\mu), p(x,t;\mu))^T = \mathbf{u}(x,t;\mu)$, the system becomes

$$\begin{cases} \partial_t q = \dfrac{\delta \mathscr{H}}{\delta p}(q,p), \\ \partial_t p = -\dfrac{\delta \mathscr{H}}{\delta q}(q,p). \end{cases}$$

This system is thus associated with the canonical Poisson bracket in (q,p) coordinates:

$$\{\mathcal{F}, \mathcal{G}\}(q,p) = \int_\Omega \left(\frac{\delta \mathcal{F}}{\delta q}(q,p)\frac{\delta \mathcal{G}}{\delta p}(q,p) - \frac{\delta \mathcal{F}}{\delta p}(q,p)\frac{\delta \mathcal{G}}{\delta q}(q,p) \right) dx. \tag{2.7}$$

Remark. *Thus, the presented theory can be extended to non-canonical coordinates, with $\mathcal{J}(u)$ depends on the field, to non-autonomous Hamiltonians, and even to Hamiltonian systems defined on manifolds, among other generalizations. However, these extensions will not be discussed in this work.*

Then, we give examples of Hamiltonian functionals of the systems explored in each chapter and given in Sec. 1.2 along with their Poisson bracket.

Nonlinear wave equation We recall the equation

$$\partial_{tt} v(x,t;\mu) - \mu_a \partial_x \left[w'\left(\partial_x v(x,t;\mu), \mu_b\right)\right] + g'(v(x,t;\mu), \mu_c) = 0, \tag{1.1}$$

of solution $v : \mathbb{R}/(L\mathbb{Z}) \times [0,T] \times \Gamma \to \mathbb{R}$ the vertical displacement on a periodic domain of length L with $\mu = (\mu_a\ \mu_b\ \mu_c)^T \in \Gamma \subset \mathbb{R}^3$ some parameters. With the momentum $p := \partial_t v$, the Hamiltonian is

$$\mathscr{H}[v,p] = \int_{\mathbb{R}/(L\mathbb{Z})} \left(\frac{1}{2}p^2 + \mu_a w(\partial_x v, \mu_b) + g(v, \mu_c) \right) dx$$

associated with the canonical Poisson bracket in (v,p) coordinates defined in Eq. (2.7).

2D shallow-water equations We remind the system

$$\begin{cases} \partial_t \chi(\mathbf{x}, t; \mu) + \nabla \cdot ((1 + \chi(\mathbf{x}, t; \mu)) \nabla \phi(\mathbf{x}, t; \mu)) = 0, \\ \partial_t \phi(\mathbf{x}, t; \mu) + \dfrac{1}{2} |\nabla \phi(\mathbf{x}, t; \mu)|^2 + \chi(\mathbf{x}, t; \mu) = 0, \end{cases} \tag{1.2}$$

where $\chi, \phi : \mathbb{R}^2/(L\mathbb{Z}^2) \times [0, T] \times \Gamma \to \mathbb{R}$ are the perturbation from the equilibrium and the scalar velocity potential, respectively, on a square periodic domain of length L. With a canonical Poisson bracket from Eq. (2.7) in (χ, ϕ) variables, the Hamiltonian is

$$\mathscr{H}[\chi, \phi] = \frac{1}{2} \int_{\mathbb{R}^2/(L\mathbb{Z}^2)} \left((1 + \chi) |\nabla \phi|^2 + \chi^2 \right) d\mathbf{x}. \tag{2.8}$$

1D-1V Vlasov-Poisson equations This system is defined as follows

$$\begin{cases} \partial_t f(t, x, v; \mu) + v\, \partial_x f(t, x, v; \mu) + \dfrac{q}{m} E(t, x; \mu)\, \partial_v f(t, x, v; \mu) = 0, \\ \partial_x E(t, x; \mu) = q \displaystyle\int f(t, x, v; \mu)\, dv - 1, \end{cases} \tag{1.3}$$

with solution a particles distribution $f : [0, T] \times \mathbb{R}/(2\pi\mathbb{Z}) \times \mathbb{R} \to \mathbb{R}$. It admits a Hamiltonian formulation with the Hamiltonian

$$\mathscr{H}[f] = \frac{m}{2} \int_{\mathbb{R}} v^2 f(t, x, v; \mu)\, dxdv + \frac{1}{2} \int_{\mathbb{R}/(2\pi\mathbb{Z})} |E(t, x; \mu)|^2\, dx,$$

with a non-canonical Poisson bracket

$$\{\mathcal{F}, \mathcal{G}\} = \int_{\mathbb{R}/(2\pi\mathbb{Z}) \times \mathbb{R}} \frac{\delta \mathcal{F}}{\delta f}(f) \left\{ \frac{\delta \mathcal{G}}{\delta f}(f), f \right\}_{x,v} dxdv,$$

where $\{\cdot, \cdot\}_{x,v}$ is the canonical Poisson bracket in (x, v) coordinates given in Eq. (2.7). In this setting, we note that the structure is no longer canonical. More details are presented in [11].

As previously mentioned, the Hamiltonian often corresponds to the total energy of the system, while the Poisson bracket encodes its structural properties. The identification of appropriate Poisson brackets for complex physical systems remains an active area of research. In practice, we are interested in the Hamiltonian ODEs arising from such Hamiltonian PDEs.

Towards ordinary differential equations In the last part of this section, we present a formal semi-discretization of Hamiltonian PDEs into Hamiltonian ODEs in 1D. We replace $u(x, t; \mu) \in V$ by a finite dimensional approximation $\mathbf{u}_h(t; \mu) \in \mathbb{R}^{2N}$ solution of an Hamiltonian ODE of the form

$$\frac{d\, \mathbf{u}_h(t; \mu)}{d\, t} = \mathcal{J}_h \nabla \mathscr{H}_h(\mathbf{u}_h(t; \mu))$$

with ∇ the standard gradient of $\mathbb{R}^{2N}$ in place of the functional derivative $\delta/\delta u$, and $h > 0$ a given characteristic size of the discretization. In this manuscript, we use three different families of discretization techniques.

1. **With finite differences :** we consider a mesh $x_i = ih$ over Ω of an uniform cell size h. The approximation is $u_i(t; \mu) \approx u(x_i, t; \mu)$ with u_i the i-th value of $\mathbf{u}_h$. Then, derivatives are replaced by difference quotients, for example

$$\partial_x u(x_i, t; \mu) \approx \frac{u_{i+1} - u_{i-1}}{2h}, \quad \partial_{xx} u(x_i, t; \mu) \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}$$

and integrals are replaced by a quadrature formula in the expression of $\mathscr{H}$ and we obtain $\mathscr{H}_h$.

2. **With finite elements :** we consider a set of $2N$ basis functions $\{\alpha_i(x)\}_i$ e.g. hat functions or splines to approximate $u(x,t;\mu)$ in the form

$$u_h(x,t;\mu) = \sum_{i=1}^{2N} u_i(t;\mu)\alpha_i(x).$$

After that, we test the weak formulation of $\mathscr{H}$ against α_j and we derive a discrete Hamiltonian $\mathscr{H}_h$ possibly composed of some mass or stiffness matrices.

3. **With particle-based approximation :** we approach $u(t,x;\mu)$ by a discrete distribution of particles $\{x_i(t;\mu)\}_i$ in the phase space such that

$$u_h(x,t;\mu) \approx \sum_{i=1}^{2N} \omega_i \delta(x - x_i(t;\mu)),$$

with ω_i are some weights.

Thus, in canonical coordinates, a natural discretization of $\mathcal{J}$ is the canonical symplectic matrix $\mathcal{J}_{2N}$ defined as

$$\mathcal{J}_{2N} = \begin{pmatrix} 0 & I_N \\ -I_N & 0 \end{pmatrix} \in \mathcal{M}_{2N}(\mathbb{R})$$

with I_N the identity matrix of size N.

The main difficulty is to preserve the Hamiltonian structure at the discrete level. To this end, $\mathcal{J}_h$ must be skew-symmetric, i.e. $\mathcal{J}_h^T = -\mathcal{J}_h$, which is natural in the canonical case, and $\mathscr{H}_h$ is conserved.

$$\frac{d}{dt}\mathscr{H}_h(\mathbf{u}_h(t;\mu)) = 0.$$

For instance, in the canonical case, centered finite difference schemes may suffice. For more general systems, however, the numerical approximation of the bracket $\{\cdot,\cdot\}_h$ must be built so that it is anti-symmetric and satisfy the Jacobi identity. The development of Hamiltonian semi-discretization techniques remains an active area of research. In the context of finite difference methods, a comprehensive review can be found in [12, 13]. For finite element methods, appropriate variational integrators must be constructed, as discussed in [14]. Finally, for particle-based methods, the GEMPIC framework [15] provides a Hamiltonian Particle-In-Cell (PIC) scheme for solving the Vlasov–Maxwell system. The specific numerical methods employed for each equation considered will be detailed in the corresponding chapters.

Hereafter, (i) Hamiltonian functionals $\mathscr{H}[u]$ are replaced by Hamiltonian functions $\mathscr{H}(u)$, (ii) the functional derivative $\delta\mathscr{H}/\delta u$ is replaced the vector gradient operator $\nabla\mathscr{H}$ and (iii) the skew-adjoint operator $\mathcal{J}$ is replaced by a skew-symmetric matrix $\mathcal{J}_{2N}$. To maintain clarity and consistency, we omit the subscript h and retain the same notation throughout.

2.2.2 Hamiltonian ODEs: definition & properties

This section addresses the dynamics of canonical Hamiltonian ODEs resulting from the semi-discretization of Hamiltonian PDEs. Those systems are described using a vector state variable

$$\mathbf{u}(t;\mu) := \begin{pmatrix} \mathbf{q}(t;\mu) \\ \mathbf{p}(t;\mu) \end{pmatrix} \in \mathbb{R}^{2N},$$

where $\mathbf{q}$ and $\mathbf{p}$ are the generalized coordinates and the generalized momentum, respectively, $t \in [0,T]$ is the time, $N > 0$ the degrees of freedom and $\mu \in \Gamma \subset \mathbb{R}^p$ are $p \geq 0$ parameters.

Definition 2.2.1 (Canonical Hamiltonian ODE). *Let $\mathbf{u}(t;\mu) \in C^1(\mathbb{R}, \mathbb{R}^{2N})$ be a state variable at time $t > 0$ and parameters $\mu \in \Gamma$. $\mathbf{u}$ follows an Hamiltonian dynamics with an associated Hamiltonian function $\mathscr{H} : \mathbb{R}^{2N} \to \mathbb{R}$ if*

$$\begin{cases} \dfrac{d\,\mathbf{u}(t;\mu)}{d\,t} = \mathcal{J}_{2N} \nabla_{\mathbf{u}} \mathscr{H}(\mathbf{u}(t;\mu)), \\ \mathbf{u}(0;\mu) = \mathbf{u}_{init}(\mu), \end{cases} \tag{2.9}$$

with $\mathcal{J}_{2N} = \begin{pmatrix} 0_N & I_N \\ -I_N & 0_N \end{pmatrix} \in \mathcal{M}_{2N}(\mathbb{R})$ is the canonical symplectic matrix and $\mathbf{u}_{init}(\mu)$ is a given initial condition. I_N and 0_N are the identity matrix of size N and the null matrix of size N, respectively.

Remark. *In this work, we consider time-independent Hamiltonian functions. Many of the concepts discussed below can be extended to the time-dependent case. However, this will not be addressed here.*

We note that the system (2.9) is equivalent to Hamilton's equations (2.3) provided an initial condition. Some of the systems described above are a special case where the Hamiltonian $\mathscr{H}$ is said to be separable.

Definition 2.2.2 (Separable Hamiltonian). *An Hamiltonian $\mathscr{H} : \mathbb{R}^{2N} \to \mathbb{R}$ is separable if there exists $\mathcal{K} : \mathbb{R}^N \to \mathbb{R}$ and $\varphi : \mathbb{R}^N \to \mathbb{R}$ such that*

$$\mathscr{H}(\mathbf{q}, \mathbf{p}) = \mathcal{K}(\mathbf{p}) + \varphi(\mathbf{q}). \tag{2.10}$$

In this case, the Hamiltonian system (2.9) simplifies

$$\begin{cases} \dfrac{d\,\mathbf{q}(t;\mu)}{d\,t} = \nabla \mathcal{K}(\mathbf{p}(t;\mu)), \\ \dfrac{d\,\mathbf{p}(t;\mu)}{d\,t} = -\nabla \varphi(\mathbf{q}(t;\mu)). \end{cases}$$

A variety of physical systems are described with such separable Hamiltonians. Nevertheless, the notion of Hamiltonian systems is more general than this. For instance, the Hamiltonian of the shallow-water system is not separable, for more details see Chap. 3.

The term symplectic in this context arises from the fact that the phase space in which the state variable $\mathbf{u}$ evolves has a symplectic structure. It comes from the Greek *symplektikos* and means entwined as the two aggregates of $\mathbf{u}$, namely $\mathbf{q}$ and $\mathbf{p}$, are tied together in the phase space, in some sens. The study of Hamiltonian systems takes place within a broader framework of symplectic geometry given by Poisson brackets as briefly seen in Sec. 2.2.1.

Hamiltonian systems possess several remarkable properties, including the preservation of certain quantities such as the volume and the Hamiltonian, or the reversibility. Herein, we highlight some of these important properties. We first need to define the flow of $d > 0$ dimensional ODE in a general context, which will be needed to describe the properties mentioned above.

Definition 2.2.3 (Flow of an ODE). *Let $\mathbf{y}(t) \in C^1(\mathbb{R}, \mathbb{R}^d)$ be a solution of the ODE*

$$\begin{cases} \dfrac{d\,\mathbf{y}(t)}{d\,t} = \boldsymbol{\mathcal{F}}(\mathbf{y}(t)), \\ \mathbf{y}(0) = \mathbf{y}_{init}, \end{cases} \tag{2.11}$$

of vector field $\boldsymbol{\mathcal{F}} \in C^1(\mathbb{R}^d, \mathbb{R}^d)$ and a given initial condition $\mathbf{y}_{init} \in C^1(\mathbb{R}, \mathbb{R}^d)$. The flow $\boldsymbol{\Phi}_t$ of this ODE is the map between the initial state $\mathbf{y}_{init}$ and the solution $\mathbf{y}(t)$ at time t. It is defined as

$$\begin{aligned} \mathbb{R}^d &\longrightarrow \mathbb{R}^d, \\ \mathbf{y}_{init} &\longmapsto \boldsymbol{\Phi}_t(\mathbf{y}_{init}) := \mathbf{y}(t). \end{aligned}$$

Next, we provide some useful properties of the flow $\mathbf{\Phi}_t$ in this general setting for the remainder of this section.

Lemma 2.2.4. *Let $\mathbf{\Phi}_t$ be the flow of the ODE defined in* (2.11). *$\mathbf{\Phi}_t$ satisfies the following properties:*

1. $\mathbf{\Phi}_0 = \mathrm{id}$,
2. $\mathbf{\Phi}_{t_1+t_2} = \mathbf{\Phi}_{t_2} \circ \mathbf{\Phi}_{t_1}$,
3. *$\mathbf{\Phi}_t$ is a C^1-diffeomorphism and its inverse is given by $\mathbf{\Phi}_t^{-1} = \mathbf{\Phi}_{-t}$.*

In the following paragraphs, we consider flows $\mathbf{\Phi}_t$ of Hamiltonian systems as characterized in Def. 2.2.1, which have additional properties.

Preservation of the Hamiltonian In Sec. 2.1, we established that total energy is conserved over time for conservative systems. This result holds for the Hamiltonian $\mathscr{H}$ of Hamiltonian ODEs as introduced in Def. 2.2.1.

Property 2.2.5. *Consider a Hamiltonian $\mathscr{H} : \mathbb{R}^{2N} \to \mathbb{R}$ that does not explicitly depend on time, then it is preserved over time*

$$\frac{d}{dt}\mathscr{H}(\mathbf{u}(t;\mu)) = 0.$$

In terms of flow $\mathbf{\Phi}_t$, it implies that, for all $\mathbf{u}(t;\mu) \in \mathbb{R}^{2N}$ solution of (2.9) *that*

$$\forall t \geq 0,\ \mathscr{H}(\mathbf{\Phi}_t(\mathbf{u})) = \mathscr{H}(\mathbf{u}).$$

Proof. Indeed

$$\begin{aligned}\frac{d\,\mathscr{H}(\mathbf{q},\mathbf{p})}{dt} &= \nabla_{\mathbf{q}}\mathscr{H}(\mathbf{q},\mathbf{p}) \cdot \frac{d\,\mathbf{q}}{dt} + \nabla_{\mathbf{p}}\mathscr{H}(\mathbf{q},\mathbf{p}) \cdot \frac{d\,\mathbf{p}}{dt} \\ &= \nabla_{\mathbf{q}}\mathscr{H}(\mathbf{q},\mathbf{p}) \cdot \nabla_{\mathbf{p}}\mathscr{H}(\mathbf{q},\mathbf{p}) - \nabla_{\mathbf{p}}\mathscr{H}(\mathbf{q},\mathbf{p}) \cdot \nabla_{\mathbf{q}}\mathscr{H}(\mathbf{q},\mathbf{p}) \\ &= 0.\end{aligned}$$

□

We retrieve the result from conservatives force in Newtonian mechanics. Prop. 2.2.5 is fundamental for Hamiltonian systems. Its conservation, along with a symplectic structure, guarantees faithful dynamics with respect to physical behaviors. Thus, as it is constant over time, knowing its initial value allows us to study its future behavior. Indeed, certain states correspond to potential/kinetic energies that cannot be reached: the system is therefore constrained.

Volume preservation Let $A \subset \mathbb{R}^{2N}$ be a measurable set, its volume $\mathrm{vol}(A)$ is defined as

$$\mathrm{vol}(A) = \int_A d\mathbf{y}.$$

Given that $\mathbf{\Phi}_t$ is a C^1-diffeomorphism, we can define the volume associated to the flow $\mathbf{\Phi}_t$ as

$$\mathrm{vol}(\mathbf{\Phi}_t(A)) = \int_{\mathbf{\Phi}_t(A)} d\mathbf{y} = \int_A |\det(D\mathbf{\Phi}_t(y))|\, d\mathbf{y},$$

with $D\mathbf{\Phi}_t = \left(\left(\frac{\partial(\mathbf{\Phi}_t)_i}{\partial y_j}\right)\right)_{i,j}$ is the Jacobian of $\mathbf{\Phi}_t$.Then, we mention the following property about volume preservation.

Property 2.2.6. *Consider a flow $\mathbf{\Phi}_t$ associated to a Hamiltonian system* (2.9)*, then the volume is conserved*

$$\mathrm{vol}(\mathbf{\Phi}_t(A)) = \mathrm{vol}(A),$$

or, equivalently

$$\frac{d}{dt}\det(D\mathbf{\Phi}_t) = 0.$$

The proof of this property relies on the application of Liouville's theorem. More details in [4, 5].

Reversibility The reversibility of a dynamic system implies that reversing the direction of the velocity vector while keeping all other quantities unchanged, the system's motion will simply be reversed. In other words, an invertible system can return to a previous state by inverting its velocity vector.

We denote $\boldsymbol{\rho}_0 : \mathbb{R}^{2N} \to \mathbb{R}^{2N}$ the linear application that invert the velocity direction

$$\boldsymbol{\rho}_0 \begin{pmatrix} \mathbf{q} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} I_N & 0_N \\ 0_N & -I_N \end{pmatrix} \begin{pmatrix} \mathbf{q} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{q} \\ -\mathbf{p} \end{pmatrix}.$$

Invoking the Picard-Lindelöf theorem, we define the concept of a reversible ODE and study the reversibility of Hamiltonian ODEs.

Definition 2.2.7 (Reversible ODE)**.** *The ODE $\dot{\mathbf{y}} = \boldsymbol{\mathcal{F}}(\mathbf{y})$ defined in* (2.11) *is reversible or $\boldsymbol{\rho}_0$-reversible if*

$$\forall \mathbf{y} \in \mathbb{R}^d,\ \boldsymbol{\rho}_0 \boldsymbol{\mathcal{F}}(\mathbf{y}) = -\boldsymbol{\mathcal{F}}(\boldsymbol{\rho}_0 \mathbf{y}).$$

Property 2.2.8. *Let $\mathscr{H}(\boldsymbol{\rho}_0\mathbf{u}) = \mathscr{H}(\mathbf{u})$ hold for all $\mathbf{u} \in \mathbb{R}^{2N}$, then the Hamiltonian ODE $\dot{\mathbf{u}} = \mathcal{J}_{2N}\nabla_{\mathbf{u}}\mathscr{H}(\mathbf{u})$ defined in* (2.9) *is reversible.*

Proof. Provided Def. 2.2.7 of a reversible ODE, we have to prove that

$$\forall \mathbf{u} \in \mathbb{R}^{2N},\ \boldsymbol{\rho}_0 \mathcal{J}_{2N} \nabla_{\mathbf{u}} \mathscr{H}(\mathbf{u}) = -\mathcal{J}_{2N} \nabla_{\boldsymbol{\rho}_0 \mathbf{u}} \mathscr{H}(\boldsymbol{\rho}_0 \mathbf{u}).$$

Considering the LHS; for all $\mathbf{u} \in \mathbb{R}^{2N}$

$$\boldsymbol{\rho}_0 \mathcal{J}_{2N} \nabla_{\mathbf{u}} \mathscr{H}(\mathbf{u}) = \begin{pmatrix} 0_N & I_N \\ I_N & 0_N \end{pmatrix} \begin{pmatrix} \nabla_{\mathbf{q}} \mathscr{H}(\mathbf{u}) \\ \nabla_{\mathbf{p}} \mathscr{H}(\mathbf{u}) \end{pmatrix} = \begin{pmatrix} \nabla_{\mathbf{p}} \mathscr{H}(\mathbf{u}) \\ \nabla_{\mathbf{q}} \mathscr{H}(\mathbf{u}) \end{pmatrix}.$$

Continuing with the RHS

$$-\mathcal{J}_{2N} \nabla_{\boldsymbol{\rho}_0 \mathbf{u}} \mathscr{H}(\boldsymbol{\rho}_0 \mathbf{u}) = \begin{pmatrix} 0_N & -I_N \\ I_N & 0_N \end{pmatrix} \begin{pmatrix} \nabla_{\mathbf{q}} \mathscr{H}(\mathbf{u}) \\ \nabla_{-\mathbf{p}} \mathscr{H}(\mathbf{u}) \end{pmatrix} = \begin{pmatrix} \nabla_{\mathbf{p}} \mathscr{H}(\mathbf{u}) \\ \nabla_{\mathbf{q}} \mathscr{H}(\mathbf{u}) \end{pmatrix},$$

which shows that LHS=RHS. □

In practice, the condition $\mathscr{H}(\boldsymbol{\rho}_0\mathbf{u}) = \mathscr{H}(\mathbf{u})$ does not impose any significant restriction. Indeed, in physical systems, $\mathscr{H}$ typically depends on $\|\mathbf{p}\|^2$ through the kinetic energy, rather than directly on $\mathbf{p}$.

We can derive an expression of Def. 2.2.7 in terms of flow $\mathbf{\Phi}_t$; the system is reversible if

$$(\boldsymbol{\rho}_0 \circ \mathbf{\Phi}_t)^2 = \mathrm{id} \iff \boldsymbol{\rho}_0 \circ \mathbf{\Phi}_t = \mathbf{\Phi}_{-t} \circ \boldsymbol{\rho}_0,$$

as $\mathbf{\Phi}_t$ is a diffeomorphism. We illustrate this property on the simple gravity pendulum as shown in Fig. 2.4. Starting from an initial condition $\mathbf{u}(0)$, we compute $(\boldsymbol{\rho}_0 \circ \mathbf{\Phi}_t)\mathbf{u}(0)$ and $(\mathbf{\Phi}_{-t} \circ \boldsymbol{\rho}_0)\mathbf{u}(0)$ and we observe it leads to the same endpoint $\boldsymbol{\rho}_0\mathbf{u}(t)$. Details on the numerical integration are presented in Sec. 2.2.3.

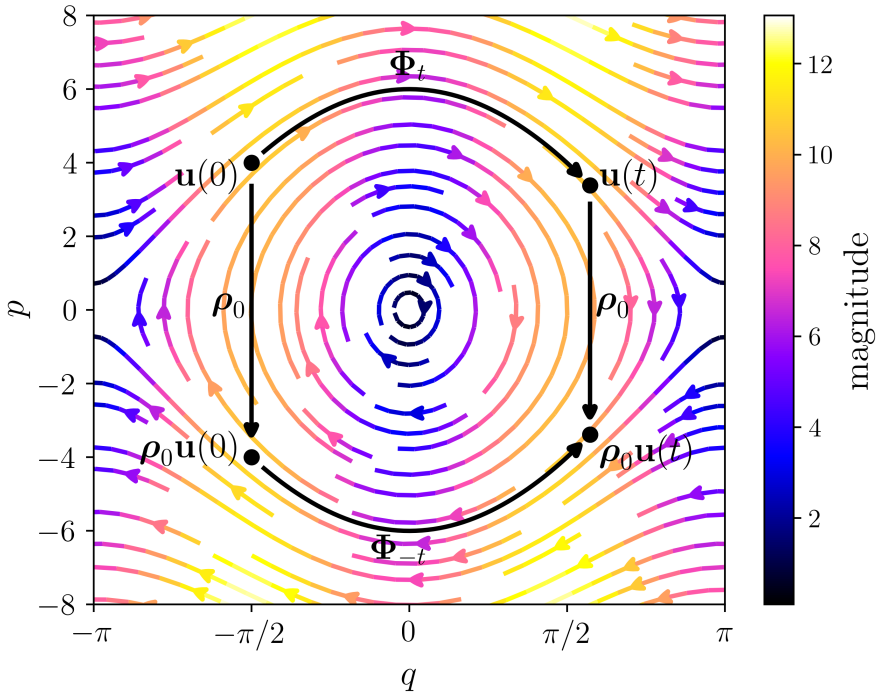


Figure 2.4: (Simple gravity pendulum) Streamline plot for $m = l = 1, g = 10$. The color-map represents the values of the magnitude $\|\dot{u}\|_2$. Superimposed, a representation of reversibility showing that $\boldsymbol{\rho}_0 \circ \mathbf{\Phi}_t = \mathbf{\Phi}_{-t} \circ \boldsymbol{\rho}_0$ for an initial condition $\mathbf{u}(0)$, in black.

The Hamiltonian flow is symplectic Lastly, yet significantly, we give an overview of some geometric results related to the flow of a Hamiltonian ODE. These results will be useful for the construction of numerical methods and reduced models that preserve the Hamiltonian structure, as will be presented in Secs. 2.2.3 and 2.3. First, we define the mappings that preserve the symplectic structure.

Definition 2.2.9 (Symplectic map & symplectic matrix)**.** *We consider two open sets $X \subset \mathbb{R}^{2n}$ and $Y \subset \mathbb{R}^{2m}$ such that $n \leq m$. A differentiable function $f : X \to Y$ is a symplectic map if the Jacobian matrix $Df(x) \in \mathcal{M}_{2m,2n}(\mathbb{R})$ of f is a symplectic matrix i.e.*

$$\forall x \in X,\ (Df(x))^T\, \mathcal{J}_{2m}\, (Df(x)) = \mathcal{J}_{2n}.$$

There exists a strong connection between symplectic maps and Hamiltonian ODEs flows, as established in the property below.

Property 2.2.10. *The flow has the following properties*

1. *if the flow is symplectic, then it conserves the volume,*
2. *the flow $\mathbf{\Phi}_t$ of a canonical Hamiltonian ODE in Def. 2.2.1 is symplectic i.e.*

$$\forall t \geq 0,\ \forall \mathbf{u} \in \mathbb{R}^{2N},\ (D\mathbf{\Phi}_t(\mathbf{u}))^T\, \mathcal{J}_{2N}\, (D\mathbf{\Phi}_t(\mathbf{u})) = \mathcal{J}_{2N}.$$

3. *if the flow of an ODE $\dot{\mathbf{y}} = \boldsymbol{\mathcal{F}}(\mathbf{y})$ is symplectic and $\boldsymbol{\mathcal{F}} \in C^1(\mathbb{R}^{2N})$ then the system is Hamiltonian i.e. there exists $\mathscr{H} \in C^1(\mathbb{R}^{2N}, \mathbb{R})$ such that $\forall y \in \mathbb{R}^{2N}, \boldsymbol{\mathcal{F}}(\mathbf{y}) = \mathcal{J}_{2N} \nabla_{\mathbf{y}} \mathscr{H}(\mathbf{y})$.*

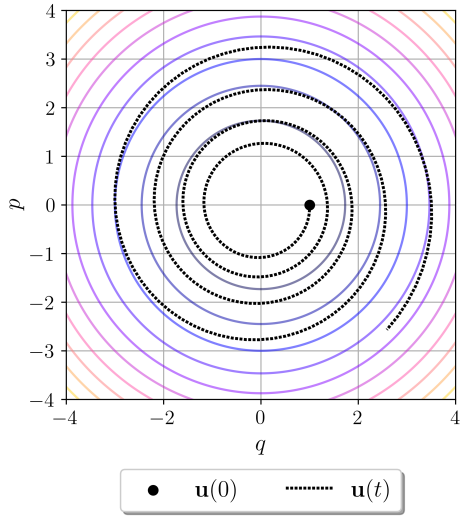
Assertion (1) highlights the geometric implications of symplecticity derived from Liouville's theorem. Thus, symplecticity preserves more than just the volume but it is a key property. Assertions (2) and (3) are mutually converse : there is a deep connection between geometry (symplecticity) and dynamics (Hamiltonian structure) as shown by the Poisson bracket in Eq. (2.5). Symplecticity is a core attribute of Hamiltonian systems that guarantees the presence of numerous invariants that guide the evolution of the system. More details and proofs can be found in [4, 5].

2.2.3 Numerical methods for time integration

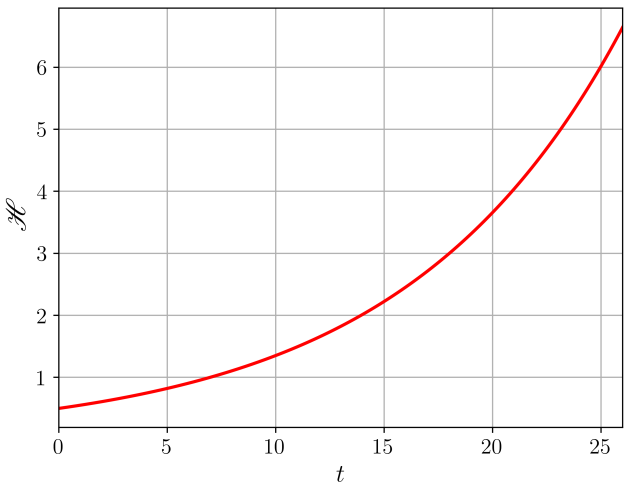
The results on Hamiltonian systems described in Sec. 2.2.2 provide substantial guarantees in terms of long-term stability [16, 17]. As we will briefly see in this section, special care must be taken to ensure that these properties are preserved at the discrete level, thus guaranteeing good stability properties and ensuring that our numerical results behave in a way that is faithful to the physics. First, we introduce an uniform temporal discretization of $[0, T]$ with $n_t + 1$ interval of size $\Delta t > 0$ and denote $t^n = n\Delta t, n \in \{0, \dots, n_t\}$ the n-th time-step. The approximate numerical solution at the n-th time-step is $\mathbf{u}^n \approx \mathbf{u}(t^n; \mu)$, and $\mathbf{u}^0 = \mathbf{u}_{\text{init}}$. We then integrate the ODE Eq. (2.9) on each time interval $[t_n, t_{n+1}]$ leading to the integration formula

$$\mathbf{u}(t^{n+1}; \mu) = \mathbf{u}(t^n; \mu) + \int_{t^n}^{t^{n+1}} \mathcal{J}_{2N} \nabla_{\mathbf{u}} \mathscr{H}(\mathbf{u}(t; \mu))\, dt. \tag{2.12}$$

Then, applying a quadrature rule on Eq. (2.12) yields a time integrator. For instance, with the left rectangle rule, we derive the explicit Euler scheme $\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t\, \mathcal{J}_{2N} \nabla_{\mathbf{u}} \mathscr{H}(\mathbf{u}^n)$. By integrating the harmonic oscillator using this scheme, we get the results in Fig. 2.5. More precisely, in Fig. 2.5a, we remark that the numeric solution drifts away from the analytical solution, which is supposed to be a circle. It is due to the fact that this scheme does not preserve the symplectic structure of ODE, as we can note in Fig. 2.5b that the Hamiltonian $\mathscr{H}$ grows linearly in time instead of being constant over time.



(a) Numeric solution $\mathbf{u}(t)$ in the phase space.



(b) Evolution of $\mathscr{H}$.

Figure 2.5: (Harmonic oscillator) Euler explicit time integration of the system with parameters $T = 26, \Delta t = 1 \times 10^{-1}, \omega = 1$ and $\mathbf{u}(0) = (1\ 0)^T$. Left: numeric solution $(q, p)(t)$ in the phase space in black. Colored circles indicate the iso-values of $\mathscr{H}$, with warmer colors corresponding to larger values. Right: evolution of the energy $\mathscr{H}(q(t), p(t))$ with respect to time $t \in [0, T]$.

This phenomenon can be better understood in term of the numerical flow $\mathbf{\Psi}_{\Delta t}$ defined such that

$$\mathbf{\Psi}_{\Delta t} : \mathbf{u}^n \longmapsto \mathbf{u}^{n+1}.$$

Ideally, this flow is an approximation (in some sense) of the actual continuous flow $\mathbf{\Psi}_{\Delta t} \approx \mathbf{\Phi}_{\Delta t}$. A numerical flow is said to be of order $\mathfrak{p}$ accurate with $\mathfrak{p} \geq 1$ if

$$\mathbf{\Psi}_{\Delta t} = \mathbf{\Phi}_{\Delta t} + \mathcal{O}(\Delta t^{\mathfrak{p}+1}).$$

The Euler explicit numerical flow $\mathbf{\Psi}^{\text{Euler}}_{\Delta t}(\mathbf{u}^n) = \mathbf{u}^n + \Delta t \mathcal{J}_{2N} \nabla_{\mathbf{u}} \mathscr{H}(\mathbf{u}^n)$ is first order ($\mathfrak{p} = 1$) accurate. The problem lies rather in the fact that this flow is not symplectic. We can compute the discrete energy $\mathscr{H}^n$ at time t_n as a function of its initial value $\mathscr{H}^0$; we find a dependence in time $\mathscr{H}^n = \mathscr{H}^0 + \mathcal{O}(\Delta t \times t_n)$. As a result, $\mathscr{H}^n$ has a linear growth. In general, the discrete Hamiltonian from a non-symplectic flow of order $\mathfrak{p}$ grows for any time step Δt and any order $\mathfrak{p}$ [4, prop. 12.17]

$$\mathscr{H}^n = \mathscr{H}^0 + \mathcal{O}(t_n (\Delta t)^{\mathfrak{p}}).$$

Of course, the lack of symplecticity also implies more than just an energy drift, for instance it leads to a progressive degradation of the dynamics quality with respect to physical behaviors, a possible absence of time reversibility, a distortion of the phase-space and a deficiency in long-term stability. To tackle these issues, many symplectic schemes have been developed to preserve the symplectic properties of Hamiltonian systems. A thorough description of these methods is given by E. Hairer, G. Wanner and C. Lubich in [18]. From Def. 2.2.9, we remind that $\mathbf{\Psi}_{\Delta t}$ is symplectic if its Jacobian matrix is symplectic

$$\forall \mathbf{u}^n \in \mathbb{R}^{2N},\ \left(D\mathbf{\Psi}_{\Delta t}(\mathbf{u}^n)\right)^T \mathcal{J}_{2N} \left(D\mathbf{\Psi}_{\Delta t}(\mathbf{u}^n)\right) = \mathcal{J}_{2N}.$$

For instance, a symplectic scheme leads to a bounded energy error over time. Indeed, for a symplectic scheme of order $\mathfrak{p}$, it holds [4] that

$$\forall \mathfrak{m} > \mathfrak{p},\ \mathscr{H}^n = \mathscr{H}^0 + \mathcal{O}((\Delta t)^{\mathfrak{p}} + t_n (\Delta t)^{\mathfrak{m}+1}),$$

meaning that $\mathscr{H}$ does not grow over time anymore. Thus, it is preserved up to $\mathcal{O}((\Delta t)^{\mathfrak{p}})$. In this manuscript, we use two different symplectic schemes. The implicit midpoint scheme,

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \mathcal{J}_{2N} \nabla_{\mathbf{u}} \mathscr{H}\left(\frac{\mathbf{u}^n + \mathbf{u}^{n+1}}{2}\right), \tag{2.13}$$

is a second-order accurate symplectic scheme. The Störmer-Verlet scheme

$$\begin{aligned}
\mathbf{q}^{n+\frac{1}{2}} &= \mathbf{q}^n + \frac{\Delta t}{2} \nabla_{\mathbf{p}} \mathscr{H}\left(\mathbf{q}^{n+\frac{1}{2}}, \mathbf{p}^n\right), \\
\mathbf{p}^{n+1} &= \mathbf{p}^n - \frac{\Delta t}{2} \left[\nabla_{\mathbf{q}} \mathscr{H}\left(\mathbf{q}^{n+\frac{1}{2}}, \mathbf{p}^n\right) + \nabla_{\mathbf{q}} \mathscr{H}\left(\mathbf{q}^{n+\frac{1}{2}}, \mathbf{p}^{n+1}\right)\right], \\
\mathbf{q}^{n+1} &= \mathbf{q}^{n+\frac{1}{2}} + \frac{\Delta t}{2} \nabla_{\mathbf{p}} \mathscr{H}\left(\mathbf{q}^{n+\frac{1}{2}}, \mathbf{p}^{n+1}\right),
\end{aligned} \tag{2.14}$$

and its variant

$$\begin{aligned}
\mathbf{p}^{n+\frac{1}{2}} &= \mathbf{p}^n - \frac{\Delta t}{2} \nabla_{\mathbf{q}} \mathscr{H}\left(\mathbf{q}^n, \mathbf{p}^{n+\frac{1}{2}}\right), \\
\mathbf{q}^{n+1} &= \mathbf{q}^n + \frac{\Delta t}{2} \left[\nabla_{\mathbf{p}} \mathscr{H}\left(\mathbf{q}^n, \mathbf{p}^{n+\frac{1}{2}}\right) + \nabla_{\mathbf{p}} \mathscr{H}\left(\mathbf{q}^{n+1}, \mathbf{p}^{n+\frac{1}{2}}\right)\right], \\
\mathbf{p}^{n+1} &= \mathbf{p}^{n+\frac{1}{2}} - \frac{\Delta t}{2} \nabla_{\mathbf{q}} \mathscr{H}\left(\mathbf{q}^{n+1}, \mathbf{p}^{n+\frac{1}{2}}\right),
\end{aligned}$$

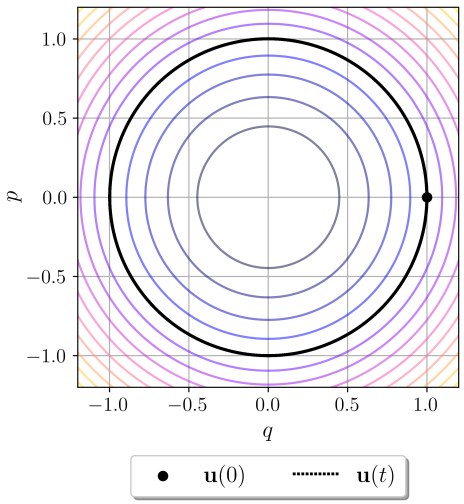
are also second-order accurate. Both of these schemes are of the same order and implicit: the implicit midpoint in Eq. (2.13) involves a single implicit step, while the Störmer-Verlet scheme in Eq. (2.14) requires two implicit steps. In practice, the former would generally be preferred, except when dealing with a separable Hamiltonian. If the Hamiltonian is separable i.e. $\mathscr{H}(\mathbf{q},\mathbf{p}) = \mathcal{K}(\mathbf{p}) + \varphi(\mathbf{q})$, Eq. (2.14) takes the form

$$\begin{aligned} \mathbf{q}^{n+\frac{1}{2}} &= \mathbf{q}^n + \frac{\Delta t}{2}\nabla\mathcal{K}(\mathbf{p}^n), \\ \mathbf{p}^{n+1} &= \mathbf{p}^n - \Delta t\nabla\varphi(\mathbf{q}^{n+\frac{1}{2}}), \\ \mathbf{q}^{n+1} &= \mathbf{q}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\nabla\mathcal{K}(\mathbf{p}^{n+1}), \end{aligned} \tag{2.15}$$

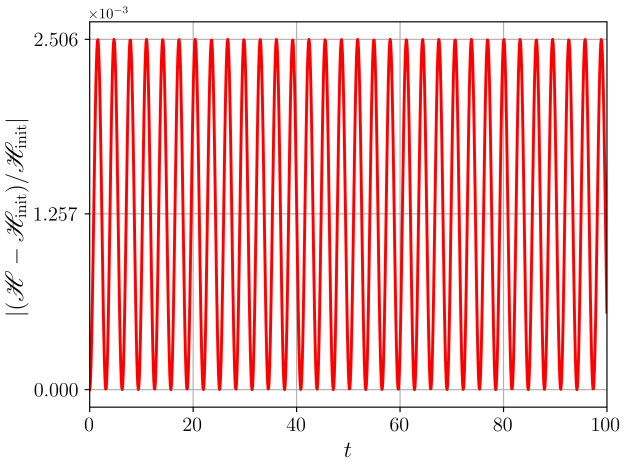
and the numerical scheme is completely explicit. Thus, if $\mathcal{K}(\mathbf{p}^n) = \frac{1}{2m}\|\mathbf{p}^n\|^2$, we have $\nabla\mathcal{K}(\mathbf{p}^n) = \frac{1}{m}\mathbf{p}^n$. Consequently, it becomes more cost-effective than an implicit midpoint scheme. More details on symplectic schemes can be found in [4, 18].

Remark. *Both the implicit midpoint schemes and Störmer-Verlet schemes have time reversible flow, these methods are called symmetric. Nonetheless, there exists symplectic schemes that are not symmetric, and conversely symmetric schemes that are not symplectic [18, p. VI.4.2].*

We conclude this section with a few numerical examples. We revisit the previous example of the harmonic oscillator and compute a numerical solution with a Störmer-Verlet scheme. We increase the final time $T = 100$ and the other parameters remain unchanged. We observe the results in Fig. 2.6. On the left Fig. 2.6a, we verify that the numerical trajectory closely approximates a circle of the correct radius and center, and remains near it over time. On the right Fig. 2.6b, we observe that the Hamiltonian is bounded up to an error of $\mathcal{O}((\Delta t)^2)$ where $\Delta t = 1\times10^{-1}$, to within the error of the numerical flow. The most important aspect is that the amplitude of the error does not increase over time.



(a) Numeric solution $\mathbf{u}(t)$ in the phase space.



(b) Evolution of the relative error on $\mathscr{H}$.

Figure 2.6: (Harmonic oscillator) Störmer-Verlet time integration of the system with parameters $T = 100, \Delta t = 1\times10^{-1}, \omega = 1$ and $\mathbf{u}(0) = (1\ 0)^T$. Left: numeric solution $(q,p)(t)$ in the phase space in black. Colored circles indicate the iso-values of $\mathscr{H}$, with warmer colors corresponding to larger values. Right: evolution of the relative error on the energy $\mathscr{H}(q(t),p(t))$ with respect to time $t \in [0,T]$.

At this stage, we introduced the family of canonical Hamiltonian ODEs. They are ubiquitous in physics and other fields. Their structure is rich in symmetries and invariants. Conserving these properties in a discrete setting with tailored numerical methods allows to obtain more accurate, stable, and efficient long-term simulations. Properties and numerical methods we have presented can be extended to non-canonical ODEs ($\mathcal{J} := \mathcal{J}(\mathbf{u})$) and non autonomous Hamiltonians as detailed in [18].

Since these ODEs arise from the semi-discretization of PDEs, they have a often large dimension and they are complex to solve numerically. As a consequence, we now turn to structure-preserving model order reduction techniques for Hamiltonian systems, aiming to ease the associated computational burden.

2.3 Linear model order reduction for Hamiltonian ODEs

Solving $2N$-dimensional ODE of the form

$$\begin{cases} \dfrac{d\,\mathbf{u}(t;\mu)}{d\,t} = \mathcal{J}_{2N}\nabla_{\mathbf{u}}\mathscr{H}(\mathbf{u}(t;\mu)), \\ \mathbf{u}(0;\mu) = \mathbf{u}_{\text{init}}(\mu), \end{cases} \tag{2.9}$$

often proves to be numerically challenging, particularly with a large dimension $2N \gg 1$ and/or a Hamiltonian gradient $\nabla_{\mathbf{u}}\mathscr{H}$ which can be expensive to compute. This is particularly the case for optimization algorithms, where multiple numerical evaluations of the solutions to Eq. (2.9) are required, or in real-time control settings, where numerical solutions must be computed on the fly. We are faced with the need to compute numerous solutions $\mathbf{u}(t;\mu)$ for a range of times and parameters $(t,\mu) \in \mathcal{T} \times \Gamma \subset \mathbb{R} \times \mathbb{R}^p$ within a reasonable time. This requires speeding up the calculations at the undeniable cost of some accuracy. Model Order Reduction (MOR) refers to a technique used to reduce the numerical complexity/cost of mathematical models. It aims to build a Reduced Order Model (ROM), i.e. an ODE of diminished complexity with solutions closely related to the Full Order Model (FOM), i.e. Eq. (2.9).

In practice, model order reduction is divided in two stages called the offline/online decomposition to implement an efficient reduction algorithm.

(i) offline stage: we build the FOM and the ROM. Most importantly, we precompute quantities. It means that we generate snapshots, we build the reduced basis (more details later) and we compress the full order operators onto the reduced space. This phase is computationally expensive but parametrically independent, and it is performed once.

(ii) online stage: we aim to quickly solve the reduced parameters with new parameters and evaluate outputs of interest, if any. Hence, we use the precomputed quantities from the offline stage to obtain a much faster computational speed. This stage is done for every new parameter.

Then, A MOR technique is said to be structure-preserving when it retains some properties of the FOM, e.g. the Hamiltonian structure. We will address two challenges which are to (i) preserve the Hamiltonian structure and (ii) embed dynamics nonlinearities.

We consider the symplectic solution manifold

$$\mathcal{M} := \{\mathbf{u}(t;\mu) \,|\, (t,\mu) \in \mathcal{T} \times \Gamma\} \subset \mathbb{R}^{2N},$$

formed by the values of the solutions of Eq. (2.9) for various times and parameters, the manifold structure resulting from the Picard-Lindelöf theorem with some regularity assumptions on the Hamiltonian $\mathscr{H}$. The fundamental hypothesis of MOR is that this manifold $\mathcal{M}$ can be approximated with a trial manifold $\widehat{\mathcal{M}}$ that reads

$$\widehat{\mathcal{M}} := \mathbf{u}_{\text{ref}}(\mu) + \{\mathcal{D}\left[\bar{\mathbf{u}}(t;\mu)\right] \,|\, (t,\mu) \in \mathcal{T} \times \Gamma\},$$

where $\mathcal{D} : \mathbb{R}^{2K} \to \mathbb{R}^{2N}$ is a reconstruction/decoding operator or decoder, $\mathbf{u}_{\text{ref}}(\mu) \in \mathbb{R}^{2N}$ is a reference state, and $\bar{\mathbf{u}}(t;\mu) \in \mathbb{R}^{2K}$ is the reduced state or latent variable. Put differently, we search for an approximation $\hat{\mathbf{u}}(t;\mu)$ of the full state $\mathbf{u}(t;\mu)$ such that

$$\hat{\mathbf{u}}(t;\mu) := \mathbf{u}_{\text{ref}}(\mu) + \mathcal{D}\left[\bar{\mathbf{u}}(t;\mu)\right] \approx \mathbf{u}(t;\mu).$$

For convenience, we define an encoding operator $\mathcal{E} : \mathbb{R}^{2N} \to \mathbb{R}^{2K}$ or encoder such that

$$\bar{\mathbf{u}}(t;\mu) = \mathcal{E}\left[\mathbf{u}(t;\mu)\right].$$

It is a pseudo-inverse of the decoder $\mathcal{D}$. The reference state often represents a state of equilibrium and the reduced model is typically formulated in terms of deviation from this reference. In the following, we set $\mathbf{u}_{\text{ref}}(\mu) = \mathbf{0}$, as it is done in [19, 20]. Otherwise, it is only needed when reconstructing the full order approximation. In certain cases, the approximability of $\mathcal{M}$ can be evaluated by examining the decay of the Kolmogorov N-width, as illustrated in the context of wave propagation problems in [21].

In this section, we present several techniques for constructing Hamiltonian reduced states. In the first part, we present the Proper Symplectic Decomposition (PSD) [22] method, as first introduced by L. Peng and K. Mohseni in 2015. It is the symplectic variant of the well-known Proper Orthogonal Decomposition (POD) [23, 24] method. POD is a cornerstone of model reduction. However, in the following we focus on the PSD. It closely resembles the POD but incorporates additional structural constraints. We highlight these key differences throughout the discussion.

First, we present the method, and then we examine the strengths and limitations of this approach. Second, we discuss possible improvements to overcome these limitations, as well as alternative model reduction techniques available in the literature, for both standard and Hamiltonian model reduction.

2.3.1 Proper Symplectic Decomposition

The PSD method assume that the trial manifold is a linear subspace or a vector subspace defined as $\widehat{\mathcal{M}} = \text{span}(\mathbf{a}_i, i \in \{1, \dots, 2K\})$ or, equivalently

$$\hat{\mathbf{u}}(t;\mu) = \mathcal{D}\left[\bar{\mathbf{u}}(t;\mu)\right] = A\bar{\mathbf{u}}(t;\mu),$$

with $A \in \mathcal{M}_{2N,2K}(\mathbb{R})$ the corresponding decoding matrix. Expressed differently, the dynamics of the state $\mathbf{u}(t;\mu)$ can be more or less captured in a $2K$-dimensional subspace of basis $\{\mathbf{a}_i\}_{i=1}^{2K}$. Thus, we impose that the decoder $\mathcal{D}(\bar{\mathbf{u}}) = A\bar{\mathbf{u}}$ is a symplectic map as defined in Def. 2.2.9 such that

$$A \in \mathcal{S}_{2N,2K}(\mathbb{R}) := \left\{A \in \mathcal{M}_{2N,2K}(\mathbb{R}) \middle| A^T \mathcal{J}_{2N} A = \mathcal{J}_{2K}\right\},$$

where $\mathcal{S}_{2N,2K}(\mathbb{R})$ is the symplectic Stiefel manifold, which is also the set of symplectic matrices. This is mandatory to preserve the symplectic structure in the reduced system. At last, we must define the corresponding encoder $\mathcal{E}$.

Definition 2.3.1 (Symplectic inverse)**.** *The symplectic inverse of $A \in \mathcal{S}_{2N,2K}(\mathbb{R})$ is the matrix A^+ defined as*

$$A^+ := \mathcal{J}_{2K}^T A^T \mathcal{J}_{2N}.$$

and such that $A^+ A = I_{2K}$.

Thus, we define the encoder

$$\bar{\mathbf{u}}(t;\mu) = \mathcal{E}\left[\mathbf{u}(t;\mu)\right] = A^+ \mathbf{u}(t;\mu).$$

Furthermore AA^+ is the symplectic projection operator onto $\mathrm{Im}(A) = \widehat{\mathcal{M}}$.

After defining the encoding and decoding operators, the next step is to derive the reduced model. We summarize the key elements in the following definition, which we will prove subsequently.

Property 2.3.2 (PSD's Hamiltonian reduced model [22])**.** *We consider an Hamiltonian FOM of solution $\mathbf{u}(t;\mu)$ with time $t \in \mathcal{T}$ and parameters $\mu \in \Gamma$ that reads*

$$\begin{cases} \dfrac{d\,\mathbf{u}(t;\mu)}{d\,t} = \mathcal{J}_{2N}\nabla_{\mathbf{u}}\mathscr{H}(\mathbf{u}(t;\mu)), \\ \mathbf{u}(0;\mu) = \mathbf{u}_{init}(\mu), \end{cases} \tag{2.9}$$

and the encoder $\mathcal{E}(\mathbf{u}) = A^+\mathbf{u} = \bar{\mathbf{u}}$ and its symplectic inverse the decoder $\mathcal{D}(\bar{\mathbf{u}}) = A\bar{\mathbf{u}}$ with the symplectic matrix $A \in \mathcal{S}_{2N,2K}(\mathbb{R})$. The reduced Hamiltonian model of solution $\bar{\mathbf{u}}(t;\mu)$ is given by the symplectic Galerkin projection of Eq. (2.9)

$$\begin{cases} \dfrac{d\,\bar{\mathbf{u}}(t;\mu)}{d\,t} = \mathcal{J}_{2K}\nabla_{\bar{\mathbf{u}}}\bar{\mathscr{H}}(\bar{\mathbf{u}}(t;\mu)), \\ \bar{\mathbf{u}}(0;\mu) = A^+\mathbf{u}_{init}(\mu), \end{cases}, \tag{2.16}$$

with $\bar{\mathscr{H}} := \mathscr{H} \circ A$ is the reduced Hamiltonian or the Hamiltonian of the ROM.

Proof. We consider the reconstructed solution $\hat{\mathbf{u}}(t;\mu) = A\bar{\mathbf{u}}(t;\mu)$ and we derive

$$\frac{d\,\hat{\mathbf{u}}(t;\mu)}{d\,t} = A\frac{d\,\bar{\mathbf{u}}(t;\mu)}{d\,t}.$$

We define the time-dependent residual $\mathbf{r}(t;\mu)$ of $\hat{\mathbf{u}}$ with respect to the FOM

$$\mathbf{r}(t;\mu) = \frac{d\,\hat{\mathbf{u}}(t;\mu)}{d\,t} - \mathcal{J}_{2N}\nabla_{\mathbf{u}}\mathscr{H}(\hat{\mathbf{u}}(t;\mu)).$$

Arguing the symplectic Galerkin projection, the symplectic projection of the residual onto the subspace $\widehat{\mathcal{M}}$ must vanish

$$\begin{aligned} \mathbf{0} &:= A^+\mathbf{r}(t;\mu), \\ &= A^+A\frac{d\,\bar{\mathbf{u}}(t;\mu)}{d\,t} - A^+\mathcal{J}_{2N}\nabla_{\mathbf{u}}\mathscr{H}(\hat{\mathbf{u}}(t;\mu)), \\ &= \frac{d\,\bar{\mathbf{u}}(t;\mu)}{d\,t} - \mathcal{J}_{2K}A^T\nabla_{\mathbf{u}}\mathscr{H}(A\bar{\mathbf{u}}(t;\mu)), \\ &= \frac{d\,\bar{\mathbf{u}}(t;\mu)}{d\,t} - \mathcal{J}_{2K}\nabla_{\bar{\mathbf{u}}}\mathscr{H}(A\bar{\mathbf{u}}(t;\mu)), \end{aligned}$$

using the chain rule. We directly obtain Eq. (2.16). □

Remark. *Although the reduced model is an ODE of dimension $2K$, $K \ll N$, the reduced Hamiltonian $\bar{\mathscr{H}}$ still depends on the original Hamiltonian $\mathscr{H}$. As a result, the computational cost generally remains high.*

In the POD method, the symplectic matrix A is replaced by an orthogonal matrix $A^TA = I_{2K}$. Thus, the symplectic inverse is replaced by the transpose and the operator AA^T is the orthogonal projection. The Galerkin projection supposes then that the orthogonal projection of the residual onto the subspace vanishes.

To conclude this part, we must address the actual construction of the basis $\{\mathbf{a}_i\}_{i=1}^{2K}$. In general, we do not have any expression for the solution manifold $\mathcal{M}$. We only have access to $P > 0$ data points $\{\mathbf{u}(t_j;\mu_j)\}_{j=1}^{P}$ called FOM snapshots. They are solutions of the FOM Eq. (2.9) for several times and parameters $(t_j,\mu_j) \in \mathcal{T}\times\Gamma, j \in \{1,\dots,P\}$. As a consequence, we aim to find the best basis $\{\mathbf{a}_i\}_i$ to express the shape and variability of the point cloud $\{\mathbf{u}(t_j;\mu_j)\}_j$. In other words, we join together the snapshots in a large snapshot matrix U

$$U = \begin{bmatrix}\mathbf{u}(t_1;\mu_1) & \dots & \mathbf{u}(t_P;\mu_P)\end{bmatrix} \in \mathcal{M}_{2N,P}(\mathbb{R}).$$

The best matrix $A \in \mathcal{S}_{2N,2K}(\mathbb{R})$ is a solution to the optimization problem corresponding to minimizing the reconstruction error on the snapshots in a least square sense. It reads

$$\min_{A^T\mathcal{J}_{2N}A=\mathcal{J}_{2K}} \left\|U - AA^+U\right\|_F. \tag{2.17}$$

2.3.2 Building the projection matrix

Computing a direct solution of Eq. (2.17) can be very expensive with a large $N \gg 1$. In this manuscript, we employ two of the methods presented in [22] to compute an approximated optimal solution. These methods are described in detail below. We recall the factorization technique for rectangular matrices: the Singular Value Decomposition (SVD) [25]. It can be thought of as a generalization of the eigendecomposition to non-square or nondiagonalizable matrices.

Definition 2.3.3 (Singular Value Decomposition)**.** *The Singular Value Decomposition (SVD) of a matrix $X \in \mathcal{M}_{m,n}(\mathbb{C})$ is the factorization*

$$X = W\Sigma V^*$$

with $W \in \mathcal{U}_m(\mathbb{C}), V \in \mathcal{U}_n(\mathbb{C})$ unitary matrices i.e. $WW^* = W^*W = I_m$ and $VV^* = V^*V = I_n$ and $\Sigma \in \mathcal{M}_{m,n}(\mathbb{C})$ is a rectangular diagonal matrix, V^* is the conjugate-transpose of V. The diagonal values $\sigma_i = \Sigma_{i,i}$ are sorted in descending order. W and V columns $\{\mathbf{w}_i\}_i$ and $\{\mathbf{v}_i\}_i$ respectively are orthonormal bases. With $\operatorname{rank}(X) \leq \min(m,n)$ the rank of X, this decomposition is equivalent to

$$X = \sum_{i=1}^{\operatorname{rank}(X)} \sigma_i \mathbf{w}_i \mathbf{v}_i^*$$

arguing $\sigma_i = 0$, $i > \operatorname{rank}(X)$. Σ is uniquely determined by X but not W and V. σ_i is called a singular value, $\mathbf{w}_i, \mathbf{v}_i$ are the corresponding left singular vector and right singular vector, respectively. If X is real-valued, the same decomposition exists with orthogonal matrices and the conjugate-transpose becomes the transpose.

The SVD is closely related to dimensionality reduction and low-rank approximations of matrices. We admit the following result, which will be useful for the construction of A.

Theorem 2.3.4 (Low-rank approximation - Eckart–Young–Mirsky theorem)**.** *We consider $X \in \mathcal{M}_{m,n}(\mathbb{C})$. The best rank $r \leq \operatorname{rank}(X)$ approximation in Frobenius norm of X is given by the truncation of the SVD given in Def. 2.3.3 of X, keeping the r largest singular values. The truncation $X_r \in \mathcal{M}_{m,n}(\mathbb{C})$ is defined as*

$$X_r = \sum_{i=1}^{r} \sigma_i \mathbf{w}_i \mathbf{v}_i^* \iff X_r = W_r\Sigma_r V_r^*,$$

with $\Sigma_r = \mathrm{diag}(\sigma_1, \dots, \sigma_r)$*, and* W_r, V_r *represent the matrices* W, V *truncated to their first* r *columns, respectively. In other words,* X_r *is solution of*

$$X_r = \underset{\substack{X' \in \mathcal{M}_{m,n}(\mathbb{C}) \\ \mathrm{rank}(X') \leq r}}{\arg\min} \|X - X'\|_F.$$

with the Frobenius norm $\|X\|_F = \mathrm{tr}(X^*X)^{1/2} = (\sum_{i,j} |x_{i,j}|^2)^{1/2}$ *where* tr *is the trace operator.*

Viewing X as a point cloud in $\mathbb{R}^{2N}$, the SVD extracts a hyperplane spanned by the basis vectors $\{\mathbf{w}_i\}_i$ that captures the dominant directions of variance in the cloud. Thm. 2.3.4 argues that this hyperplane is indeed optimal to express the variance of the point cloud: a projection onto this hyperplane results in a minimal loss of information.

Thereafter, we describe the two above-mentioned methods presented in [22] to compute an approximated optimal solution A of Eq. (2.17), in the form of two properties. To do so, we add some prior on $A \in \mathcal{S}_{2N,2K}(\mathbb{R})$ so that the minimization problem can be solved using an SVD.

Property 2.3.5 (Cotangent lift [22])**.** *Under the assumption that the matrix* A *solution to Eq.* (2.17) *is block diagonal i.e.*

$$A \in \mathcal{S}_{2N,2K}(\mathbb{R}) \cap \left\{ \begin{pmatrix} A_1 & 0 \\ 0 & A_1 \end{pmatrix} \middle| A_1 \in \mathcal{M}_{N,K}(\mathbb{R}) \right\},$$

the optimal solution to the modified Eq. (2.17) *is* $A_1 = W_r := W_{:,:r}$ *with* W_r *the truncated matrix obtained from the SVD of the extended snapshot matrix* U_1 *that reads*

$$U_1 = \begin{bmatrix} \mathbf{q}(t_1;\mu_1) & \dots & \mathbf{q}(t_P;\mu_P) & \mathbf{p}(t_1;\mu_1) & \dots & \mathbf{p}(t_P;\mu_P) \end{bmatrix} \in \mathcal{M}_{N,2P}(\mathbb{R}).$$

Proof. We reformulate Eq. (2.17) starting with the minimized norm. If A is block diagonal, we have

$$\left\| U - AA^TU \right\|_F = \left\| U_1 - A_1A_1^TU_1 \right\|_F.$$

On the constraint, $A \in \mathcal{S}_{2N,2K}(\mathbb{R})$ and $A = \mathrm{diag}(A_1, A_1)$ implies that $A_1^TA_1 = I_K$. Hence, the modified minimization problem derived from Eq. (2.17) that we seek to solve is

$$\min_{A_1^TA_1 = I_K} \left\| U_1 - A_1A_1^TU_1 \right\|_F. \tag{2.18}$$

In term of matrix approximation, $A_1A_1^TU_1$ is the orthogonal projection onto the column space of A_1 i.e. the best approximation of U_1 of at most rank $K \geq \mathrm{rank}(A_1A_1^TU_1)$.

Then, let $U_1 = W\Sigma V^T$ be the SVD of U_1 as given in Def. 2.3.3. Arguing Eckart–Young–Mirsky Thm. 2.3.4, the best approximation of U_1 which is at most rank K is given by the truncated SVD $U_{1,K} = W_K\Sigma_KV_K^T$ such that

$$\|U_1 - U_{1,K}\|_F = \min_{\mathrm{rank}(X') \leq r} \|U_1 - X'\|_F.$$

Thus, we take $A_1 = W_K$ and we denote $r = \mathrm{rank}(X)$

$$A_1A_1^TU_1 = W_KW_K^TU_1 = \sum_{i=1}^{r} \sigma_i W_KW_K^T\mathbf{w}_i\mathbf{v}_i^T.$$

However, $W_KW_K^T$ is the orthogonal projection onto $\mathrm{span}\,(\mathbf{w}_i, i \in \{1, \dots, K\})$, so

$$W_KW_K^T\mathbf{w}_i = \begin{cases} \mathbf{w}_i \text{ if } i \leq K, \\ \mathbf{0} \text{ else.} \end{cases}$$

Then,

$$A_1 A_1^T U_1 = \sum_{i=1}^{K} \sigma_i \mathbf{w}_i \mathbf{v}_i^T = U_{1,K}.$$

Indeed, the best approximation is $U_{1,K}$ and $A_1 = W_K$ is the solution of Eq. (2.18). □

Property 2.3.6 (Complex SVD [22])**.** *Under the assumption that the matrix A solution to Eq.* (2.17) *is block skew-symmetric i.e.*

$$A \in \mathcal{S}_{2N,2K}(\mathbb{R}) \cap \left\{ \begin{pmatrix} A_2 & -B_2 \\ B_2 & A_2 \end{pmatrix} \middle| A_2, B_2 \in \mathcal{M}_{N,K}(\mathbb{R}) \right\},$$

the optimal solution of the modified Eq. (2.17) *is $(A_2, B_2) = (\Re(W_r), \Im(W_r))$ the real, resp. imaginary part of $W_r := W_{:,:r}$, the truncated matrix obtained from the SVD of the complex snapshot matrix U_2 that reads*

$$U_2 = \begin{bmatrix} \mathbf{q}(t_1;\mu_1) + i\mathbf{p}(t_1;\mu_1) & \dots & \mathbf{q}(t_P;\mu_P) + i\mathbf{p}(t_P;\mu_P) \end{bmatrix} \in \mathcal{M}_{N,P}(\mathbb{C}).$$

Proof. The modified minimization problem is

$$\min_{D_2^* D_2 = I_K} \|U_2 - D_2 D_2^* U_2\|_F .$$

with $D_2 = A_2 + iB_2$. The remainder of the proof is the same as in Prop. 2.3.5 proof with complex-valued matrices. □

This raises a natural question: which criteria should guide the choice between the cotangent lift and the complex SVD methods ? The cotangent lift has a clear separation between reduced coordinates $\bar{\mathbf{q}}$ and reduced momenta $\bar{\mathbf{p}}$. When implementing the reduced model, treating separately the reduced position $\mathbf{q} = A_1\bar{\mathbf{q}}$ and momentum $\mathbf{p} = A_1\bar{\mathbf{p}}$ variables may lead to certain simplifications. On the contrary, by considering a broader class of symplectic matrices, the complex SVD might results in a better approximation but it is not clear considering the test cases in [22, 26]. It is more a problem-dependent choice.

Regarding the POD method, the minimization problem given in Eq. (2.17) is the minimization of $\|U - AA^T U\|_F$ under the constraints $A^T A = I_{2K}$: it is a standard least squares problem that can be solved directly via SVD decomposition of U.

2.3.3 Hyper-reduction with symplectic DEIM

As above-mentioned in Sec. 2.3.1, in general, the reduced Hamiltonian $\bar{\mathscr{H}} = \mathscr{H} \circ A$ still depends on the reference Hamiltonian $\mathscr{H}$: the computational burden is increased. Indeed, in addition to computing its gradient $\nabla_{\mathbf{u}}\mathscr{H}$ at each time-step for the time integration, we also need to decompress the reduced variable $\bar{\mathbf{u}}$ and compress back the gradient i.e. computing $A^T \circ \nabla_{\mathbf{u}}\mathscr{H} \circ A$.

Linear Hamiltonian system

Property 2.3.7 (Linear Hamiltonian system - reduced model)**.** *If the Hamiltonian function is of the form $\mathscr{H}(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T D\mathbf{u}$ where $D \in \mathcal{M}_{2N}(\mathbb{R})$ is symmetric i.e. $D = D^T$. Let $B := \mathcal{J}_{2N} D$, the linear Hamiltonian system derived from Eq.* (2.9) *reads*

$$\frac{d\,\mathbf{u}}{d\,t} = B\mathbf{u}.$$

Applying the symplectic projection A as given in Prop. 2.3.2, the reduced model is

$$\frac{d\,\bar{\mathbf{u}}}{d\,t} = \bar{B}\bar{\mathbf{u}} \text{ with } \bar{B} = A^+ BA.$$

The full and reduced solutions, $\mathbf{u}(t;\mu)$ and $\bar{\mathbf{u}}(t;\mu)$, respectively, can be directly computed for all $t \in \mathcal{T}$.

Symplectic Discrete Empirical Interpolation Method (SDEIM) As we can directly compute solutions of a linear Hamiltonian system, an idea to treat the general case is to split the Hamiltonian $\mathscr{H}$ in two parts : (i) a term $\frac{1}{2}\mathbf{u}^T D\mathbf{u}$ as given in in Prop. 2.3.7 and (ii) a nonlinear term $H_N(\mathbf{u})$ such that $\mathscr{H}(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T D\mathbf{u} + H_N(\mathbf{u})$. The subscript N indicates that computation cost of H_N depends on N the FOM dimension. We denote $\boldsymbol{h}_N(\mathbf{u}) := \nabla_{\mathbf{u}} H_N(\mathbf{u})$. The PSD reduced model from Prop. 2.3.2 then reads

$$\frac{d\,\bar{\mathbf{u}}}{d\,t} = \mathcal{J}_{2K} A^T \nabla_{A\bar{\mathbf{u}}} \mathscr{H}(A\bar{\mathbf{u}}) = \mathcal{J}_{2K}(A^T D A)\bar{\mathbf{u}} + \mathcal{J}_{2K} A^T \boldsymbol{h}_N(A\bar{\mathbf{u}}) \tag{2.19}$$

The system's linear term $A^T DA$ is computed once offline and as a result, the numerical evaluation of the linear vector field has a cost of $\mathcal{O}(K)$ instead of $\mathcal{O}(\alpha(N))$ for the full Hamiltonian gradient, where α is some function of N.

The nonlinear term $\boldsymbol{\psi}_N := A^T \circ \boldsymbol{h}_N \circ A$ comes with a cost of $\mathcal{O}(NK + \alpha(N))$ which can quickly become prohibitive. The idea of the DEIM [27] is to project the term $\boldsymbol{\psi}_N$ onto a subspace that approximates it well and which is spanned by a basis of dimension $m \ll N$. In practice, this proper subspace is given by the SVD on snapshots of the nonlinear function $\boldsymbol{\psi}_N$. In this paragraph, we focus on the DEIM's variant for Hamiltonian system called SDEIM [22]. The sole practical difference between these two methods lies in the multiplication of $\boldsymbol{\psi}_N$ by $\mathcal{J}_{2K}$ in Eq. (2.19), which is absent in the DEIM approach.

In details, we consider the snapshot matrix $U_{\boldsymbol{\psi}}$ formed by snapshots of $\boldsymbol{\psi}_N$

$$U_{\boldsymbol{\psi}} = \begin{pmatrix} \boldsymbol{\psi}_N\left[\mathbf{u}(t_1;\mu_1)\right] & \dots & \boldsymbol{\psi}_N\left[\mathbf{u}(t_P;\mu_P)\right] \end{pmatrix} \in \mathcal{M}_{2N,P}(\mathbb{R}).$$

As explained in Sec. 2.3.2, we compute the SVD of $U_{\boldsymbol{\psi}}$ and keep the truncated matrix $A_{\boldsymbol{\psi}} \in \mathcal{M}_{2N,m}(\mathbb{R})$ whose columns are the m first left singular vectors sorted by the magnitude of their singular values in descending order. The low rank approximation of $\boldsymbol{\psi}_N$ on the column space of $A_{\boldsymbol{\psi}}$ then reads

$$\boldsymbol{\psi}_N(\mathbf{u}) \approx A_{\boldsymbol{\psi}} \widehat{\boldsymbol{\psi}}(\mathbf{u})$$

where $\widehat{\boldsymbol{\psi}}(\mathbf{u}) \in \mathbb{R}^m$ are the unknown coefficients of $\boldsymbol{\psi}_N(\mathbf{u})$ on the basis given by the columns of $A_{\boldsymbol{\psi}}$. To determine $\widehat{\boldsymbol{\psi}}(\mathbf{u})$, we remark that the system $\boldsymbol{\psi}_N(\mathbf{u}) = A_{\boldsymbol{\psi}} \widehat{\boldsymbol{\psi}}(\mathbf{u})$ is overdetermined hence we select m appropriates rows of indices $\{i_1, \dots, i_m\}$ such that $\widehat{\boldsymbol{\psi}}(\mathbf{u})$ is uniquely determined by

$$P^T \boldsymbol{\psi}_N(\mathbf{u}) = \left(P^T A_{\boldsymbol{\psi}}\right) \widehat{\boldsymbol{\psi}}(\mathbf{u}),$$

where $P \in \mathcal{M}_{2N,m}(\mathbb{R})$ is a selection matrix

$$P = \begin{pmatrix} \boldsymbol{e}_{i_1} & \dots & \boldsymbol{e}_{i_m} \end{pmatrix},$$

with $\boldsymbol{e}_i$ is the i-th column of the identity matrix. Indeed, P selects those above-mentioned m rows of the overdetermined system. Under the assumption that $P^T A_{\boldsymbol{\psi}}$ is invertible, we can approximate the nonlinear term with

$$\boldsymbol{\psi}_N(\mathbf{u}) \approx A_{\boldsymbol{\psi}} \widehat{\boldsymbol{\psi}}(\mathbf{u}) = A_{\boldsymbol{\psi}} \left(P^T A_{\boldsymbol{\psi}}\right)^{-1} P^T \boldsymbol{\psi}_N(\mathbf{u}).$$

Property 2.3.8 (SDEIM hyper-reduction [22])**.** *We consider a Hamiltonian ODE with a Hamiltonian $\mathscr{H}(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T D\mathbf{u} + H_N(\mathbf{u})$ and its PSD's reduced model as given in Prop. 2.3.2. The SDEIM hyper-reduction results in the approximated reduced model*

$$\frac{d\,\bar{\mathbf{u}}}{d\,t} = \bar{B}\bar{\mathbf{u}} + \mathcal{J}_{2K} W \boldsymbol{h}_m(\bar{\mathbf{u}})$$

with the matrices $\bar{B} := A^+ \mathcal{J}_{2N} DA, W := A_{\boldsymbol{\psi}} \left(P^T A_{\boldsymbol{\psi}}\right)^{-1}$, and $\boldsymbol{h}_m(\bar{\mathbf{u}}) := P^T \boldsymbol{h}_N(A\bar{\mathbf{u}})$.

The primary objective of hyper-reduction is to reduce the numerical complexity while ensuring that the approximation error remains controlled as detailed for the POD-DEIM method in [28, 29]. If the nonlinear part of the Hamiltonian system $\boldsymbol{h}_N$ is small and not strongly nonlinear, we can expect to find an appropriate value of $m \ll N$ that introduces an error smaller than, or comparable to, that imposed by the PSD. Thus, the matrices $\bar{B}$ and W are computed once offline. As said above, the linear term has a cost $\mathcal{O}(K)$. Then, the nonlinear term has a cost $\mathcal{O}(mK + \alpha(m))$: it does not depend on N anymore.

Remark. *In practice, the primary challenge in achieving an efficient implementation of SDEIM lies in ensuring that the evaluation of $\boldsymbol{h}_m = P^T \circ \boldsymbol{h}_N \circ A$ remains independent of the full-order dimension N. A straightforward implementation would still involve evaluating $\boldsymbol{h}_N(\mathbf{u})$, which defeats the purpose of hyper-reduction. To address this, it is essential to reconstruct only the necessary rows of the state vector $\bar{\mathbf{u}}$, allowing $\boldsymbol{h}_N$ to be evaluated at a reduced set of points. This approach ensures that the multiplication by P^T becomes implicit or effectively eliminated. This method is intrusive and system-dependent.*

We finish this section with the construction of the selection matrix P as detailed in [27]. The interpolation points $\{i_1, \dots, i_m\}$ are selected with a greedy algorithm offline. Essentially, we select the index that maximizes the approximation error between the SVD basis and its approximation at the interpolation points at each iteration of the algorithm.

Algorithm 1 Symplectic Discrete Empirical Interpolation Method (SDEIM)

Require: Left singular vectors $A_{\boldsymbol{\psi}} = [\mathbf{a}_{\boldsymbol{\psi},1}, \dots, \mathbf{a}_{\boldsymbol{\psi},m}] \in \mathcal{M}_{2N,m}(\mathbb{R})$ from PSD of $\boldsymbol{\psi}_N$
Ensure: Interpolation indices $\{i_1, \dots, i_m\}$, selection matrix $P \in \mathcal{M}_{2N,m}(\mathbb{R})$
1: Initialize $i_1 = \arg\max |\mathbf{a}_{\boldsymbol{\psi},1}|$
2: Set $P = [\boldsymbol{e}_{i_1}]$, $U = [\mathbf{a}_{\boldsymbol{\psi},1}]$
3: **for** $k = 2$ to m **do**
4: Solve $(P^T U)\mathbf{c} = P^T \mathbf{a}_{\boldsymbol{\psi},k}$ for $\mathbf{c}$
5: Compute residual $\mathbf{r} = \mathbf{a}_{\boldsymbol{\psi},k} - U\mathbf{c}$
6: $i_k = \arg\max |\mathbf{r}|$
7: $P \leftarrow [P\ \boldsymbol{e}_{i_k}]$, $U \leftarrow [U\ \mathbf{a}_{\boldsymbol{\psi},k}]$
8: **end for**
9: **return** $\{i_1, \dots, i_m\}$, P

Remark. *The SDEIM hyper-reduction is not Hamiltonian per se. However, when the DEIM technique offers a good approximation of the reduced vector field, we can expect the reduced dynamics to remain close to Hamiltonian ones.*

2.3.4 A practical example: the shallow-water system

We present a practical example of PSD model order reduction with SDEIM hyper-reduction on the 2D shallow-water system defined in Eq. (1.2) as

$$\begin{cases} \partial_t \chi(\mathbf{x}, t; \mu) + \nabla \cdot ((1 + \chi(\mathbf{x}, t; \mu))\, \nabla \phi(\mathbf{x}, t; \mu)) = 0, \\ \partial_t \phi(\mathbf{x}, t; \mu) + \dfrac{1}{2} |\nabla \phi(\mathbf{x}, t; \mu)|^2 + \chi(\mathbf{x}, t; \mu) = 0, \end{cases} \tag{1.2}$$

where $\chi, \phi : [-4, 4]^2 \times [0, T] \times \Gamma \to \mathbb{R}$ are the perturbation from the equilibrium and the scalar velocity potential, respectively, on a periodic domain. Its Hamiltonian functional writes:

$$\mathscr{H}[\chi, \phi] = \frac{1}{2} \int_{[-4,4]^2} \left((1 + \chi) |\nabla \phi|^2 + \chi^2 \right) d\mathbf{x}. \tag{2.8}$$

We consider a uniform mesh over the square $[-4,4]^2$ resulting in semi-discretized coordinates $\boldsymbol{\chi}(t;\mu), \boldsymbol{\phi}(t;\mu) \in \mathbb{R}^N$ (in bold symbols) along with second-order accurate central finite difference operators D_x and D_y to discretize the spatial derivatives in the x and y direction, respectively. It leads to a canonical Hamiltonian ODE and the gradient of $\mathscr{H}(\mathbf{u})$ with respect to $\mathbf{u} = (\boldsymbol{\chi}, \boldsymbol{\phi})^T$ reads

$$\nabla_{\mathbf{u}}\mathscr{H}(\mathbf{u}) = \begin{pmatrix} \frac{1}{2}\left[(D_x\boldsymbol{\phi})^2 + (D_y\boldsymbol{\phi})^2\right] + \boldsymbol{\chi} \\ -D_x\left([1+\boldsymbol{\chi}] \odot D_x\boldsymbol{\phi}\right) - D_y\left([1+\boldsymbol{\chi}] \odot D_y\boldsymbol{\phi}\right), \end{pmatrix}$$

where $\odot$ is the Hadamard or element-wise product between two vectors.

PSD reduced model We discretize the space domain with $N = 64^2$ cells and the time domain with $T = 15; \Delta t = 1\times10^{-3}$ and an implicit midpoint scheme, and we consider a parametric initial condition with two parameters $\mu = (\alpha, \beta) \in [0.2, 0.5] \times [1, 1.7]$, given by

$$\chi_{\text{init}}(\mathbf{x};\mu) = \alpha \exp\left(-\beta\, \mathbf{x}^T\mathbf{x}\right), \quad \phi_{\text{init}}(\mathbf{x};\mu) = 0.$$

We sample 20 different couples of parameter (α, β) regularly spaced in the segment $[(0.2, 1), (0.5, 1.7)]$. Then, we build the snapshot matrix and compute its SVD.

We recall that the SVD computes a decomposition in the form of singular vectors and their singular values. If we view the snapshot collection as points in a high-dimensional space, each singular value measure how much of the total variability in the data is explained by the corresponding singular vectors.

Consequently, the reduced basis is constructed by selecting the most significant singular vectors first. Increasing the number of selected singular vectors, i.e. increasing K, leads to a higher explained variance over the dataset, thereby yielding a reduced basis of improved quality.

In Fig. 2.7, we see the relative magnitude of the singular values as a function of their index. To begin with, we can verify that the singular values are indeed positive and sorted in descending order. Thus, we remark a fast decay of the magnitude, it means that a small number of singular vectors are sufficient to explain most of the variability in the data. It is encouraging to build a reduced model based on the PSD : a relatively small reduced dimension K might be enough to capture the most important dynamics.

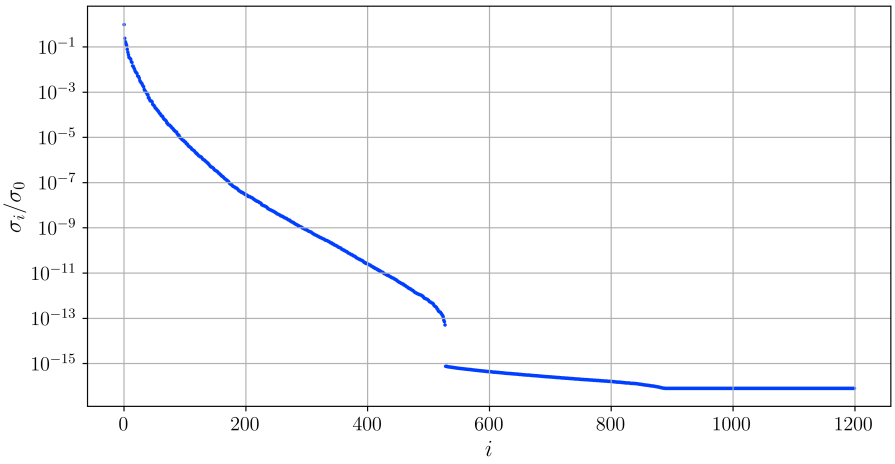


Figure 2.7: (Shallow water 2D) Relative magnitude of the first 1200 singular values of the snapshot matrix SVD as a function of their index. σ_i is the i-th singular value.

To illustrate the importance of the reduced dimension K, we set $\alpha = 0.35, \beta = 1.35$ and we compute the L^2 error as a function of time for several values of K. As discussed above, a smaller error is expected for larger values of K, which is indeed observed in Fig. 2.8.

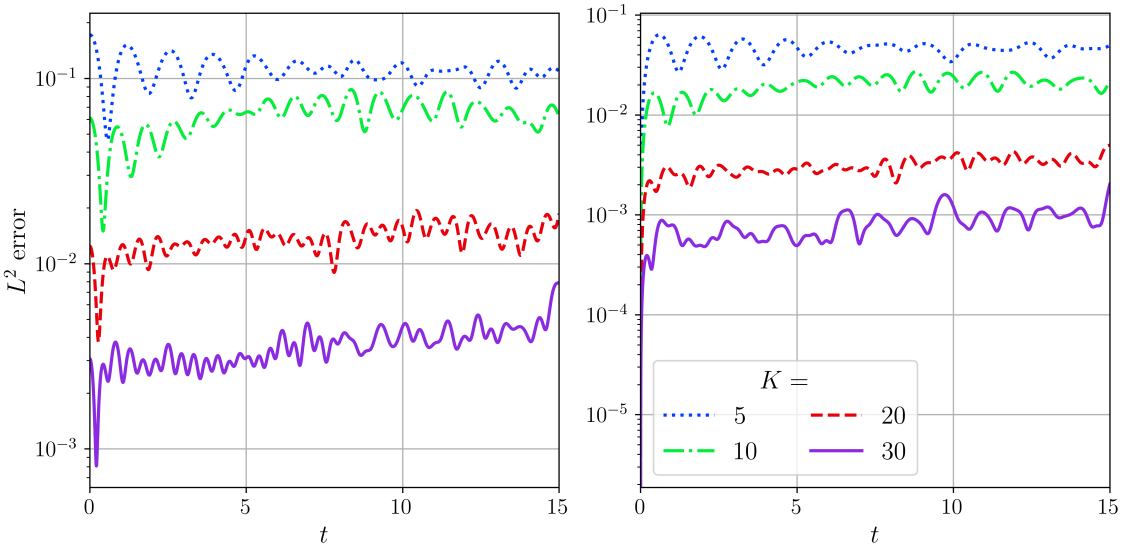


Figure 2.8: (Shallow water 2D) L^2 error as a function of time between the full solution and the reduced solution, respectively $\boldsymbol{\chi}(t;\mu)$ with $A\bar{\boldsymbol{\chi}}(t;\mu)$ (left), and $\boldsymbol{\phi}(t;\mu)$ with $A\bar{\boldsymbol{\phi}}(t;\mu)$ (right), for $\mu = (0.35, 1.35)$ and $K \in \{5, 10, 20, 30\}$.

In practice, we set $K = 20$ during the offline phase, based on the acceptable level of approximation error. In certain cases, a priori error estimates or error bounds are available to assist in the decision-making process: for instance, errors bounds are derived for randomized complex SVD basis in [30], a Hamiltonian dynamical low-rank reduced model is built in [31] using a priori error bounds.

With a given value of K, we can precompute the matrix A as defined in Prop. 2.3.2 with a cotangent lift algorithm from Prop. 2.3.5. The online stage is limited to the computation of the reduced Hamiltonian gradient $A^T \circ \nabla\mathscr{H} \circ A$ at each time-step of the simulation.

SDEIM hyper-reduction In fact, the PSD reduced model is slower than the full model. Arguing Sec. 2.3.3, we anticipated a poor performance in terms of computation time, so we need to implement a hyper-reduction technique to improve it. First, we split the full order Hamiltonian vector field as in Prop. 2.3.8. It leads to

$$\begin{aligned}\frac{d}{dt}\mathbf{u} &= B\mathbf{u} + \mathcal{J}_{2N}\boldsymbol{h}_N(\mathbf{u}) \\ &= \begin{pmatrix} 0 & -D_x^2 - D_y^2 \\ -I & 0 \end{pmatrix}\mathbf{u} + J_{2N}\begin{pmatrix} \frac{1}{2}\left[(D_x\boldsymbol{\phi})^2 + (D_y\boldsymbol{\phi})^2\right] \\ -D_x\left(\boldsymbol{\chi}\odot D_x\boldsymbol{\phi}\right) - D_y\left(\boldsymbol{\chi}\odot D_x\boldsymbol{\phi}\right)\end{pmatrix},\end{aligned}$$

and we can identify B and $\boldsymbol{h}_N$. Next, we apply Prop. 2.3.8 and identify the quantities that can be precomputed to reduce the online computational cost. The reduced system's linear part $\bar{B} = A^+BA$ is computed once offline. We then apply the SDEIM method by assembling an additional snapshot matrix composed of the evaluations of $\boldsymbol{h}_N(\mathbf{u})$, computing its SVD, and selecting the m most dominant singular vectors to form the columns of A_ψ. The selection matrix P is then constructed using the DEIM algorithm described in Sec. 2.3.3. Thus, the matrix $W = A_\psi\left(P^TA_\psi\right)^{-1}$ is precomputed.

The main difficulty arises in the treatment of the nonlinear function $\boldsymbol{h}_m := P^T \circ \boldsymbol{h}_N \circ A$. Indeed, if implemented naively, the evaluation of $\boldsymbol{h}_N$ would still incur a high computational cost, thereby nullifying the efficiency gains expected from the SDEIM. This is why the method

is considered intrusive: it requires a detailed examination of $\boldsymbol{h}_m$ to manually identify which computations can be performed in advance and whether this leads to a reduction in the online computational cost.

To simplify this process explanation, let us assume that the system is one-dimensional and that the following term $\mathbf{y}$ appears.

$$\mathbf{y} := P^T D_x A\, \bar{\mathbf{u}}$$

In other words, we must compute a (numerical) derivative of the reconstructed state at the indices $\{i_1, \dots, i_m\}$ selected by the DEIM algorithm, that is, the indices associated with the matrix P. D_x is a centered finite difference so to compute the derivative on i_j, we only need to reconstruct the state of indices $i_j + 1$ and $i_j - 1$, i.e.

$$\begin{aligned} y_j &= \frac{1}{h}\left(\hat{u}_{i_j+1} - \hat{u}_{i_j-1}\right) \\ &= \frac{1}{h}\left(A_{i_j+1,:} \cdot \bar{\mathbf{u}} - A_{i_j-1,:} \cdot \bar{\mathbf{u}}\right) \\ &= \frac{1}{h}\left(\sum_k A_{i_j+1,k}\bar{u}_k - \sum_k A_{i_j-1,k}\bar{u}_k\right), \end{aligned}$$

where $A_{i,:}$ is the i-th line of A and $\cdot$ the scalar product. As a consequence, we only need to decompress at most $2m$ reduced coefficients (with m corresponding to left and right neighbors, respectively); the computation cost therefore depends on m rather then on N. Alternatively, we can precompute $P^T D_x A \in \mathcal{M}_{m,2K}(\mathbb{R})$ offline. Depending on the programming language and other technical considerations, we may prefer either to select indices or to precompute the matrix.

In practice, we must be very careful about (i) considering $\boldsymbol{\psi}$ and $\boldsymbol{\phi}$ separately, (ii) noting that in multiple dimensions the neighboring nodes are not necessarily adjacent in the array, (iii) distinguishing interpolation indices belonging to $\boldsymbol{\psi}$ from those belonging to $\boldsymbol{\phi}$, (iv) accounting for the structure of A, which in this case is block-diagonal, and (v) the fact that $\boldsymbol{h}_N$ is often more complex than a simple derivative. Note also that, sometimes, excessive precomputation can actually harm the final computational cost. For instance, working directly with the Hadamard product would lead to a third-order tensor, which increases the numerical complexity.

Subsequently, we test the SDEIM on our test case. We note that this hyper-reduction introduces additional errors compared to the PSD reduced model. In Fig. 2.9, we observe the L^2 error of the PSD-SDEIM solution compared to the full order solution. We notice that with $m \approx 50$ points, the SDEIM-PSD error (dashed red line) is comparable to the sole PSD error in a (black full line).

In addition, in Fig. 2.10 we see the relative error on the energy $\left|\mathscr{H}(\cdot) - \mathscr{H}_{\text{init}}\right| / \mathscr{H}_{\text{init}}$ with $\mathscr{H}_{\text{init}} = \mathscr{H}(\mathbf{u}_{\text{init}})$. As explained in Sec. 2.2.3, the true Hamiltonian $\mathscr{H}$ is preserved up to the precision of the numerical scheme. We note that the hyper-reduced solution does not preserve the Hamiltonian as accurately, there are small variations. This is due to the fact that the SDEIM method is, strictly speaking, not Hamiltonian-preserving, as mentioned in Sec. 2.3.3.

Finally, we plot solutions at time $t = 8$ for the full model and both reduced model in Fig. 2.11. A very small difference is visible in the corners; however, the physical dynamics is well reproduced.

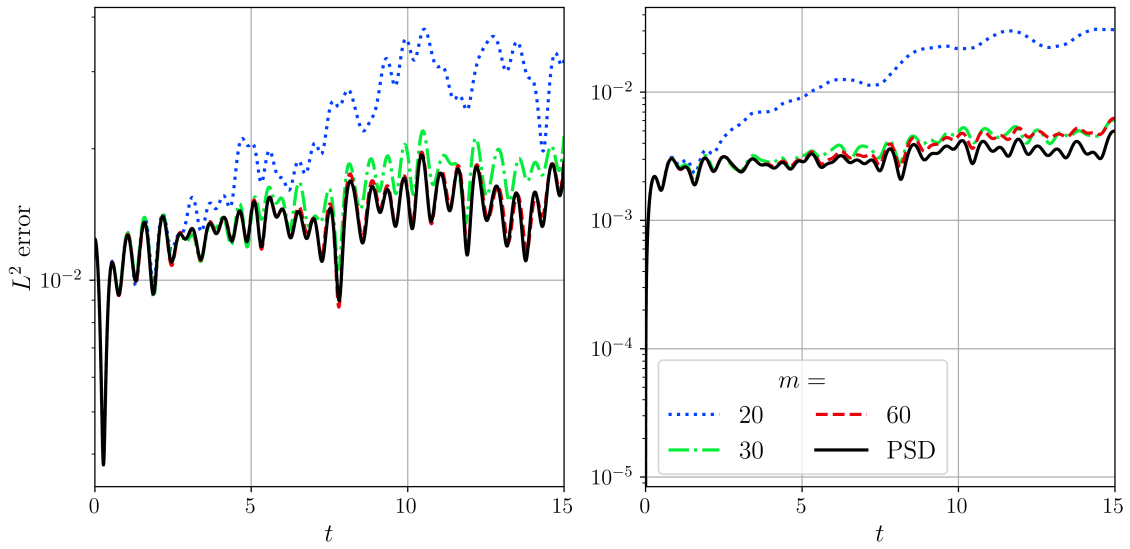


Figure 2.9: (Shallow water 2D) L^2 error as a function of time between the full solution $\mathbf{u}(t;\mu)$ and the hyper-reduced PSD-SDEIM solution $A\mathbf{u}(t;\mu)$, on the $\boldsymbol{\chi}$ variable (left) and on the $\boldsymbol{\phi}$ variable (right), for $\mu = (0.35, 1.35)$, $K = 20$ and $m \in \{20, 30, 60\}$.

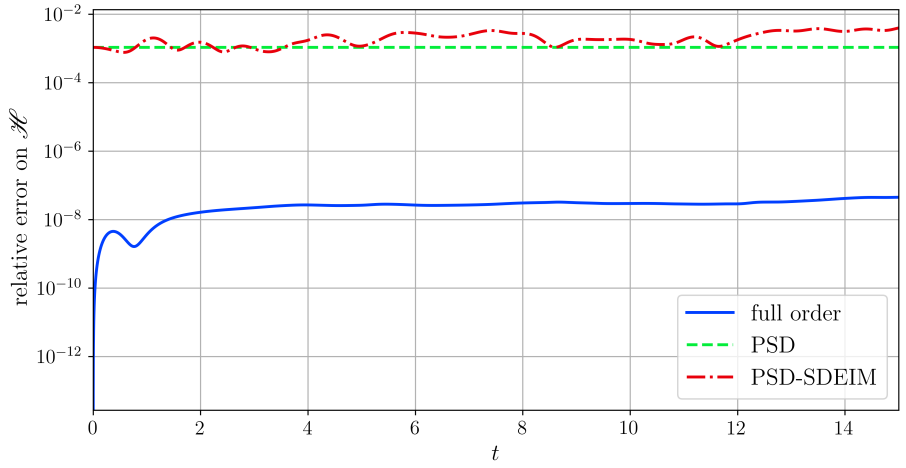


Figure 2.10: (Shallow water 2D) Relative error on $\mathscr{H}$ as a function of time between the full solution $\mathbf{u}(t;\mu)$, the PSD reduced solution and the PSD-SDEIM reduced solution for $\mu = (0.35, 1.35)$, $K = 20$ and $m = 60$.

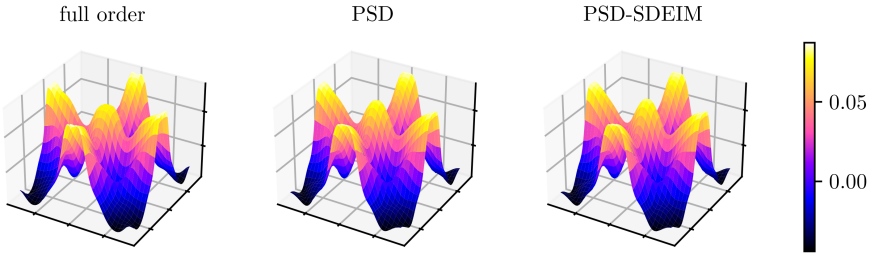


Figure 2.11: (Shallow water 2D) Full order solution $\chi(t;\mu)$ (left), PSD solution (center) and PSD-SDEIM solution (right) at $t = 8$ for $\mu = (0.35, 1.35)$, $K = 20$ and $m = 60$.

2.3.5 Other reduction methods

We have previously focused on POD-type reduction methods. Beyond this class of methods, there are many other model reduction techniques. In this section, we present a brief overview. While not all of these approaches will be employed in the remainder of this manuscript, their presentation serves to contextualize the techniques considered herein. As explained in [32], model order reduction techniques, structure-preserving techniques included, can be expressed as

$$u(x,t;\mu) \approx \sum_{i=1}^{2K} a_i(x) s_i(t;\mu) \tag{2.20}$$

where $a_i \in V$ with V a Hilbert space and $s_i \in S$ where S is a space of $\mathcal{T} \times \Gamma \to \mathbb{R}$ functions and $K \ll N$ is a small number in comparison to the FOM dimension. Model order reduction techniques can be broadly classified into three categories. The first category constructs a reduced basis $\{a_i\}_i$ in the space V. The second category, which can be seen as somewhat dual to the first, constructs a reduced basis $\{s_i(t;\mu)\}_i$ in the space S. The third category directly formulates a reduced model in the form of Eq. (2.20). These three approaches are closely interrelated and exhibit structural similarities as they all ultimately yield a low-rank approximation $\bar{u}(t;\mu)$ on the tensor space $V \otimes S$. Subsequently, we provide a more detailed discussion of each of the aforementioned categories, illustrating them with some examples in both standard and structure-preserving settings.

1. Building a reduced basis $\{a_i\}_i$ in the space V can be referred to as projection-based model order reduction. This is the case for the PSD method, which is described in Sec. 2.3.1 and the basis vectors are the columns of the matrix A as defined in Prop. 2.3.2 as the SVD of a snapshot matrix. For standard ROM, we can cite the reduced basis method [33, 34], the POD [24] or the Proper Generalized Decomposition (PGD) [35]. In a Hamiltonian preserving setting, the reduced basis method can be extended [31, 36], along with the PSD and so on.

2. Constructing a reduced basis $s_i(t;\mu)_i$ in the space S includes sparse approximation methods with, for example, the sparse grid method [37].

3. Approximations focusing directly on an expression of the form of Eq. (2.20) are called low-rank approximation methods: the solution of the system is approached by a low-rank tensor decomposition with a rank as small as possible, then reduced solutions are computed with structure-preserving methods as in [38, 39, 40].

Naturally, model reduction techniques are not restricted to Eq. (2.20). More generally, the objective is to construct low-rank, reduced-size approximations, or even other approximations that significantly decrease computational complexity compared to the full order model. For example, we can employ a time-adaptive reduced basis approach, as proposed in [41]. In this framework, the reduced basis $\{a_i(t)\}_i$ in V is dynamical, leading to a reduced model that consists of two coupled evolution equations. Another, more distant example of non structure-preserving reduction comes from [42]. They introduce a micro-macro decomposition of the Vlasov–Poisson dynamics to separate fast and slow-scale behaviors, allowing for a reduction in the computational burden by avoiding very small time steps in part of the numerical resolution, thereby speeding up the overall computation.

2.4 Deep learning tools for Hamiltonian model order reduction

This section provides a brief overview of deep learning tools that we will use to construct nonlinear model order reduction for Hamiltonian ODEs. In practice, we rely on several neural network architectures to build nonlinear operators for reduction. First, we give a definition on neural networks and we explain their fitting process called training. Then, we describe the two primary neural networks utilized in this manuscript: the Hamiltonian neural network and the convolutional autoencoder.

2.4.1 Outline of neural networks

Neural networks as parametric functions An artificial Neural Network (NN) can be seen as a parametric function g_θ of parameters $\theta \in \Theta$ with $\Theta \subset \mathbb{R}^{p_\Theta}$ a parameter space with $p_\Theta > 0$ the number, eventually large, of parameters. This function is used to identify a particular relationship between data in a set $\mathcal{Y}$ and data in set $\mathcal{Z}$, without knowing an prior analytical expression for it.

$$\begin{aligned} g_\theta : \mathcal{Y} &\longrightarrow \mathcal{Z}, \\ y &\longmapsto g_\theta(y) \approx z. \end{aligned}$$

$y \in \mathcal{Y}$ and $z \in \mathcal{Z}$ can come in a very wide variety of forms : numbers, vectors, matrix, images, time series, meshes, words, etc. A notable historical example comes from the MNIST database [43], which consists of tens of thousands of handwritten digits. Since its creation in 1994, researchers have sought to establish a mapping g_θ between each picture of an handwritten digit y and its corresponding numerical value z with the goal of automated recognition of handwritten digits.

However, the development of NN is much more recent. In the 1960's, F. Rosenblatt introduced the perceptron [44], one the first actually implemented NN. These concepts, along with others that we will not discuss, have evolved over the years and resulted in a modern definition of multiplayer perceptron or feedforward NN, which we will adopt here. Though not entirely accurate, we will use the term NN to describe multiplayer perceptron. In the following, $\mathcal{Y} = \mathbb{R}^{n_{\text{in}}}$ and $\mathcal{Z} = \mathbb{R}^{n_{\text{out}}}$.

Definition 2.4.1 (Multiplayer perceptron)**.** *A multiplayer perceptron, or NN, is a parametric function $g_\theta : \mathbb{R}^{n_{in}} \to \mathbb{R}^{n_{out}}$ of parameters $\theta \in \Theta$, $n_{in}, n_{out} > 0$ constituted by the composition of $l > 0$ elementary functions $g^{[k]} : \mathbb{R}^{n^{[k-1]}} \to \mathbb{R}^{n^{[k]}}$ called layers such that*

$$g_\theta = g^{[l]} \circ g^{[l-1]} \circ \cdots \circ g^{[1]}.$$

$n^{[k]} > 0, i \in \{1, \dots, l\}$ is the number of units or output dimension of the i-th layer, $n^{[0]} = n_{in}$ and $n^{[l]} = n_{out}$. A layer $g^{[k]} : \mathbb{R}^{n^{[k-1]}} \to \mathbb{R}^{n^{[k]}}$ is a simple function of the form

$$g^{[k]}(\mathbf{y}) = \sigma\left(W^{[k]}\mathbf{y} + \mathbf{b}^{[k]}\right) \tag{2.21}$$

with a weight matrix $W^{[k]} \in \mathcal{M}_{n^{[k]},n^{[k-1]}}(\mathbb{R})$, a bias $\mathbf{b}^{[k]} \in \mathbb{R}^{n^{[k]}}$ and $\sigma : \mathbb{R} \to \mathbb{R}$ a nonlinear function. We denote $\sigma(\mathbf{y}) := (\sigma(y_i))_i$.

The NN parameters θ is the set of every layer's weight matrix and bias i.e.

$$\theta \in \Theta := \left\{ W^{[k]}, b^{[k]} \middle| W^{[k]} \in \mathcal{M}_{n^{[k]},n^{[k-1]}}(\mathbb{R}), \mathbf{b}^{[k]} \in \mathbb{R}^{n^{[k]}}, k \in \{1, \dots, l\} \right\}$$

In general, the activation function σ is fixed for every layer with the exception of the last one.

There are many options available for the activation function σ, for instance the Rectified Linear Unit (ReLU) $\mathbf{y} \to \max(0, \mathbf{y})$, the hyperbolic tangent $\mathbf{y} \to \tanh \mathbf{y}$ or the softplus $\mathbf{y} \to \log(1 + \exp x)$.

In the literature, the term "hidden layer" refers to any layer that is neither the input layer nor the output layer. Then, layers $g^{[k]}$ defined in Eq. (2.21) are named dense or fully-connected and a MLP with these layers is called "dense neural network".

In some cases, a part or all an MLP layers can be replaced with convolutional layers. Essentially, the weight matrix multiplication is exchanged with a discrete convolution with a small kernel as detailed in Def. 2.4.2. MLPs with convolution layers are called "convolutional neural networks".

Definition 2.4.2 (Convolutional layer)**.** *A convolutional layer $g^{[k]} : \mathbb{R}^{n^{[k-1]}} \to \mathbb{R}^{n^{[k]}}$ is a function of the form*

$$g^{[k]}(\mathbf{y}) = \sigma\left(K^{[k]} * \mathbf{y} + \mathbf{b}^{[k]}\right)$$

with a kernel $K^{[k]}$ and a bias $\mathbf{b}^{[k]}$. We denote $*$ the discrete 1D concolution operator

$$(K^{[k]} * \mathbf{y})_i = \sum_j y_{i+j} K_j^{[k]}$$

The kernel replaces the weight matrix in the definition of parameters. The remainder is unchanged.

Remark. *In practice, the inputs to convolutional layers consist of multiple channels i.e. columns. As a result, convolutional layers perform several convolutions in parallel, one for each channel of the input signal.*

Many other types of layers and neural networks more complex than a simple composition of functions. They will not be discussed in this document. More details are available in [45, 46].

Training a neural network The next step is to find appropriate parameters $\theta \in \Theta$ such that g_θ approximate a target function g^*. We assume that we know the value of g^* on a given set of points $(\mathbf{y}_i)_i$. Let us denote $\mathbf{z}_i = g^*(\mathbf{y}_i)$, and we have a set of input-output data $\mathscr{D}$

$$\mathscr{D} := \{(\mathbf{y}_i, \mathbf{z}_i)_i\}$$

Then, we define the loss function, $\mathcal{L} : \Theta \to \mathbb{R}$, also called cost function, as the error between g_θ and g^* on the dataset $\mathscr{D}$

$$\mathcal{L}(\theta) = \sum_{(\mathbf{y}_i, \mathbf{z}_i) \in \mathscr{D}} \|\mathbf{z}_i - g_\theta(\mathbf{y}_i)\|_2 = \sum_{(\mathbf{y}_i, \mathbf{z}_i) \in \mathscr{D}} \|g^*(\mathbf{y}_i) - g_\theta(\mathbf{y}_i)\|_2.$$

An appropriate θ such that $g_\theta \approx g^*$ is a local minimizer of the loss function, i.e. a local solution of the optimization problem

$$\theta \in \operatorname{argmin}_{\theta \in \Theta} \mathcal{L}(\theta).$$

Searching for a minimizer θ is called the training process. The simplest method is the so-called gradient descent method: we define a sequence of parameters $(\theta^k)_k$ as follows

$$\theta^{k+1} = \theta^k - \eta \nabla_\theta \mathcal{L}(\theta^k), \tag{2.22}$$

where $\eta > 0$ is the step size. The gradient of loss function is computed with the chain rule, using the backpropagation algorithm [45, 47].

This simple version of the gradient descent algorithm is rarely used as is, for two main reasons. First, the dimension of the parameter space Θ is usually very large. As a result the method

might converges poorly due to gradient averaging effects. Second, the loss function $\mathcal{L}$ is not convex a priori: we can expect numerous local minima and a wide range of gradient magnitudes. Hence, the gradient descent will also converge poorly, if at all. One solution to address these issues is to use an adaptive stochastic gradient descent with momentum. Very briefly, adaptive means that the step size η depends on the training iteration to give more robustness to the algorithm. A stochastic descent computes the gradient on a small subset of the dataset $\mathscr{D}$ randomly chosen at each iteration: it fastens computations and circumvents the averaging effects. Momentum consists in adding a fraction of the previous iteration's gradient to the current parameter update, thus reducing oscillations in the descent direction and avoiding getting stuck in a sub-optimal local minimum. More details can be found in [45, 48].

Remark. *The situation considered so far, in which the input–output relationship g^* is known, is referred to as supervised learning. This is not always the case, when such a relationship is unknown, the problem is called unsupervised learning. We will see an example in the following Sec. 2.4.3.*

A natural question that arises concerns the quality of the approximations produced by such architectures. Given a target function g^*, can a neural network g_θ be shown, in theory, to converge to it? The answer is affirmative: both dense and convolutional neural networks fall under the category of universal approximators, meaning they are theoretically capable of approximating any continuous function to arbitrary accuracy under suitable conditions [49, 50, 51].

Thereafter, we introduce two practical neural networks models which will play a key role in the construction of nonlinear reduction methods : the Hamiltonian neural network and the convolutional autoencoder.

2.4.2 Learning symplectic flows with Hamiltonian Neural Networks

For the construction of nonlinear reduction methods, we use neural networks to approximate the time evolution of Hamiltonian systems : they learn the underlying symplectic flow of an (unknown) Hamiltonian ODE directly from data. The goal of such network is to replicate/predict the system dynamics. These models incorporate symplecticity -either by design or through loss constraints-to ensure long-term stable and physically accurate predictions. Several methods has been developed, for instance the Hamiltonian Neural Network (HNN) [52] learns an Hamiltonian function, SympNets [53] are designed to learn symplectic maps and SympFlow [54] are neural network-based symplectic integrators.

In this work, we focus on the HNN [52] first introduced by S. Greydanus, M. Dzamba and J. Yosinski in 2019. The HNN $\mathscr{H}_{\theta_h}$ of parameters θ_h represents the Hamiltonian of a system

$$\begin{aligned} \mathscr{H}_{\theta_h} : \mathbb{R}^{2N} \times \Gamma &\to \mathbb{R}, \\ (\mathbf{u}; \mu) &\mapsto \mathscr{H}_{\theta_h}(\mathbf{u}; \mu). \end{aligned}$$

with a state $\mathbf{u}(t; \mu) \in \mathbb{R}^{2N}$ and some parameters $\mu \in \Gamma$. The HNN is implemented as a standard feedforward neural network with dense layers or MLP as given in Def. 2.4.1. The parameters μ are just concatenated to the first layer's input. As the output of interest is often the gradient $\nabla_{\mathbf{u}} \mathscr{H}_{\theta_h}(\mathbf{u}; \mu)$, we remove the bias and set a linear activation function $\sigma = \mathrm{id}$ in the last layer i.e. $g^{[l]}(\mathbf{y}) = W^{[l]}\mathbf{y}$. The implementation is made rather practical for gradient computation by automatic differentiation methods implemented in standard machine learning libraries.

The HNN learns a system's flow through its vector field. Hence, we enforce that the gradient of the neural network approximates the vector field of the underlying system with the prediction loss $\mathcal{L}_{\text{pred}}(\theta_h)$ in the form

$$\mathcal{L}_{\text{pred}}(\theta_h) = \sum_{\mathbf{u} \in \mathscr{D}} \left\| \dot{\mathbf{u}} - \nabla_{\mathbf{u}} \mathscr{H}_{\theta_h}(\mathbf{u}; \mu) \right\|_2$$

In most cases, the time derivative of $\mathbf{u}$ is not known, we can replace it by any numerical estimate, for example a central finite difference $(\mathbf{u}^{n+1} - \mathbf{u}^{n-1})/(2\Delta t)$. We could also replace the term inside the norm with its integrated form and thus use a numerical scheme.

Remark. *Strictly speaking, with this loss function, the HNN learns a slightly modified Hamiltonian that depends on the numerical scheme and its order of accuracy (see more details in [18, Chapter 9]). Nevertheless, this minor difference is deemed acceptable in this context.*

The definitions provided in this section fully generalize to the case of learning a reduced model of reduced state $\bar{\mathbf{u}}(t;\mu)$ and reduced learned Hamiltonian $\bar{\mathscr{H}}_{\theta_h}$, as we do in this manuscript.

2.4.3 Convolutional AutoEncoder for low-dimensional representations

Convolution AutoEncoders (CAEs) are a family of neural networks architectures designed to learn efficient, low-dimensional representations of high-dimensional data [45]. Put differently, they are built as nonlinear compression tools. CAEs are composed of two parts : an encoder which compress the data into a latent/reduced space, and a decoder which reconstructs the input from its latent form. The encoder is made of convolutional layers followed by dense layers, and conversely for the decoder. In fact, the simplest autoencoders consisted exclusively of dense layers, with dimensionality reduction achieved by progressively decreasing the number of units in each successive layer. Convolutional layers are added for two main reasons

(i) convolutional layers exploit the input's spatial structure, making them efficient for data that exhibits this type of structure such as wave propagation or fluid dynamics, and essentially mesh-discretized data $\mathbf{u}$,

(ii) the CAE is significantly more parameter-efficient, offering comparable performance with a lower number of parameters, especially with a large full dimension N.

Thus, the encoder-decoder architecture of a CAE is implemented as a pair of neural networks $\mathcal{E}_{\theta_e}, \mathcal{D}_{\theta_d}$ called an encoder and a decoder, respectively. The encoder is a mapping $\mathcal{E}_{\theta_e} : \mathbb{R}^n \to \mathbb{R}^m$ with $m \ll n$ which output reduced/compressed representations. Then, the decoder $\mathcal{D}_{\theta_d} : \mathbb{R}^m \to \mathbb{R}^n$ decompress reduced data.

These networks are trained together such that the following reconstruction loss function is enforced

$$\mathcal{L}_{\text{AE}}(\theta_e, \theta_d) = \sum_{\mathbf{y} \in \mathscr{D}} \|\mathbf{y} - \mathcal{D}_{\theta_d}\left(\mathcal{E}_{\theta_e}(\mathbf{y})\right)\|_2 \tag{2.23}$$

meaning that the training aims to achieve minimal information loss during the compression-decompression process.

In practice, the encoder and decoder are designed as mirror images of one another. As illustrated in Fig. 2.12, the encoder takes as input a high-dimensional vector $\mathbf{y} \in \mathbb{R}^n$. The encoding process starts with a sequence of encoder blocks, each consisting of a standard convolutional layer followed by a downsampling layer—typically implemented as a convolutional layer with a stride two. This stride implies that the kernel advances across the input with a step size of two rather than one, effectively halving the spatial resolution. Simultaneously, the number of feature channels is doubled; for example, a transformation from a shape $(2l, f)$ to $(l, 2f)$. After passing through multiple encoder blocks, the resulting output is fed to series of fully connected (dense) layers. These layers progressively reduce the dimensionality, ultimately yielding a latent representation $\bar{\mathbf{y}} \in \mathbb{R}^m$. The decoder is built as a mirror of the prior. Downsampling layers are replaced by upsampling layers which are transposed convolutional layers. Ultimately, the decoder outputs $\hat{\mathbf{y}} \approx \mathbf{y} \in \mathbb{R}^n$.

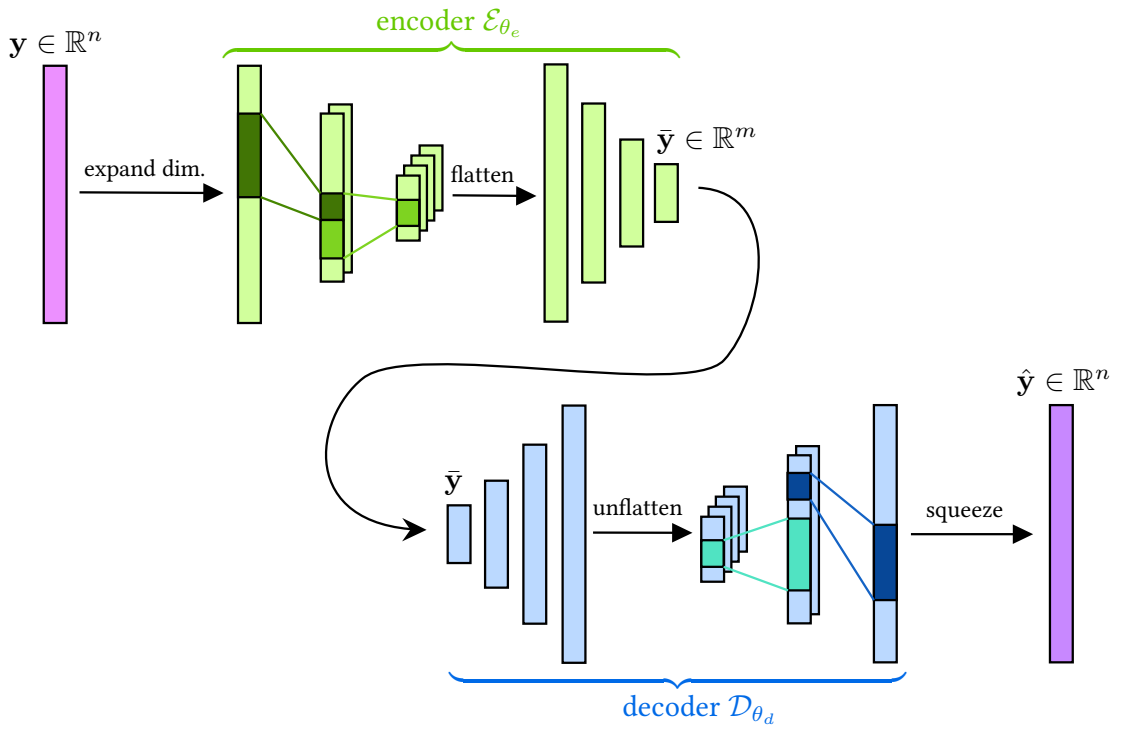


Figure 2.12: CAE architecture: from the full state $\mathbf{u}(t;\mu)$, a reduced state $\bar{\mathbf{u}}(t;\mu) = \mathcal{E}_{\theta_e}(\mathbf{u}(t;\mu))$ is computed with the decoder. Newt, the reduced state is decompress with the decoder leading to an approximation of the full state $\hat{\mathbf{u}}(t;\mu) = \mathcal{D}_{\theta_d}(\bar{\mathbf{u}}(t;\mu))$.

Downsampling and upsampling can be replaced by pooling layers. For example, a max pooling layer partitions the input into segments of a specified stride length and outputs the maximum value from each segment. Furthermore, a stride of two can be replaced by any higher value, resulting in harsher compression. Thus, multiple convolutions can be applied in each block before the sampling layer. Also, 1D convolution can easily be replaced by their 2D or 3D counterparts for specific cases.

References

[1] I. S. Newton. *Philosophiae naturalis principia mathematica*. William Dawson & Sons, Ltd., London, 1687, pp. viii+510+i. DOI: 10.5479/sil.52126.39088015628399.

[2] W. R. Hamilton. *On a General Method of Expressing the Paths of Light, and of the Planets, by the Coefficients of a Characteristic Function*. Hardy, P. D., 1833.

[3] K. R. Meyer and G. R. Hall. *Introduction to Hamiltonian dynamical systems and the N-body problem*. Vol. 90. Applied Mathematical Sciences. Springer-Verlag, New York, 1992, pp. xii+292. ISBN: 0-387-97637-X. DOI: 10.1007/978-1-4757-4073-8.

[4] V. Michel-Dansac. "Structure-preserving methods for Hamiltonian ODEs". Lecture notes for a course given the Master 2 Mathématiques Fondamentales at the University of Strasbourg. 2023.

[5] E. Franck. "Apprentissage et calcul scientifique". Lecture notes for courses given at the University of Strasbourg. Oct. 2024. URL: https://sciml.gitlabpages.inria.fr/scimllectures/meta_frontmatter.html.

[6] A. Ghosh. *Introduction to analytical mechanics*. Springer, Singapore, 2024, pp. xvii+128. ISBN: 978-981-97-2484-0. DOI: 10.1007/978-981-97-2484-0.

[7] P. J. Olver. *Applications of Lie groups to differential equations*. Second. Vol. 107. Graduate Texts in Mathematics. Springer-Verlag, New York, 1993, pp. xxviii+513. ISBN: 0-387-95000-1. DOI: 10.1007/978-1-4612-4350-2.

[8] V. I. Arnold. *Mathematical methods of classical mechanics*. Vol. 60. Graduate Texts in Mathematics. Translated from the Russian by K. Vogtmann and A. Weinstein. Springer-Verlag, New York-Heidelberg, 1978, pp. x+462. ISBN: 0-387-90314-3.

[9] J. E. Marsden and T. S. Ratiu. *Introduction to mechanics and symmetry*. Second. Vol. 17. Texts in Applied Mathematics. A basic exposition of classical mechanical systems. Springer-Verlag, New York, 1999, pp. xviii+582. ISBN: 0-387-98643-X. DOI: 10.1007/978-0-387-21792-5.

[10] D. D. Holm. *Geometric mechanics. Part I*. Dynamics and symmetry. Imperial College Press, London; distributed by World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2008, pp. xx+354. ISBN: 1-84816-195-6.

[11] F. Casas et al. "High-order Hamiltonian splitting for the Vlasov-Poisson equations". In: *Numer. Math.* 135.3 (2017), pp. 769–801. ISSN: 0029-599X,0945-3245. DOI: 10.1007/s00211-016-0816-z.

[12] P. J. Morrison. "Hamiltonian description of the ideal fluid". In: *Rev. Modern Phys.* 70.2 (1998), pp. 467–521. ISSN: 0034-6861,1539-0756. DOI: 10.1103/RevModPhys.70.467.

[13] T. J. Bridges and S. Reich. "Numerical methods for Hamiltonian PDEs". In: *J. Phys. A* 39.19 (2006), pp. 5287–5320. ISSN: 0305-4470,1751-8121. DOI: 10.1088/0305-4470/39/19/S02.

[14] A. Lew, J. Marsden, and M. West. "An Overview of Variational Integrators". In: *Finite Element Methods: 1970's and Beyond* (2004).

[15] M. Kraus et al. "GEMPIC: geometric electromagnetic particle-in-cell methods". In: *J. Plasma Phys.* 83.4 (2017). ISSN: 1469-7807. DOI: 10.1017/s002237781700040x.

[16] H. E. Cabral and L. Brandão Dias. *Normal forms and stability of Hamiltonian systems*. Vol. 218. Applied Mathematical Sciences. With a foreword by Kenneth Meyer. Springer, Cham, 2023, pp. xxi+337. ISBN: 978-3-031-33045-2. DOI: 10.1007/978-3-031-33046-9.

[17] P. Lochak. “Stability of Hamiltonian systems over exponentially long times: the near-linear case”. In: *Hamiltonian dynamical systems (Cincinnati, OH, 1992)*. Vol. 63. IMA Vol. Math. Appl. Springer, New York, 1995, pp. 221–229. ISBN: 0-387-94437-0. DOI: 10.1007/978-1-4613-8448-9_16.

[18] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration*. Second. Vol. 31. Springer Series in Computational Mathematics. Structure-preserving algorithms for ordinary differential equations. Springer-Verlag, Berlin, 2006, pp. xviii+644. ISBN: 978-3-540-30663-4.

[19] B. M. Afkham and J. S. Hesthaven. “Structure preserving model reduction of parametric Hamiltonian systems”. In: *SIAM J. Sci. Comput.* 39.6 (2017), A2616–A2644. ISSN: 1064-8275,1095-7197. DOI: 10.1137/17M1111991.

[20] J. S. Hesthaven, C. Pagliantini, and N. Ripamonti. “Adaptive symplectic model order reduction of parametric particle-based Vlasov-Poisson equation”. In: *Math. Comp.* 93.347 (2024), pp. 1153–1202. ISSN: 0025-5718,1088-6842. DOI: 10.1090/mcom/3885.

[21] C. Greif and K. Urban. “Decay of the Kolmogorov N-width for wave problems”. In: *Appl. Math. Lett.* 96 (2019), pp. 216–222. ISSN: 0893-9659,1873-5452. DOI: 10.1016/j.aml.2019.05.013.

[22] L. Peng and K. Mohseni. “Symplectic model reduction of Hamiltonian systems”. In: *SIAM J. Sci. Comput.* 38.1 (2016), A1–A27. ISSN: 1064-8275,1095-7197. DOI: 10.1137/140978922.

[23] Y. C. Liang et al. “Proper orthogonal decomposition and its applications. I. Theory”. In: *J. Sound Vibration* 252.3 (2002), pp. 527–544. ISSN: 0022-460X. DOI: 10.1006/jsvi.2001.4041.

[24] C. Gräßle, M. Hinze, and S. Volkwein. “Model order reduction by proper orthogonal decomposition”. In: *Volume 2 Snapshot-Based Methods and Algorithms*. Ed. by P. Benner et al. Model Order Reduction. De Gruyter, 2021, pp. 47–96. ISBN: 9783110671490. DOI: 10.1515/9783110671490-002.

[25] K. Lange. “Singular Value Decomposition”. In: *Numerical analysis for statisticians*. Second. Statistics and Computing. Springer, New York, 2010, pp. 129–142. ISBN: 978-1-4419-5945-4. DOI: 10.1007/978-1-4419-5945-4_9.

[26] T. Tyranowski and M. Kraus. “Symplectic model reduction methods for the Vlasov equation”. In: *Contrib. Plasma Phys.* 63.5-6 (2023), e202200046. ISSN: 0863-1042. DOI: 10.1002/ctpp.202200046.

[27] S. Chaturantabut and D. C. Sorensen. “Nonlinear model reduction via discrete empirical interpolation”. In: *SIAM J. Sci. Comput.* 32.5 (2010), pp. 2737–2764. ISSN: 1064-8275,1095-7197. DOI: 10.1137/090766498.

[28] S. Chaturantabut and D. C. Sorensen. “A state space error estimate for POD-DEIM nonlinear model reduction”. In: *SIAM J. Numer. Anal.* 50.1 (2012), pp. 46–63. ISSN: 0036-1429,1095-7170. DOI: 10.1137/110822724.

[29] D. Wirtz, D. C. Sorensen, and B. Haasdonk. “A posteriori error estimation for DEIM reduced nonlinear dynamical systems”. In: *SIAM J. Sci. Comput.* 36.2 (2014), A311–A338. ISSN: 1064-8275,1095-7197. DOI: 10.1137/120899042.

[30] R. Herkert et al. *Error Analysis of Randomized Symplectic Model Order Reduction for Hamiltonian systems*. May 2024. DOI: 10.48550/arXiv.2405.10465. arXiv: 2405.10465 [math.NA].

[31] C. Pagliantini. “Dynamical reduced basis methods for Hamiltonian systems”. In: *Numer. Math.* 148.2 (2021), pp. 409–448. ISSN: 0945-3245. DOI: 10.1007/s00211-021-01211-w.

[32] A. Nouy. "Low-rank tensor methods for model order reduction". In: *Handbook of uncertainty quantification. Vol. 1, 2, 3.* Springer, Cham, 2017, pp. 857–882. ISBN: 978-3-319-12385-1. DOI: 10.1007/978-3-319-12385-1_21.

[33] Buffa, Annalisa et al. "A priori convergence of the Greedy algorithm for the parametrized reduced basis method". In: *ESAIM: M2AN* 46.3 (2012), pp. 595–603. DOI: 10.1051/m2an/2011056.

[34] J. S. Hesthaven, C. Pagliantini, and G. Rozza. "Reduced basis methods for time-dependent problems". In: *Acta Numer.* 31 (2022), pp. 265–345. ISSN: 0962-4929,1474-0508. DOI: 10.1017/S0962492922000058.

[35] F. Chinesta, P. Ladeveze, and E. Cueto. "A Short Review on Model Order Reduction Based on Proper Generalized Decomposition". In: *Arch. Computat. Methods Eng.* 18.4 (2011), pp. 395–404. ISSN: 1886-1784. DOI: 10.1007/s11831-011-9064-7.

[36] J. S. Hesthaven and C. Pagliantini. "Structure-preserving reduced basis methods for Poisson systems". In: *Math. Comput.* 90.330 (2021), pp. 1701–1740. DOI: 10.1090/mcom/3618.

[37] K. Kormann and E. Sonnendrücker. "Sparse grids for the Vlasov–Poisson equation". In: *Sparse Grids and Applications, 2014.* Ed. by Garcke, Jochen and Pflüger, Dirk. Lecture Notes in Computational Science and Engineering. Springer International Publishing, 2016, pp. 163–190. ISBN: 978-3-319-28262-6. DOI: 10.1007/978-3-319-28262-6_7.

[38] V. Ehrlacher and D. Lombardi. "A dynamical adaptive tensor method for the Vlasov-Poisson system". In: *J. Comput. Phys.* 339 (2017), pp. 285–306. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2017.03.015.

[39] L. Einkemmer and I. Joseph. "A mass, momentum, and energy conservative dynamical low-rank scheme for the Vlasov equation". In: *J. Comput. Phys.* 443 (2021), Paper No. 110495, 16. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2021.110495.

[40] L. Einkemmer and C. Lubich. "A low-rank projector-splitting integrator for the Vlasov-Poisson equation". In: *SIAM J. Sci. Comput.* 40.5 (2018), B1330–B1360. ISSN: 1064-8275,1095-7197. DOI: 10.1137/18M116383X.

[41] J. S. Hesthaven, C. Pagliantini, and N. Ripamonti. "Rank-adaptive structure-preserving model order reduction of Hamiltonian systems". In: *ESAIM: M2AN* 56.2 (2022), pp. 617–650. DOI: 10.1051/m2an/2022013.

[42] N. Crouseilles et al. "Two-Scale Macro–Micro Decomposition of the Vlasov Equation with a Strong Magnetic Field". In: *Math. Models Methods Appl. Sci.* 23.8 (2013), pp. 1527–1559. DOI: 10.1142/S0218202513500152.

[43] Y. LeCun, C. Cortes, and C.-J. Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[44] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychol. Rev.* 65.6 (1958), pp. 386–408. ISSN: 0033-295X. DOI: 10.1037/h0042519.

[45] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[46] A. Apicella et al. "A survey on modern trainable activation functions". In: *Neural Netw.* 138 (2021), pp. 14–32. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2021.01.026.

[47] R. Rojas. "The Backpropagation Algorithm". In: *Neural Networks: A Systematic Introduction*. Springer Berlin Heidelberg, 1996, pp. 149–182. ISBN: 978-3-642-61068-4. DOI: 10.1007/978-3-642-61068-4_7.

[48] C. C. Aggarwal. "Training Deep Neural Networks". In: *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing, 2018, pp. 105–167. ISBN: 978-3-319-94463-0. DOI: 10.1007/978-3-319-94463-0_3.

[49] M. Leshno et al. "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function". In: *Neural Netw.* 6.6 (1993), pp. 861–867. ISSN: 0893-6080. DOI: 10.1016/s0893-6080(05)80131-5.

[50] A. Pinkus. "Approximation theory of the MLP model in neural networks". In: *Acta numerica, 1999*. Vol. 8. Acta Numer. Cambridge Univ. Press, Cambridge, 1999, pp. 143–195. ISBN: 0-521-77088-2. DOI: 10.1017/S0962492900002919.

[51] D.-X. Zhou. "Universality of deep convolutional neural networks". In: *Appl. Comput. Harmon. Anal.* 48.2 (2020), pp. 787–794. ISSN: 1063-5203,1096-603X. DOI: 10.1016/j.acha.2019.06.004.

[52] S. Greydanus, M. Dzamba, and J. Yosinski. "Hamiltonian neural networks". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates Inc., 2019. DOI: 10.48550/arXiv.1906.01563.

[53] P. Jin et al. "SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems". In: *Neural Netw.* 132 (2020), pp. 166–179. DOI: 10.1016/j.neunet.2020.08.017.

[54] P. Canizares et al. *Hamiltonian Matching for Symplectic Neural Integrators*. 2024. DOI: 10.48550/arXiv.2410.18262. arXiv: 2410.18262 [cs.LG].

Chapter 3

Hamiltonian reduction using a convolutional autoencoder coupled to an Hamiltonian neural network

This chapter has been published as an article in *Communications in Computational Physics*.

R. Côte, E. Franck, L. Navoret, G. Steimer, and V. Vigon. "Hamiltonian reduction using a convolutional auto-encoder coupled to a Hamiltonian neural network". In: *Commun. Comput. Phys.* 37.2 (2025), pp. 315–352. ISSN: 1815-2406,1991-7120. DOI: 10.4208/cicp.OA-2023-0300

Abstract

The reduction of Hamiltonian systems aims to build smaller reduced models, valid over a certain range of time and parameters, in order to reduce computing time. By maintaining the Hamiltonian structure in the reduced model, certain long-term stability properties can be preserved. In this paper, we propose a nonlinear reduction method for models coming from the spatial discretization of partial differential equations: it is based on convolutional autoencoders and Hamiltonian neural networks. Their training is coupled in order to simultaneously learn the encoder-decoder operators and the reduced dynamics. Several test cases on nonlinear wave dynamics and fluid dynamics (shallow water) show that the method has better reduction properties than standard linear Hamiltonian reduction methods.

Chapter's contents

3.1 Introduction . . . 53
3.2 Parameterized Hamiltonian systems and reduction . . . 55
3.2.1 Parameterized Hamiltonian dynamics . . . 55
3.2.2 Hamiltonian reduced order modeling . . . 57
3.3 A nonlinear Hamiltonian reduction method . . . 58
3.3.1 Reduction with an Auto Encoder (AE) . . . 58
3.3.2 Reduced model with a Hamiltonian Neural Network (HNN) . . . 59
3.3.3 Strong coupling of the neural networks . . . 61
3.3.4 Training hyper-parameters . . . 61
3.3.5 Numerical complexity . . . 63
3.4 Numerical results . . . 64
3.4.1 Wave equations . . . 64
3.4.2 1D shallow water system . . . 72
3.4.3 2D shallow water system . . . 77
3.5 Conclusion . . . 83
References . . . 84

3.1 Introduction

Hamiltonian reduced order modeling techniques have been successfully developed in order to perform accelerated numerical simulations of some parameterized Hamiltonian models of large dimension [2, 3, 4]. The spatial discretization of some wave-like partial differential equations gives rise to very large such Hamiltonian systems. Reduced order models can be essential for real-time simulations or when a large number of simulation instances are required as part of a control, optimization or uncertainty quantification algorithm. Starting from the initial model, a large differential system, the methods consist into constructing a differential system of a smaller size that can produce valid approximate solutions for a predefined range of times and parameters. Many physical models have a Hamiltonian structure and this gives the system a certain number of geometrical properties like the conservation of energy and the symplecticity of the phase space flows. In particular, the preservation of this structure at the discrete level enables to ensure large-time stability of the numerical simulations [5]. In order to build consistent and robust reduced models, it is therefore interesting to preserve this Hamiltonian structure through the reduction.

The construction of reduced models can be divided in two steps: (i) find a so-called pair of encoder and decoder operators that goes from the full to the reduced variables and inversely; (ii) identify the dynamics followed by the reduced variables. The construction of the encoder and decoder operators relies on a large number of data produced by numerical simulations in the range of time and parameters of interest.

The first approach to reduce a large Hamiltonian system relies on a linear approximation: the solutions manifold is approximated with a symplectic vector space of small dimension [2]. The encoder is here a linear mapping, which is also constructed to be symplectic so that the Hamiltonian structure is preserved into the reduced model. Such symplectic mapping can be constructed from data through greedy algorithms [6] or through a Singular Value Decomposition (SVD) methodology: this is the Proper Symplectic Decomposition (PSD) proposed in [2]. In this work, several algorithms have been proposed to define approximated optimal symplectic mappings: for instance, the cotangent-lift algorithm devise a symplectic mapping which is also

orthogonal and have a block diagonal structure. Then the reduced model is obtained using the Galerkin projection method: the model is constructed by supposing that a symplectic projection of the residual vanishes, where the residual stands for the error obtained after replacing the original variables by the decoded reduced variables.

Such linear reductions, however, can hardly handle nonlinear dynamics: this is the case for convection-dominated or nonlinear wave-like problems for which the solution manifold is badly approximated by hyperplanes. In order to build more expressive reduced models, one possibility is to consider time adaptive reduced methods [7, 8]. Another widely investigated possibility is to consider nonlinear reduction methods.

Regarding the construction of nonlinear encoder-decoder operators, a first class of methods rely on manifold learning techniques [9, 10]. Such methods are based on the geometrical analysis of the neighbors graph of the data thanks to the computation of geodesic distances (ISOMAP method, [11]), of eigenfunctions of the graph Laplacian (EigenMaps method [12]) or of diffusion processes (DiffusionMaps method [13]). This provides reduced variables for each data that can be further interpolated using the Nyström formula [14].

Since the explosion of deep learning in the early 2010s, new dimension reduction methods grounded on neural networks have been developed. The convolutional autoencoder architecture [15] seems particularly appropriate since its very purpose is to determine latent variables: the neural network is indeed divided into an encoder part and decoder part and they are trained simultaneously so that the sequence of encoder and decoder is close to the identity map. This was originally developed for image generation, but has been also used for reduced order modeling for models coming from the spatial discretization of partial differential equations on a grid [16, 17, 18]. Indeed, convolutional neural networks have proved particularly effective to extract multiscale information of grid-structured data. Secondly, they involve far less parameters than their dense counterparts, especially when the input size of the neural network is large. In [19], the authors use autoencoder neural networks for Hamiltonian reduction: the encoder and the decoder are weakly constrained to be symplectic thanks to a penalization term in the cost functional.

Once nonlinear encoders and decoders have been devised from data, the dynamics of the reduced variables still has to be determined. Two strategies can be considered. The first one relies on a Galerkin projection of the Hamiltonian system as in the linear case [19]. Note that the reduced model is indeed Hamiltonian provided the decoder is a symplectic map. However, the reduced model still requires the evaluation of the vector field in the original large dimension space of size $2N$: this is a well-known difficulty in nonlinear reduction. To overcome this difficulty, hyper-reduction methods have been proposed like the discrete empirical interpolation method (DEIM) [20]. This method has to be adapted to not destroy the geometric structure of the full order model as in [8, 21, 22] where the authors propose to apply a DEIM algorithm to the gradient of the Hamiltonian thus, with some additional strategies, preserve the geometric structure of the full order model.

Another approach is to learn about the dynamics of the reduced variables using a neural network: given the initial state, the neural network provides the full trajectory. As the learning is done directly in the reduced dimension, the obtained reduced model does not require an evaluation of nonlinear terms in the original variables: this is a clear advantage of the method compared with projection-based ones. The reduced dynamics can be captured for instance by Recurrent Neural Networks (RNNs), Long Short Term Memory (LSTM) neural networks [23], or by fully connected networks [24]. This has also been considered as correction of the Galerkin-type reduced models [25].

Here we consider another strategy which consists in learning the vector field that generates the observed reduced dynamics. The neural network is trained so as to minimize its deviation from the finite difference time derivative of the reduced data obtained after encoding. As we aim at conserving the Hamiltonian structure at the reduced level, the vector field is further sup-

posed to be associated with a reduced Hamiltonian function. Therefore, we can learn directly the Hamiltonian function instead of the vector field. This is a so-called Hamiltonian Neural Network (HNN) strategy proposed in [26] where a symplectic time integrator is used. We also note that neural networks methods has also recently be used to learn hidden or reduced dynamics which also involve dissipation [27, 28, 29, 30].

The present paper proposes to combine an autoencoder strategy for the encoding-decoding part and a HNN method to learn the reduced dynamics: this will be referred to as the AE-HNN method. Note that there is a priori no reason for the autoencoder neural networks to spontaneously provide reduced variables compatible with Hamiltonian dynamics. Therefore, some constraints on the autoencoder have to be added. This can be done by imposing symplecticity weakly as in [19], where a penalization term of the symplectic constraint is added to the loss function. Here, we propose instead to train it simultaneously with the HNN. With this joint training, the autoencoder will gradually converge to a set of reduced variables compatible with a Hamiltonian system. Of course, this means that the loss functions associated with each neural network must be weighted judiciously during training. Note that such a joint training of the encoding-decoding operators and the reduce dynamics have been explored in [24, 27], but without considering Hamiltonian structures for the first and without symplectic time integrator for the second.

The outline of the article is as follows. In Section 3.2, we introduce parameterized Hamiltonian systems as well as the main steps for the construction of reduced order models. Section 3.3 then presents the nonlinear AE-HNN reduction method. In particular we describe the architectures and the loss functions used for the trainings. Finally, Section 3.4 is devoted to the numerical results: we apply our reduction method on the Hamiltonian systems obtained after spatial discretization of linear, nonlinear wave equations and a shallow water system and compare it with the linear PSD reduction technique.

3.2 Parameterized Hamiltonian systems and reduction

In this section, we introduce the notations used for the parameterized Hamiltonian systems and the main steps for the construction of a reduced model.

In the following, we often write vectors of interest with bold script letters, operators with capital italic letters, with their parameters as indices, and overline quantities when related to the reduced model.

3.2.1 Parameterized Hamiltonian dynamics

We consider a parameterized autonomous Hamiltonian system, whose solution, $\mathbf{y}(t;\mu) \in \mathbb{R}^{2N}$ with $N \in \mathbb{N}^*$, depends on time $t \in [0,T]$, with $T > 0$, and on a parameter $\mu \in \Xi \subset \mathbb{R}^d$, with $d \in \mathbb{N}$. The dynamics derive from a given Hamiltonian function $\mathscr{H} : \mathbb{R}^{2N} \times \Xi \to \mathbb{R}$ and writes

$$\begin{cases} \dfrac{d}{dt}\mathbf{y}(t;\mu) = J_{2N}\,\nabla_{\mathbf{y}}\mathscr{H}\left(\mathbf{y}(t;\mu);\mu\right), & \text{in } (0,T], \\ \mathbf{y}(0;\mu) = \mathbf{y}_{\text{init}}(\mu), \end{cases} \tag{3.1}$$

where $\mathbf{y}_{\text{init}}(\mu) \in \mathbb{R}^{2N}$ is a given initial condition and J_{2N} refers to the canonical symplectic matrix

$$J_{2N} = \begin{pmatrix} 0_N & I_N \\ -I_N & 0_N \end{pmatrix},$$

with I_N the identity matrix of dimension N.

Introducing the canonical coordinates $\mathbf{y} = (\mathbf{q}, \mathbf{p})^T$, the system becomes:

$$\begin{cases} \dfrac{d}{dt}\mathbf{q}(t;\mu) = \nabla_{\mathbf{p}}\mathscr{H}\left(\mathbf{q},\mathbf{p};\mu\right), & \text{in } (0,T], \\ \dfrac{d}{dt}\mathbf{p}(t;\mu) = -\nabla_{\mathbf{q}}\mathscr{H}\left(\mathbf{q},\mathbf{p};\mu\right), & \text{in } (0,T], \\ \mathbf{q}(0;\mu) = \mathbf{q}_{\text{init}}(\mu), \\ \mathbf{p}(0;\mu) = \mathbf{p}_{\text{init}}(\mu), \end{cases} \tag{3.2}$$

with $\mathbf{q}_{\text{init}}, \mathbf{p}_{\text{init}} \in \mathbb{R}^N$ such that $\mathbf{y}_{\text{init}} = (\mathbf{q}_{\text{init}}, \mathbf{p}_{\text{init}})^T$. A key property of such systems is that the associated flow is symplectic, meaning that $\phi_t\left(\mathbf{y}_{\text{init}}(\mu);\mu\right) = \mathbf{y}(t;\mu)$ satisfies the relation

$$\left(\nabla_{\mathbf{y}}\phi_t\left(\mathbf{y}_{\text{init}}(\mu);\mu\right)\right)^T J_{2N}\left(\nabla_{\mathbf{y}}\phi_t\left(\mathbf{y}_{\text{init}}(\mu);\mu\right)\right) = J_{2N}.$$

One consequence is that the Hamiltonian $\mathscr{H}$ is preserved along the flow

$$\forall t \in (0,T], \mu \in \Xi, \quad \mathscr{H}\left(\mathbf{y}(t;\mu);\mu\right) = \mathscr{H}\left(\mathbf{y}_{\text{init}}(\mu);\mu\right),$$

which is of particular importance when considering physical systems.

In this work, we are specifically interested in Hamiltonian systems resulting from the space discretization of one-dimensional and two-dimensional wave-type equations. In such systems, $\mathbf{q} \in \mathbb{R}^N$ refers to the height of the wave at grid points and $\mathbf{p} \in \mathbb{R}^N$ to the velocity of the wave also at grid points. Examples will be detailed in the numerical section.

In order to provide numerical approximations of the solution, specific numerical schemes have been developed to ensure the symplectic property at the discrete level [5]. These schemes also guarantee large time stability of the numerical solutions. Here we consider the standard second-order Störmer-Verlet scheme. Denoting $\mathbf{y}_\mu^n = \left(\mathbf{q}_\mu^n, \mathbf{p}_\mu^n\right)^T \in \mathbb{R}^{2N}$ the approximate solution at time $t^n = n\Delta t$, with time step $\Delta t > 0$, one iteration of the scheme is defined by:

$$\begin{cases} \mathbf{p}_\mu^{n+\frac{1}{2}} = \mathbf{p}_\mu^n - \frac{1}{2}\Delta t\, \nabla_{\mathbf{q}}\mathscr{H}\left(\mathbf{q}_\mu^n, \mathbf{p}_\mu^{n+\frac{1}{2}};\mu\right), \\ \mathbf{q}_\mu^{n+1} = \mathbf{q}_\mu^{n+\frac{1}{2}} + \Delta t\left[\nabla_{\mathbf{p}}\mathscr{H}\left(\mathbf{q}_\mu^n, \mathbf{p}_\mu^{n+\frac{1}{2}};\mu\right) + \nabla_{\mathbf{p}}\mathscr{H}\left(\mathbf{q}_\mu^{n+1}, \mathbf{p}_\mu^{n+\frac{1}{2}};\mu\right)\right], \\ \mathbf{p}_\mu^{n+1} = \mathbf{p}_\mu^{n+\frac{1}{2}} - \frac{1}{2}\Delta t\, \nabla_{\mathbf{q}}\mathscr{H}\left(\mathbf{q}_\mu^{n+1}, \mathbf{p}_\mu^{n+\frac{1}{2}};\mu\right). \end{cases} \tag{3.3}$$

Under the further assumption that the Hamiltonian $\mathscr{H}$ is separable, i.e. is the sum of a function depending only $\mathbf{q}$ and another depending only $\mathbf{p}$:

$$\mathscr{H}(\mathbf{y};\mu) = \mathscr{H}^1(\mathbf{q};\mu) + \mathscr{H}^2(\mathbf{p};\mu),$$

the implicit first two steps of (3.3) become explicit and the scheme simplifies into:

$$\begin{cases} \mathbf{p}_\mu^{n+\frac{1}{2}} = \mathbf{p}_\mu^n - \frac{1}{2}\Delta t\, \nabla_{\mathbf{q}}\mathscr{H}^1\left(\mathbf{q}_\mu^n;\mu\right), \\ \mathbf{q}_\mu^{n+1} = \mathbf{q}_\mu^n + \Delta t\, \nabla_{\mathbf{p}}\mathscr{H}^2\left(\mathbf{p}_\mu^{n+\frac{1}{2}};\mu\right), \\ \mathbf{p}_\mu^{n+1} = \mathbf{p}_\mu^{n+\frac{1}{2}} - \frac{1}{2}\Delta t\, \nabla_{\mathbf{q}}\mathscr{H}^1\left(\mathbf{q}_\mu^{n+1};\mu\right). \end{cases} \tag{3.4}$$

3.2.2 Hamiltonian reduced order modeling

Solving Hamiltonian systems with large dimension $2N \gg 1$ numerically can be relatively costly, and this is especially true when we want to solve a large number of them for a parametric study, for example. Therefore, methods have been developed in order to construct reduced Hamiltonian systems of smaller size $2K \ll 2N$, which capture the main dynamics for a range of times t and reduction parameters μ.

We first have to define an appropriate change of variable. To do that, we search for a $2K$-dimensional trial manifold $\widehat{\mathcal{M}}$ that approximates well the solution manifold

$$\mathcal{M} = \{\mathbf{y}(t;\mu) \text{ with } t \in [0,T], \mu \in \Xi\} \subset \mathbb{R}^{2N}$$

formed by the values taken by the solutions of the differential equation (3.1). The solution manifold structure results from the Cauchy-Lipschitz (Picard-Lindelöf) theorem with parameters under some regularity assumptions of the Hamiltonian. The trial manifold $\widehat{\mathcal{M}}$ is defined thanks to a so-called decoding operator $\mathcal{D}_{\theta_d} : \mathbb{R}^{2K} \to \mathbb{R}^{2N}$:

$$\widehat{\mathcal{M}} = \left\{\mathcal{D}_{\theta_d}(\bar{\mathbf{y}}) \text{ with } \bar{\mathbf{y}} \in \mathbb{R}^{2K}\right\} \subset \mathbb{R}^{2N}.$$

We also consider a pseudo-inverse operator $\mathcal{E}_{\theta_e} : \mathbb{R}^{2N} \to \mathbb{R}^{2K}$, called the encoder, which satisfies the relation

$$\mathcal{E}_{\theta_e} \circ \mathcal{D}_{\theta_d} = \mathrm{Id}_{\mathbb{R}^{2K}}.$$

To determine $\mathcal{D}_{\theta_d}$ and $\mathcal{E}_{\theta_e}$, we therefore ask for the projection operator $\mathcal{D}_{\theta_d} \circ \mathcal{E}_{\theta_e}$ onto $\widehat{\mathcal{M}}$ to be close to the identity on a data set $U \subset \mathcal{M}$:

$$\forall u \in U, \quad \mathcal{D}_{\theta_d} \circ \mathcal{E}_{\theta_e}(u) \approx u. \tag{3.5}$$

The data set U is composed of snapshots of the solutions at different times and various parameters, obtained with the symplectic algorithm defined above in (3.3); it writes

$$U = \left\{\mathbf{y}^0_{\mu_1}, \dots, \mathbf{y}^M_{\mu_1}, \dots, \mathbf{y}^0_{\mu_P}, \dots, \mathbf{y}^M_{\mu_P}\right\},$$

where $M \in \mathbb{N}^*$ is the number of time-step chosen and $P \in \mathbb{N}^*$ the number of sampled parameters.

In addition to these approximation properties, we also ask for the reduced variables,

$$\bar{\mathbf{y}}(t;\mu) = \mathcal{E}_{\theta_e}(\mathbf{y}(t;\mu)) \in \mathbb{R}^{2K},$$

to follow a reduced Hamiltonian dynamics:

$$\begin{cases} \dfrac{d}{dt}\bar{\mathbf{y}}(t;\mu) = J_{2K}\nabla_{\bar{\mathbf{y}}}\overline{\mathscr{H}}_{\theta_h}(\bar{\mathbf{y}}(t;\mu);\mu), & \text{in } (0,T], \\ \bar{\mathbf{y}}(0;\mu) = \mathcal{E}_{\theta_e}(\mathbf{y}_{\text{init}}(\mu)), \end{cases} \tag{3.6}$$

where $\overline{\mathscr{H}}_{\theta_h} : \mathbb{R}^{2K} \times \Xi \to \mathbb{R}$ is a reduced Hamiltonian to be built.

The most common approach for Hamiltonian reduced order modeling is called the Proper Symplectic Decomposition (PSD) [2]. This method is well described in Chapter 1, Sec. 2.3.1. Although efficient for linear dynamics, it fails into reducing nonlinear ones. This is why several nonlinear Hamiltonian reduction techniques have been developed [19, 8]. In the next section, we present a strategy based on the coupling of an autoencoder (AE) and a Hamiltonian Neural Network (HNN) method.

3.3 A nonlinear Hamiltonian reduction method

The method proposed in this work consists in constructing the Hamiltonian reduced model via neural networks. More precisely, we aim at defining the following three neural networks:

- a decoder $\mathcal{D}_{\theta_d} : \mathbb{R}^{2K} \to \mathbb{R}^{2N}$,
- an encoder $\mathcal{E}_{\theta_e} : \mathbb{R}^{2N} \to \mathbb{R}^{2K}$,
- a reduced Hamiltonian $\overline{\mathcal{H}}_{\theta_h} : \mathbb{R}^{2K} \times \Xi \to \mathbb{R}$,

where $(\theta_d, \theta_e, \theta_h)$ stands for their parameters, such that the resulting reduced dynamics provides a good approximation of the initial one. An autoencoder strategy will be used to define $\mathcal{D}_{\theta_d}$ and $\mathcal{E}_{\theta_e}$ while a Hamiltonian Neural Network will be considered for $\overline{\mathcal{H}}_{\theta_h}$. Note that the encoder and decoder are not enforced to be symplectic but the reduced model is.

Figure 3.1 illustrates how the reduced model is expected to be used for prediction. The initial condition $\mathbf{y}(t = 0; \mu)$ is converted by the encoder $\mathcal{E}_{\theta_e}$ to the reduced initial condition $\bar{\mathbf{y}}(t = 0; \mu)$. Then several iterations of the Störmer-Verlet scheme with the reduced Hamiltonian $\overline{\mathcal{H}}_{\theta_h}$ are performed to obtain an approximated reduced solution $\bar{\mathbf{y}}(t = T; \mu)$ at time T. Finally, by using the decoder $\mathcal{D}_{\theta_d}$, the latter is transformed into $\widehat{\mathbf{y}}(t = T; \mu) \approx \mathbf{y}(t = T; \mu)$. Note parameter μ has to be supplied to the Hamiltonian function.

In order to determine the appropriate parameters of the three neural networks $\mathcal{D}_{\theta_d}$, $\mathcal{E}_{\theta_e}$ and $\overline{\mathcal{H}}_{\theta_h}$, we have to define both their architectures and the loss functions used for their training. This section focuses on the latter.

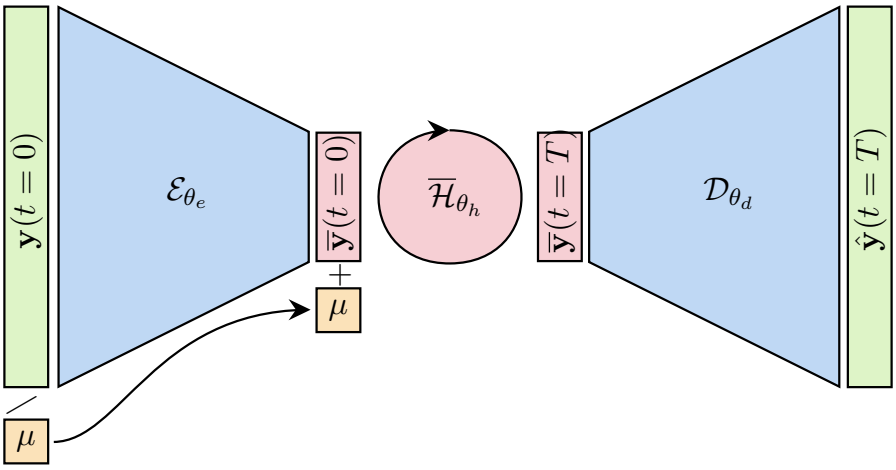


Figure 3.1: Prediction using the reduced model. The closed loop in the middle refers to the application of several iterations of the Störmer-Verlet scheme.

3.3.1 Reduction with an Auto Encoder (AE)

An autoencoder (AE) is a classical architecture of neural networks to find a reduced representation of data [15]. It is composed of two neural networks, $\mathcal{D}_{\theta_d}$ and $\mathcal{E}_{\theta_e}$, which are trained together such as to make the projection operator $\mathcal{D}_{\theta_d} \circ \mathcal{E}_{\theta_e}$ the closest to the identity map on the training data set U. Therefore, the AE is trained so as to minimize the following loss

$$\mathcal{L}_{\text{AE}}(\theta_e, \theta_d) = \sum_{\mathbf{y} \in U} \left\| \mathbf{y} - \mathcal{D}_{\theta_d} \left(\mathcal{E}_{\theta_e} \left(\mathbf{y} \right) \right) \right\|_2^2 . \tag{3.7}$$

To account for the particular structure of $\mathbf{y}$ made of coordinates and momenta, the encoder input is a tensor of size $(N, 2)$. This AE will be referred to as the bichannel AE.

Another choice would be to define two separate autoencoders for coordinates and momenta: the coordinates AE is denoted $(\mathcal{E}^1_{\theta_e,1}, \mathcal{D}^1_{\theta_d,1})$ and the momenta AE is denoted $(\mathcal{E}^2_{\theta_e,2}, \mathcal{D}^2_{\theta_d,2})$; each encoder input has shape $(N, 1)$. The AEs are trained by minimizing the loss:

$$\mathcal{L}_{\text{split,AE}}(\theta_e, \theta_d) = \sum_{(\mathbf{q},\mathbf{p})\in U} \left\|\mathbf{q} - \mathcal{D}^1_{\theta_d,1}\left(\mathcal{E}^1_{\theta_e,1}(\mathbf{q})\right)\right\|_2^2 + \left\|\mathbf{p} - \mathcal{D}^2_{\theta_d,2}\left(\mathcal{E}^2_{\theta_e,2}(\mathbf{p})\right)\right\|_2^2.$$

These AEs will be called the split AE. This split AE will be used to preserve the separability property of the Hamiltonian, where applicable (see Rem. 3.3.2 below).

The architectures of the neural networks are chosen specifically to the Hamiltonian systems in consideration. In this work, we focus on systems resulting from the spatial discretization of wave-like equations: networks will be more efficient if they take into account the spatial structure of the data. Consequently, the encoder $\mathcal{E}_{\theta_e}$ is first composed of several pairs of convolution layer and down-sampling before ending with some dense layers, as depicted in Figure 3.2. As usual for AE networks, the decoder is constructed in a mirror way, i.e. starting with some dense layers and then ending with pairs of up-sampling and convolution layers in reversed size order.

More precisely, the encoder takes as input a vector $\mathbf{y} = (\mathbf{q}, \mathbf{p})$ and starts with a succession of so-called encoder blocks, made of a stride 1 convolution with kernel size 3 and a down-sampling step (stride 2 convolution with kernel size 2). An encoder block results in an output that has twice as many channels and half as many rows as the input. After possibly composing several encoder blocks, we add a last convolution layer with kernel size 3 and then a flattening operation by concatenating every channels. Then, dense layers are added until reaching the desired reduced dimension $2K$ of $\bar{\mathbf{y}}$. As already said, the decoder is built as a mirror: the flattening operation is replaced by unflattening and the encoder blocks by the decoder blocks made of an up-sampling layer of size 2 smoothed out with a convolution with kernel size 2 and a convolution layer with kernel size 3, which symmetrically results in output that has half as many channels and twice as many rows as the input. The architecture of the autoencoder is thus defined with the number of encoder and decoder blocks and the dense layer sizes for both encoder and decoder. Figure 3.2 illustrates an example of autoencoder architecture with one block for the encoder and one block for the decoder. This AE architecture can easily be extended to 2D systems using two-dimensional convolutionlayers and up and down-sampling with appropriate dimensions.

3.3.2 Reduced model with a Hamiltonian Neural Network (HNN)

The AE constructed in the previous section enables us to define the reduced trajectories:

$$\bar{\mathbf{y}}(t;\mu) = \mathcal{E}_{\theta_e}(\mathbf{y}(t;\mu)). \tag{3.8}$$

To obtain the dynamics of these reduced variables, we propose to use a Hamiltonian Neural Network strategy [26]. We thus look for a neural network function $\overline{\mathscr{H}}_{\theta_h}$, parameterized by θ_h, such that:

$$\frac{d}{dt}\bar{\mathbf{y}}(t;\mu) = J_{2K}\nabla_{\bar{\mathbf{y}}}\overline{\mathscr{H}}_{\theta_h}(\bar{\mathbf{y}}(t;\mu);\mu).$$

Note that the Hamiltonian is supposed to depend on parameter μ. We remind that $\mu \in \Xi$ stands for known parameters of the model, unlike θ_h that is the neural networks parameters to be learnt. The architecture of the reduced Hamiltonian is a classical MLP neural network. The size of the neural network is chosen to be small so that the reduced model remains competitive. This reduced dynamics are in practice defined through a time discretization. We therefore introduce the prediction operator:

$$\mathcal{P}_s\left(\bar{\mathbf{y}}; \overline{\mathscr{H}}_{\theta_h,\mu}\right),$$

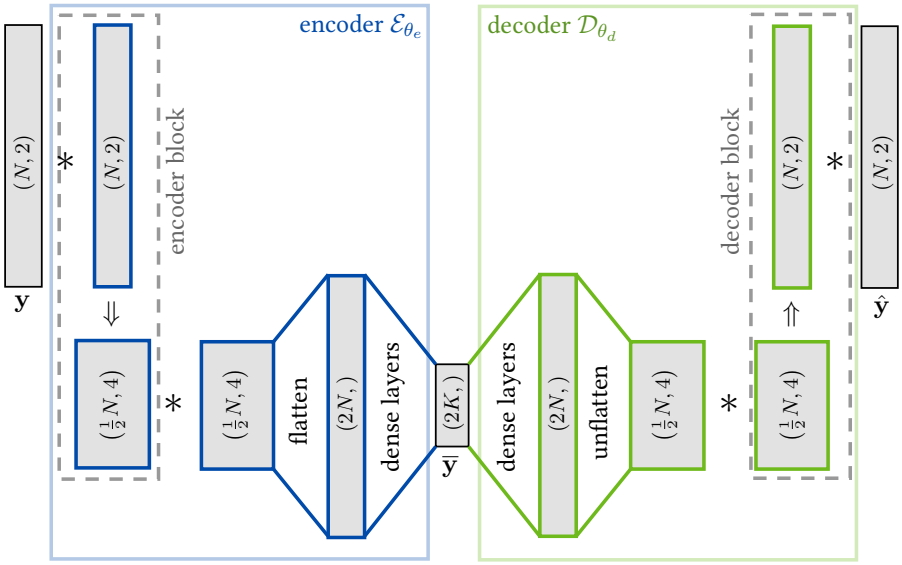


Figure 3.2: Autoencoder architecture: encoder in blue, decoder in green. Symbols. $*$: stride 1 convolution with periodic padding, $\Downarrow$ down-sampling (stride 2 convolution), $\Uparrow$: up-sampling (repeat once each value along the last axis then smooth it with a kernel size 2 convolution).

which consists in performing $s \in \mathbb{N}^*$ iterations of the Störmer-Verlet scheme, defined in (3.3), starting from $\bar{\mathbf{y}}$ and where $\overline{\mathscr{H}}_{\theta_h,\mu}$ stands for the Hamiltonian function $\overline{\mathscr{H}}_{\theta_h}(.;\mu)$. The number of steps s considered in this prediction is called the watch duration. This is a hyper-parameter of the method that has to be set. The parameters of the reduced Hamiltonian are finally obtained by minimizing the following loss:

$$\mathcal{L}^s_{\overline{\text{pred}}}(\theta_e, \theta_h) = \sum_{\mathbf{y}^n_\mu, \mathbf{y}^{n+s}_\mu \in U} \left\| \bar{\mathbf{y}}^{n+s}_\mu - \mathcal{P}_s\left(\bar{\mathbf{y}}^n_\mu; \overline{\mathscr{H}}_{\theta_h,\mu}\right) \right\|_2^2 \tag{3.9}$$

where $\mathbf{y}^n_\mu, \mathbf{y}^{n+s}_\mu \in U$ denote the sampling of random pairs $(\mathbf{y}^n_\mu, \mathbf{y}^{n+s}_\mu)$ on the data set U. In other words, random time series of size s are sampled and only the data at both ends are considered. This loss thus compares the reduced trajectories (3.8) with the ones obtained with the reduced Hamiltonian. The name of the loss function, "$\overline{\text{pred}}$", refers to the prediction in the reduced variables. Note that the encoder neural network is required to obtain the reduced data $\bar{\mathbf{y}} = \mathcal{E}_{\theta_e}(\mathbf{y})$ and this is why the loss also depends on θ_e. This kind of loss function, based on a model, has been widely used in physics based deep learning methods [31].

Then, we constrain the reduced trajectories to preserve the reduced Hamiltonian with the following loss function :

$$\mathcal{L}^s_{\overline{\text{stab}}}(\theta_e, \theta_h) = \sum_{\mathbf{y}^n_\mu, \mathbf{y}^{n+s}_\mu \in U} \left\| \overline{\mathscr{H}}_{\theta_h,\mu}\left(\bar{\mathbf{y}}^{n+s}_\mu\right) - \overline{\mathscr{H}}_{\theta_h,\mu}\left(\bar{\mathbf{y}}^n_\mu\right) \right\|_2^2, \tag{3.10}$$

where $\bar{\mathbf{y}}^{n+s}_\mu$ and $\bar{\mathbf{y}}^n_\mu$ are still obtained using the encoder $\mathcal{E}_{\theta_e}$. The aim is to ensure some stability of the reduced model, hence its name "$\overline{\text{stab}}$" . At first sight, this loss seems redundant with the prediction-reduced loss since using Störmer-Verlet schemes in the prediction step ensures that the reduced Hamiltonian is preserved at least approximately. Hence if the prediction-reduced loss becomes small, so does the stability loss. However, this additional loss may help the coupling to converge.

Remark. *The separability of the Hamiltonian could be an interesting property to preserve at the reduced level. Using a split AE to learn separately reduced coordinates and momenta, a separable reduced Hamiltonian can be designed:*

$$\overline{\mathscr{H}}_{\theta_h}(\bar{\mathbf{y}};\mu) = \overline{\mathscr{H}}^1_{\theta_h}(\bar{\mathbf{q}};\mu) + \overline{\mathscr{H}}^2_{\theta_h}(\bar{\mathbf{p}};\mu),$$

involving two neural networks. This will be referred to as the split HNN. The Störmer-Verlet scheme would then have a cost of an explicit scheme. Note however that this is not a crucial gain since the reduced models under consideration have small sizes.

Remark. *The HNN will produce an approximation of the Hamiltonian, whose associated flow would have generated the discrete solution we would like to fit. Note also that the data used to fit the HNN was obtained after encoding a discrete solution of the initial dynamics of the Hamiltonian. Thus, even this discrete dynamics is an approximation to the encoded continuous Hamiltonian flow. However, from a practical point of view, it is not essential to capture the underlying continuous dynamics (at the reduced level): the objective is rather to obtain a method capable of reproducing the discrete dynamics, with good geometric properties, for a given time step. So, in practice, we actually do not vary the time step.*

3.3.3 Strong coupling of the neural networks

The prediction-reduced and the stability-reduced losses (3.9) already introduce a coupling between the encoder neural networks and the reduced Hamiltonian one. To make the coupling stronger, we could ask for the trajectories in the initial variables to be well recovered. This is why we introduce the following fourth loss function named " pred " which refers to the model prediction in the FOM space:

$$\mathcal{L}^s_{\text{pred}}(\theta_e,\theta_d,\theta_h) = \sum_{\mathbf{y}^n_\mu,\mathbf{y}^{n+s}_\mu \in U} \left\| \mathbf{y}^{n+s}_\mu - \mathcal{D}_{\theta_d}\left(\mathcal{P}_s\left(\bar{\mathbf{y}}^n_\mu; \overline{\mathscr{H}}_{\theta_h,\mu}\right)\right)\right\|_2^2. \tag{3.11}$$

This is the only loss function that couples the three neural networks. It compares the trajectories in the initial variables with the full process of encoding, predicting in reduced variables over s iterations and then decoding.

To sum up, we use four different loss functions $\mathcal{L}_{\text{AE}}, \mathcal{L}^s_{\overline{\text{pred}}}, \mathcal{L}_{\overline{\text{stab}}}$ and $\mathcal{L}^s_{\text{pred}}$, given by (3.7)-(3.9)-(3.10)-(3.11) that couple both AE and HNN neural networks. The training aims to find the parameters $(\theta_e,\theta_d,\theta_h)$ that are a solution to the minimization problem:

$$\min_{\theta_e,\theta_d,\theta_h} \quad \omega_{\text{AE}}\,\mathcal{L}_{\text{AE}}(\theta_e,\theta_d) + \omega_{\overline{\text{pred}}}\,\mathcal{L}^s_{\overline{\text{pred}}}(\theta_e,\theta_h) + \omega_{\overline{\text{stab}}}\,\mathcal{L}^s_{\overline{\text{stab}}}(\theta_e,\theta_h) + \omega_{\text{pred}}\,\mathcal{L}^s_{\text{pred}}(\theta_e,\theta_d,\theta_h),$$

where ω_{AE}, $\omega_{\overline{\text{pred}}}$, $\omega_{\overline{\text{stab}}}$, ω_{pred} are positive weights: these are hyper-parameters of the method. The four loss functions interact during training and possibly compete with each other. Note that in the end, the only loss value that really quantifies the quality of the reduction and prediction process is the one corresponding to $\mathcal{L}^s_{\text{pred}}$. The other loss functions are only useful in the training process.

3.3.4 Training hyper-parameters

In addition to the parameters of the neural networks (number of layers, size of the layers), the training of the model also depends on several hyper-parameters.

Reduced dimension K. In classical reduction method, the larger K, the more accurate the reduced model. Regarding the AE-HNN method, as the approximation is truly nonlinear, there may be no benefit increasing the reduced dimension. In practice, the minimum possible reduced dimension should be equal to the number of variable parameters in the model.

Watch duration in predictions. One of the hyper-parameter to set is the watch duration s in the loss functions $\mathcal{L}^s_{\overline{\text{stab}}}$, $\mathcal{L}^s_{\overline{\text{pred}}}$ and $\mathcal{L}^s_{\text{pred}}$ that make predictions. This quantity should be not too small to capture the dynamics but also not too large as the computation of the gradients of the associated loss functions may generate vanishing gradient problems. In the numerical setting, the watch duration will be typically set to $s = 16$.

Loss functions weights. Losses weights have been chosen experimentally as follows:

$$\omega_{\text{pred}} = 0.1, \quad \omega_{\text{AE}} = 0.1, \quad \omega_{\overline{\text{pred}}} = 80, \quad \omega_{\overline{\text{stab}}} = 7 \times 10^{-4}.$$

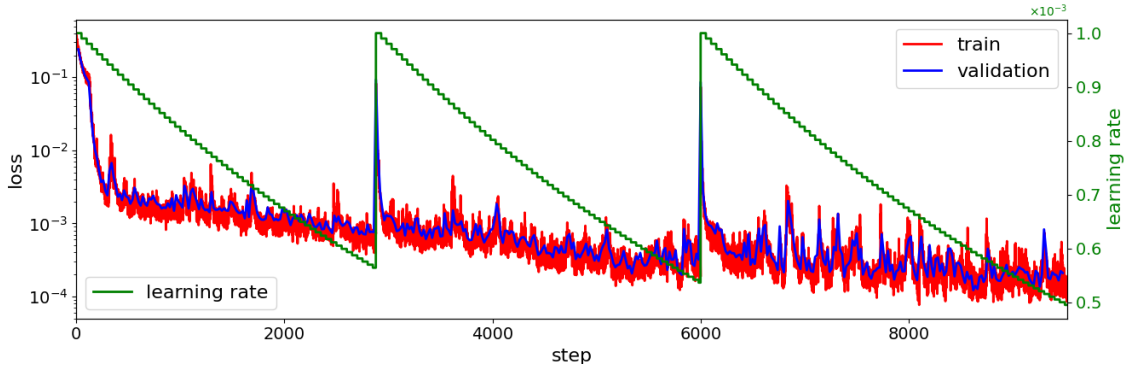
A typical loss functions history is shown in Figure 3.3b: each loss function is represented multiplied by its weight. We first notice that the prediction loss function $\mathcal{L}^s_{\text{pred}}$ and the autoencoder loss function $\mathcal{L}_{\text{AE}}$ have the same magnitude and actually are almost equal: the error in prediction is mostly due to the encoder-decoder step. We keep this behavior by assigning them the same weight $\omega_{\text{pred}} = \omega_{\text{AE}} = 0.1$. This value is determined in proportion to the learning rate chosen below. As the prediction loss function $\mathcal{L}^s_{\text{pred}}$ is the most important one for the applications, we want it to dominate over the others. We therefore set the weight $\omega_{\overline{\text{pred}}} = 80$ so that the weighted reduced prediction loss function $\omega_{\overline{\text{pred}}}\mathcal{L}^s_{\overline{\text{pred}}}$ is about 10 times smaller than the previous two. Finally, we want the weighted reduced stability loss function $\omega_{\overline{\text{stab}}}\mathcal{L}^s_{\overline{\text{stab}}}$ to act as a quality control that remains small compared to the other loss functions. To this end, we set $\omega_{\overline{\text{stab}}} = 7 \times 10^{-4}$ in order to make it about 100 times smaller than the weighted reduced prediction loss.

Gradient descent. An Adam optimizer [32] is used for the training. The learning rate follows the following rule:

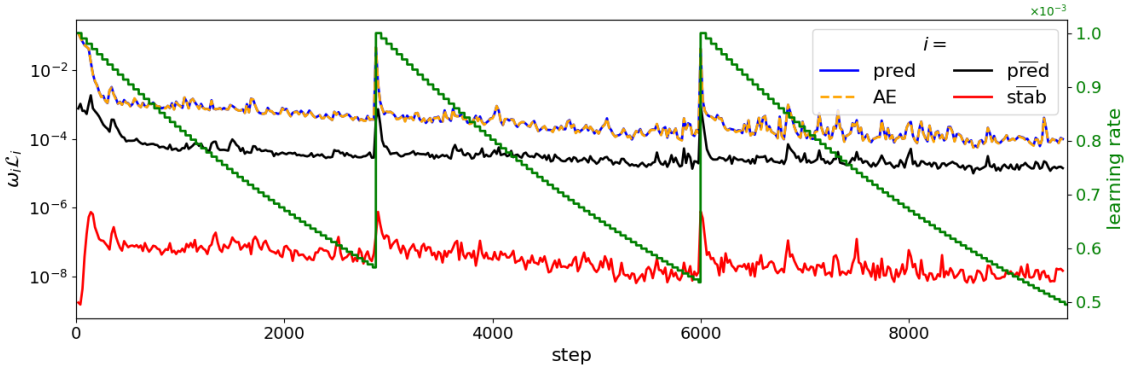
$$\rho_k = (0.99)^{k/150}\, 0.001.$$

where the division operator stands here for the integer division, and k is the train step. Thus, the learning rate is constant over 150 iterations, then decreases. It has an exponential decay with the shape of a staircase. In addition, we can reset this decay i.e. set $k = 0$ at any time if we notice that the loss is reaching a plateau. The main goal of this reset strategy is to escape poor local minima of the minimization problem with a sudden large learning rate. On Figure 3.3a is shown a typical training and validation loss history as functions of the training step k as well as the learning rate at each step. The resets enable us to make the training loss go from 1×10^{-3} to 5×10^{-4} and then to 1×10^{-4}. With the loss functions weights above-mentioned, we consider 1×10^{-5} to be a correct plateau value to stop the training process. The training phase lasts from 1 to 3 hours on a shared NVIDIA Tesla T4 GPU.

Pre-processing Data pre-processing is required to optimize the learning process. Here we use usual standardization techniques.



(a) Training loss function (blue) and validation loss function (red) as functions of the training step.



(b) All the weighted loss functions as functions of the training step.

Figure 3.3: Example of loss functions history during a training, overlaid with the evolution of the learning rate (green).

3.3.5 Numerical complexity

Here, we briefly compare the computational gain in using the reduced models in the online phase. For the original system, the main cost comes from evaluating the N components of the Hamiltonian gradient. If we denote by α the evaluation complexity for one component, the computational cost is therefore about $O(N\alpha)$. When using the reduced PSD model, an additional cost arises from the linear encoding-decoding operations and the computational cost is equal to $O(N\alpha + NK)$. The use of the DEIM-PSD method, as presented in Chapter 1, Sec. 2.3.3, allows us to rely only on m components of the gradient of the Hamiltonian and the computation time is therefore about $O(m\alpha + mK)$, which no longer depends on the N dimension. On the other hand, the reduced HNN model relies on the evaluation of the gradient of the neural network Hamiltonian, whose evaluation complexity is, to a first approximation, equivalent to a direct evaluation. Thus, if we denote by n^k the width (i.e. the number of neurons) of the k-th layer and only count the linear operations between the layers, the total complexity of the evaluation is about $O(\sum_{k=1}^{L} n^{(k-1)}n^{(k)})$. Therefore, if the width is of the order of K^2, the complexity of the evaluation is of the order of $O(K^4)$. Depending on the values of α and m, the reduced model AE-HNN can be competitive. It should also be noted that an additional advantage of the AE-HNN reduced model is that it can naturally be evaluated for a batch of parameters in parallel.

3.4 Numerical results

This section is devoted to the numerical results obtained with the proposed Hamiltonian reduction method. We consider one-dimensional discretizations of three wave-type equations: the linear wave equation, the nonlinear wave equation and the shallow water equation.

3.4.1 Wave equations

We introduce a parameterized one-dimensional wave equation:

$$\begin{cases} \partial_{tt} u(x,t;\mu) - \mu_a \, \partial_x \left(w' \left(\partial_x u(x,t;\mu), \mu_b \right) \right) + g'(u(x,t;\mu), \mu_c) = 0, & \text{in } \Omega_1 \times (0,T], \\ u(x,0;\mu) = u_{\text{init}}(x;\mu), & \text{in } \Omega_1, \\ \partial_t u(x,0;\mu) = v_{\text{init}}(x;\mu), & \text{in } \Omega_1, \end{cases} \tag{3.12}$$

complemented with periodic boundary conditions. The solution $u(x,t;\mu)$ represents the vertical displacement of a string over the interval Ω_1. The model depends on two given functions $w, g : \mathbb{R} \to \mathbb{R}$ and three parameters: $\mu = (\mu_a, \mu_b, \mu_c)^T \in \Xi \subset \mathbb{R}^3_+$.

Defining the vertical displacement velocity $v(x,t;\mu) = \partial_t u(x,t;\mu)$, Equation (3.12) can be reformulated as a first order in time system:

$$\begin{cases} \partial_t u(x,t;\mu) - v(x,t;\mu) = 0, & \text{in } \Omega_1 \times (0,T], \\ \partial_t v(x,t;\mu) + \mu_a \partial_x \left(w' \left(\partial_x u(x,t;\mu), \mu_b \right) \right) + g'(u(x,t;\mu), \mu_c) = 0, & \text{in } \Omega_1 \times (0,T] \\ u(x,0;\mu) = u_{\text{init}}(x;\mu), & \text{in } \Omega_1, \\ v(x,0;\mu) = v_{\text{init}}(x;\mu), & \text{in } \Omega_1. \end{cases}$$

Then, we consider a spatial finite difference discretization of this system. Introducing a uniform mesh of the interval Ω_1, with N cells, space step $\Delta x = 1/(N-1)$ and nodes $x_i = i\Delta x$, for $i \in \{0, \dots, N-1\}$, the approximate solution $(\mathbf{u}, \mathbf{v}) = (u_i(t;\mu), v_i(t;\mu))_{i \in \{0,\dots,N-1\}} \in \mathbb{R}^N \times \mathbb{R}^N$ satisfies the following system:

$$\begin{cases} \dfrac{d}{dt} u_i(t;\mu) = v_i(t;\mu), & \text{in } (0,T], \\ \dfrac{d}{dt} v_i(t;\mu) = -\dfrac{\mu_a}{\Delta x} \left(w' \left(\dfrac{u_{i+1} - u_i}{\Delta x}, \mu_b \right) - w' \left(\dfrac{u_i - u_{i-1}}{\Delta x}, \mu_b \right) \right) \\ \qquad\qquad + g'(u_i, \mu_c), & \text{in } (0,T], \\ u_i(0;\mu) = u_{\text{init}}(x_i;\mu), \\ v_i(0;\mu) = v_{\text{init}}(x_i;\mu). \end{cases} \tag{3.13}$$

These equations actually form a Hamiltonian system of size $2N$, with a separable Hamiltonian function ($(\mathbf{u}, \mathbf{v})$ stands for variables $(\mathbf{q}, \mathbf{p})$) given by

$$\mathscr{H}(\mathbf{u}, \mathbf{v}; \mu) = \mathscr{H}^1(\mathbf{u}; \mu) + \mathscr{H}^2(\mathbf{v}; \mu), \tag{3.14}$$

with

$$\mathscr{H}^1(\mathbf{u};\mu) = \Delta x \sum_{i=0}^{N-1} \left(\mu_a w \left(\frac{u_{i+1} - u_i}{\Delta x}, \mu_b \right) + \mu_a w \left(\frac{u_i - u_{i-1}}{\Delta x}, \mu_b \right) + g(u_i, \mu_c) \right),$$

$$\mathscr{H}^2(\mathbf{v};\mu) = \frac{1}{2} \Delta x \sum_{i=0}^{N-1} v_i^2.$$

In the following, we test the AE-HNN method for two choices of functions w and g. The same hyper-parameters are used in both test-cases. They are summarized in Table 3.1.

		wave equations	shallow water 1D	shallow water 2D
AE	type	split	bichannel	bichannel
	nb of convolution blocks (encoder)	4	4	4
	dense layers (encoder)	$[256, 128, 64, 32]$	$[256, 128, 64, 32]$	$[512, 256, 128, 64, 32]$
	activation functions	ELU	swish	ELU
HNN	type	split	standard	standard
	dense layers	$[24, 12, 12, 12, 6]$	$[40, 20, 20, 20, 10]$	$[96, 48, 48, 48, 24]$
	activation functions	tanh	swish	tanh
watch duration	s	16	48	16

Table 3.1: Hyper-parameters. Activation functions are used except for the last layer of the neural networks. ELU refers to the function $\text{elu}(x) = x1_{x>0} + (e^x - 1)1_{x<0}$ and swish to the function $\text{swish}(x) = x/(1 + e^{-x})$. For the autoencoder (AE), the number of convolution blocks and the sizes of the hidden of layers are those of the encoder. The decoder is constructed in a mirror way.

3.4.1.1 Reduction of the linear wave equation

Here we consider the linear wave equation:

$$\begin{cases} \partial_{tt} u(x,t;\mu) - \mu_a \, \partial_{xx} u(x,t;\mu) = 0, & \text{in } \Omega_1 \times (0,T], \\ u(x,0;\mu) = u_{\text{init}}(x;\mu), & \text{in } \Omega_1. \end{cases}$$

corresponding to $w(x,\mu_b) = \frac{1}{2}x^2$ and $g(x,\mu_c) = 0$. In particular, we conserve only one parameter $\mu_a \in [0.2, 0.6]$, which corresponds to the square of the wave velocity. The initial condition is taken equal to:

$$u_{\text{init}}(x;\mu) = h(10|x - \tfrac{1}{2}|), \quad v_{\text{init}}(x;\mu) = 0. \tag{3.15}$$

where h is the compactly supported (single bump) function:

$$h(r) = \left(1 - \tfrac{3}{2}r^2 + \tfrac{3}{4}r^3\right) 1_{\Omega_1}(r) + \tfrac{1}{4}(2-r)^3 1_{(1,2]}(r). \tag{3.16}$$

The initial condition can be observed in Figure 3.5a. We consider $N = 1024$ discretization points, set $T = 0.4$ and $\Delta t = 1 \times 10^{-4}$. In Figure 3.4, we observe the solution at final time $T = 0.4$ as a function of μ_a. Each color corresponds to a different value of μ_a. As we can expect, large values of μ_a generate waves farther from the initial condition.

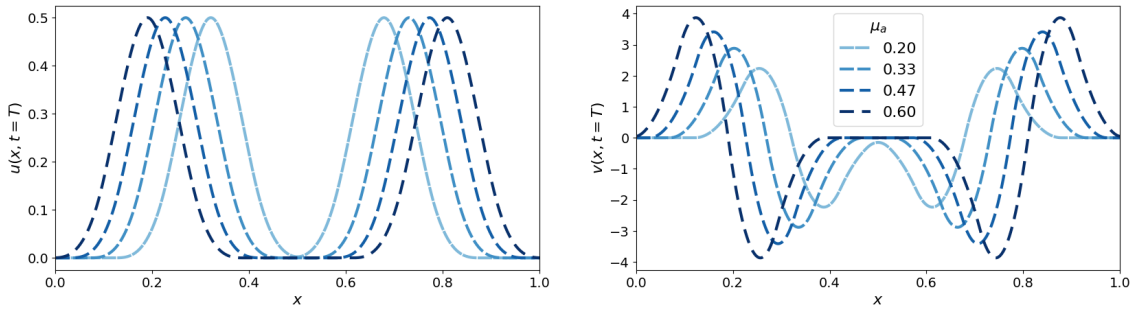


Figure 3.4: (Linear wave) Solution $(\mathbf{u}, \mathbf{v})$ at final time $T = 0.4$ for various parameters $\mu = \mu_a$

To train the model, we consider numerical solutions obtained with $P = 20$ different values μ_a taken regularly spaced in the interval $[0.2, 0.6]$. The validation set is obtained considering numerical solutions with 6 different parameter values in the same interval and 128 random pairs

$(\mathbf{y}^n, \mathbf{y}^{n+s})$ for each validation parameter. The final value of the validation loss functions are given in Table 3.2.

Then we test the obtained model on 3 test values:

$$\mu_a = 0.2385 \text{ (test 1)}, \quad \mu_a = 0.3798 \text{ (test 2)}, \quad \mu_a = 0.5428 \text{ (test 3)}.$$

To evaluate our model, we compute the relative L^2 discrete error between the reference solution u_{ref} and the reduced model prediction u_{pred}:

$$\text{err} = \frac{\sum_{n=1}^{M} \Delta t \left(\sum_{i=1}^{N} \Delta x \, (u_{\text{ref},i}^n - u_{\text{pred},i}^n)^2 \right)}{\sum_{n=1}^{M} \Delta t \left(\sum_{i=1}^{N} \Delta x \, (u_{\text{pred},i}^n)^2 \right)},$$

where M is the total number of time steps. Table 3.3 compares the errors of the AE-HNN method with the ones obtained with the PSD and the POD for different reduced dimensions K. We have colored in blue the cells of the PSD and POD methods with an error lower than the AE-HNN for $K = 1$ (in yellow). As shown in Table 3.3, we succeed in capturing the dynamics with a AE-HNN model of dimension $K = 1$ with a relative error equal to 1×10^{-2}. To obtain comparable performance, $K = 6$ modes are needed for the PSD and $K = 10$ modes for the POD. The reduction capability of the PSD is nevertheless quite good and this is due to the linearity of the problem. Also, the POD results show that taking into account the symplecticity of the problem enables us to improve significantly the reduction.

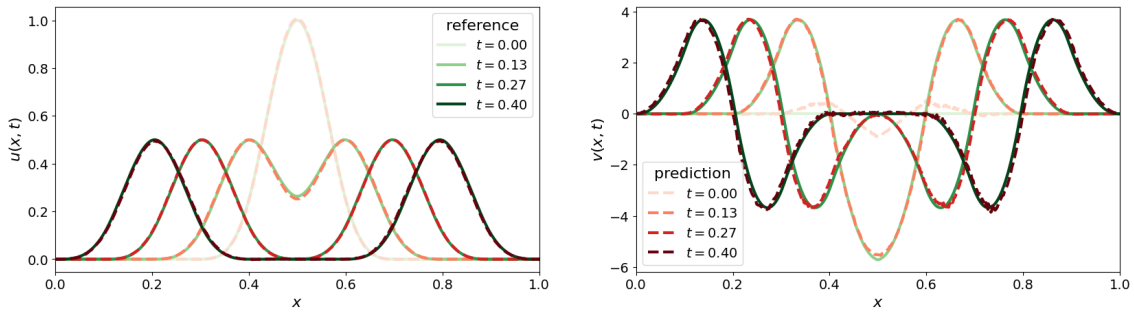
Figure 3.5 shows the simulations of the reduced models for both the AE-HNN ($K = 1$) and the PSD ($K = 6$) methods at different times for test 3. As expected, both methods provide good results.

	$\mathcal{L}^s_{\text{pred}}$	$\mathcal{L}_{\text{AE}}$	$\mathcal{L}^s_{\overline{\text{pred}}}$	$\mathcal{L}^s_{\overline{\text{stab}}}$
linear wave	8.25×10^{-4}	8.19×10^{-4}	2.52×10^{-7}	1.54×10^{-5}
nonlinear wave	3.47×10^{-5}	3.53×10^{-5}	1.84×10^{-8}	1.98×10^{-6}
shallow water 1D	6.49×10^{-5}	6.51×10^{-5}	7.70×10^{-9}	1.19×10^{-7}
shallow water 2D	9.19×10^{-5}	9.11×10^{-5}	3.80×10^{-8}	3.29×10^{-7}

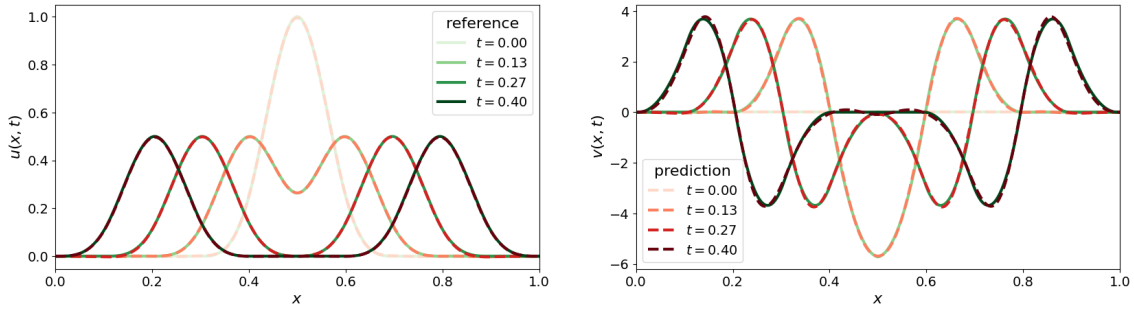
Table 3.2: Values of the loss functions on validation data for the different test-cases.

		test 1		test 2		test 3	
		error u	error v	error u	error v	error u	error v
AE-HNN	$K = 1$	1.20×10^{-2}	6.05×10^{-2}	7.07×10^{-3}	5.84×10^{-2}	1.43×10^{-2}	8.09×10^{-2}
	$K = 2$	6.71×10^{-3}	2.24×10^{-2}	1.48×10^{-2}	3.99×10^{-2}	9.71×10^{-3}	2.30×10^{-2}
	$K = 5$	1.67×10^{-2}	2.54×10^{-2}	1.48×10^{-2}	2.90×10^{-2}	1.79×10^{-2}	3.55×10^{-2}
PSD	$K = 4$	6.70×10^{-1}	3.67×10^{-1}	7.01×10^{-1}	8.69×10^{-1}	7.30×10^{-1}	3.65×10^{-1}
	$K = 5$	1.28×10^{-1}	1.34×10^{-1}	1.60×10^{-1}	1.43×10^{-1}	1.91×10^{-1}	1.52×10^{-1}
	$K = 6$	4.80×10^{-3}	2.11×10^{-2}	5.52×10^{-2}	2.14×10^{-2}	5.89×10^{-3}	2.23×10^{-2}
POD	$K = 6$	3.07×10^{-2}	1.08×10^{-1}	1.30×10^{-2}	2.67×10^{-2}	6.00×10^{-2}	1.54×10^{-1}
	$K = 8$	1.28×10^{-2}	4.89×10^{-2}	1.12×10^{-2}	2.66×10^{-2}	3.96×10^{-2}	7.05×10^{-2}
	$K = 10$	9.78×10^{-3}	4.36×10^{-2}	2.30×10^{-3}	9.07×10^{-3}	1.58×10^{-2}	5.93×10^{-2}

Table 3.3: (Linear wave) Relative L^2 errors for different reduced dimensions K. Blue cells correspond to POD and PSD simulations with lower errors than the corresponding AE-HNN simulation in yellow.



(a) AE-HNN, $K = 1$



(b) PSD, $K = 6$

Figure 3.5: (Linear wave) Solution $(\mathbf{u}, \mathbf{v})$ at different times on test 3, reference solution (full lines) and prediction (dashed lines).

3.4.1.2 Reduction of nonlinear wave equation

Now, we consider the following nonlinear wave equations over the domain $\Omega_1 = [0, 1]$:

$$\begin{cases} \partial_{tt} u(x,t;\mu) - \mu_a \, \partial_{xx} u(x,t;\mu) - \mu_b \, \partial_x \left(\cos(\mu_b \partial_x u(x,t;\mu))\right) \\ \qquad\qquad + 30\mu_c x^2 = 0, & \text{in } \Omega_1 \times (0,T], \\ u(x,0;\mu) = u_{\text{init}}(x;\mu), & \text{in } \Omega_1. \end{cases}$$

corresponding to $w(x, \mu_b) = \frac{1}{2}x^2 + \sin(\mu_b x)$ and $g(x, \mu_c) = 10\mu_c x^3$. There are three parameters $\mu_a \in [0.2, 0.6]$, $\mu_b \in [0.025, 0.5]$ and $\mu_c \in [0.4, 2.4]$. The initial condition is given by Eq. (3.15)-(3.16).

The number of discretization points is still $N = 1024$ but the final time is taken equal to $T = 0.3$ and the time step $\Delta t = 1 \times 10^{-4}$. These parameters have been chosen so that the numerical simulations remain stable despite the strong nonlinearities. In Figure 3.6, we observe the numerical solutions at final time for several sets of parameters.

Training data are obtained using $P = 20$ different values (μ_a, μ_b, μ_c) regularly spaced in the segment $[(0.2, 0.025, 0.4), (0.6, 0.5, 2.4)]$. The validation set is also made of 6 different triplets define on the same domain. At the end of the training, the validation loss takes the values given in Table 3.2.

The obtained model is tested on 3 sets of parameters:

$$\begin{aligned} (\mu_a, \mu_b, \mu_c) &= (0.2385, 0.088, 0.5485) \qquad \text{(test 1)}, \\ (\mu_a, \mu_b, \mu_c) &= (0.3785, 0.281, 1.354) \qquad \text{(test 2)}, \\ (\mu_a, \mu_b, \mu_c) &= (0.5528, 0.437, 2.128) \qquad \text{(test 3)}, \end{aligned}$$

and Table 3.4 presents the relative errors. A relative error of order 1×10^{-2} can be reached with the AE-HNN method with a reduced dimension of $K=3$ only. In comparison, the PSD and the POD require $K=15$ and $K=30$ reduced dimensions to obtain similar results.

Figure 3.7 shows numerical solutions obtained for parameters corresponding to test 3. While the AE-HNN method with $K=3$ remains close to the reference solution (Fig. 3.7a), the PSD with the same reduced dimension does not provide satisfactory results (Fig. 3.7b). Increasing the reduced dimension to $K=15$ for the PSD allows us to recover comparable results (Fig. 3.7c).

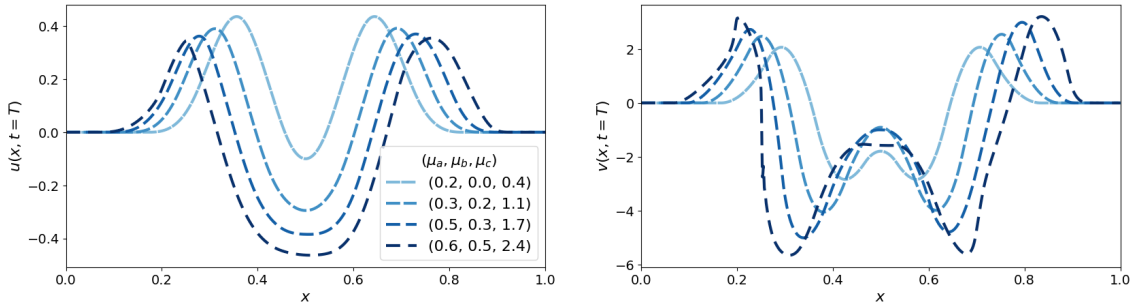
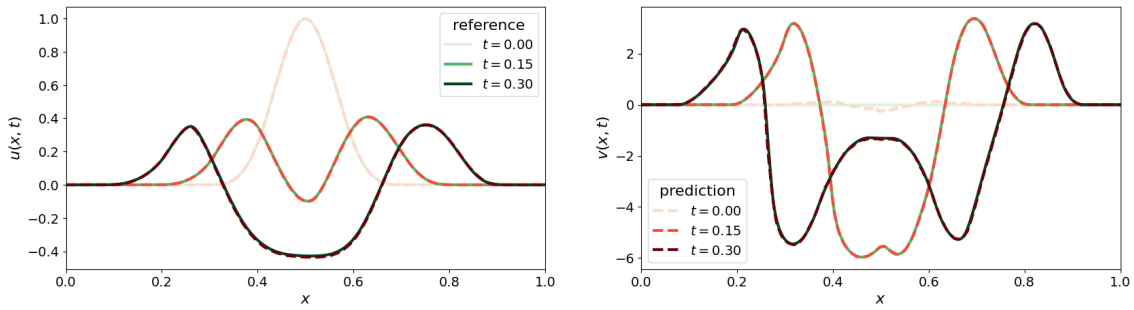


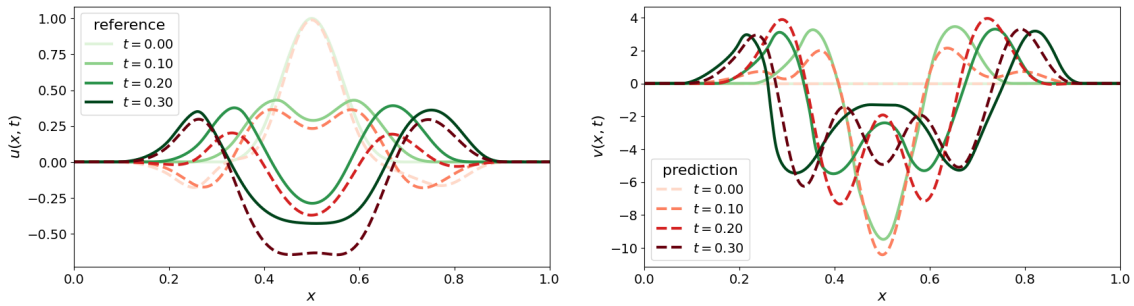
Figure 3.6: (Nonlinear wave) final condition $(\mathbf{u},\mathbf{v})(t=T;\mu)$ for various parameters $\mu=(\mu_a,\mu_b,\mu_c)$

		test 1		test 2		test 3	
		error u	error v	error u	error v	error u	error v
AE-HNN	$K=3$	4.34×10^{-3}	1.17×10^{-2}	5.82×10^{-2}	8.65×10^{-2}	1.06×10^{-2}	1.36×10^{-2}
PSD	$K=3$	4.04×10^{-1}	2.55×10^{-1}	4.21×10^{-1}	2.49×10^{-1}	4.33×10^{-1}	2.73×10^{-1}
	$K=10$	3.26×10^{-2}	4.92×10^{-2}	3.84×10^{-2}	4.53×10^{-2}	5.63×10^{-2}	4.94×10^{-2}
	$K=15$	8.48×10^{-3}	1.66×10^{-2}	6.45×10^{-3}	1.66×10^{-2}	5.29×10^{-3}	1.87×10^{-2}
POD	$K=3$	1.86×10^{-1}	3.16×10^{-1}	5.72×10^{-2}	6.66×10^{-2}	2.44×10^{-1}	2.89×10^{-1}
	$K=20$	2.08×10^{-2}	4.00×10^{-2}	3.10×10^{-2}	7.00×10^{-2}	3.66×10^{-3}	1.01×10^{-2}
	$K=30$	5.36×10^{-3}	2.09×10^{-2}	9.54×10^{-3}	2.26×10^{-2}	8.87×10^{-3}	1.93×10^{-2}

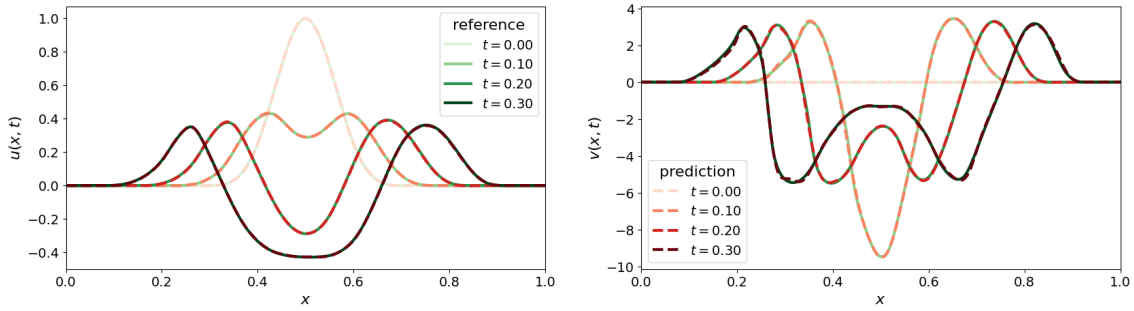
Table 3.4: (Nonlinear wave) Relative L^2 errors for different reduced dimensions K and different parameters. Blue cells correspond to POD and PSD simulations with lower errors than the corresponding AE-HNN simulation in yellow.



(a) AE-HNN, $K = 3$



(b) PSD, $K = 3$



(c) PSD, $K = 15$

Figure 3.7: (Nonlinear wave) Solutions $(\mathbf{u}, \mathbf{v})$ at different times on test 3. Reference solution in solid lines and prediction in dashed lines.

3.4.1.3 Comparison with a non Hamiltonian reduction

In this section, we want to highlight the importance of the Hamiltonian framework for the reduced dynamics. Indeed, if the Hamiltonian structure is ignored, then the reduced system can be written:

$$\begin{cases} \dfrac{d}{dt}\bar{\mathbf{y}}(t;\mu) = \overline{\mathcal{F}}(\bar{\mathbf{y}}(t;\mu);\mu), \quad \forall t \in (0,T], \\ \bar{\mathbf{y}}(0;\mu) = \mathcal{E}(\mathbf{y}_{\text{init}}(\mu)), \end{cases} \tag{3.17}$$

where $\overline{\mathcal{F}} : \mathbb{R}^K \times \Xi \to \mathbb{R}^K$ is the reduced vector field. Then the reduced dynamics can be learn by designing a neural network approximation $\overline{\mathcal{F}}_{\theta_f}$, with a classical MLP architecture. To this aim, we can consider the following loss function:

$$\mathcal{L}^s_{\text{Flow}}(\theta_e, \theta_f) = \sum_{\mathbf{y}^n_\mu, \mathbf{y}^{n+s}_\mu \in U} \left\| \bar{\mathbf{y}}^{n+s}_\mu - \mathcal{P}_s\left(\bar{\mathbf{y}}^n_\mu; \overline{\mathcal{F}}_{\theta_f}\right) \right\|_2^2, \tag{3.18}$$

where the prediction operator is here a simple RK2 scheme. We discard the stability loss function $\mathcal{L}^s_{\text{stab}}$ and keep all the other hyper-parameters identical to those of the previous section. Coupled

with the autoencoder, this reduction method will be referred to as AE-Flow.

We compare the two resulting method on the nonlinear wave test-case considered in Section 3.4.1.2. The HNN parameters are unchanged. For a fair comparison, the flow neural network $\mathcal{F}_{\theta_f}$ has the same amount of parameters: the hidden layers have $[32, 24, 16, 16]$ units, which results in 1 886 parameters instead of 1 728 for the HNN. We train both models up to reach a loss value of about 1×10^{-5}. The obtained validation loss functions are as follows

$$\mathcal{L}^s_{\text{pred}} = 9.46\times10^{-5}, \quad \mathcal{L}_{\text{AE}} = 9.88\times10^{-5}, \quad \mathcal{L}^s_{\text{Flow}} = 7.92\times10^{-7}.$$

They are of comparable magnitude to that of the AE-HNN method in Table 3.2.

Relative errors are provided in Table 3.5. The errors of the AE-HNN method are about 5 times smaller than those of the AE-Flow method. Figure 3.8 shows the time evolution of the relative error for test case 3 and Figure 3.9 depicts the solutions at different times. We observe that the AE-HNN solution remains close to the reference while the AE-Flow solution drifts from it. Furthermore, we compare in Figure 3.10 the time evolution of the Hamiltonian for the reference solution, the AE-HNN solution, and the AE-Flow solution on test case 3. We observe that the AE-HNN method results in a better preservation of the Hamiltonian than the AE-Flow solution.

	test 1		test 2		test 3	
	error u	error v	error u	error v	error u	error v
AE-HNN	4.35×10^{-3}	1.18×10^{-2}	5.82×10^{-2}	8.66×10^{-2}	1.05×10^{-2}	1.36×10^{-2}
AE-Flow	5.11×10^{-2}	5.56×10^{-2}	1.27×10^{-1}	1.50×10^{-1}	9.99×10^{-2}	1.07×10^{-1}

Table 3.5: (Non linear wave) Relative L^2 errors for the AE-Flow and the AE-HNN

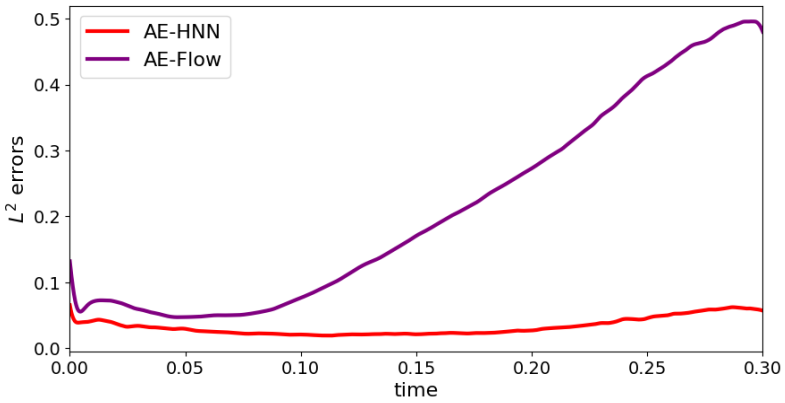


Figure 3.8: (Non linear wave) L^2 errors on test 3 as a function of time for the AE-Flow (purple) and the AE-HNN (red)

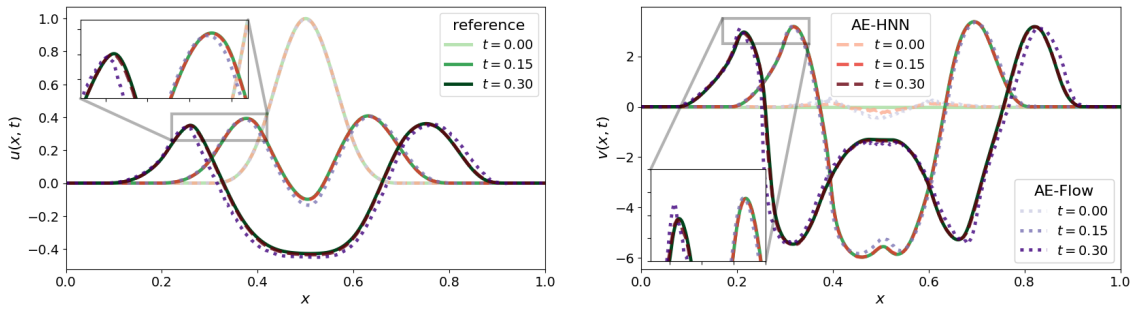


Figure 3.9: (Non linear wave) $(\mathbf{u}, \mathbf{v})$ for different times on test 3, reference solution (green), AE-HNN solution (red) and AE-Flow solution (purple)

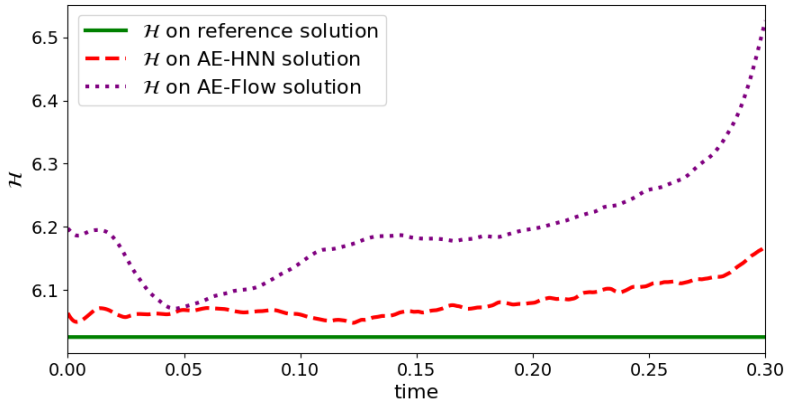


Figure 3.10: (Non linear wave) Time evolution of the Hamiltonian on test 3 for the reference solution (green), the AE-HNN solution (red) and the AE-Flow solution (purple).

3.4.1.4 Gain in computation time

In this section, we compare the actual computation time on an Intel Xeon CPU. To obtain a fair comparison, we use the same implementation of the Störmer-Verlet algorithm for all the methods.

The only difference between the PSD and the AE-HNN methods lies in the gradient computations. For the former, we use the explicit expression of the reduced Hamiltonian as defined in Chapter 1, Sec. 2.3.1 and for the latter, we use the HNN backpropagation algorithm. We use 32 bits floating-point numbers and include the encoding or decoding of the data in the computation time. We consider the nonlinear wave test case with the following parameters: $N = 1024, K = 3, T = 0.4$ and $\Delta t = 1 \times 10^{-4}$.

The reference solution without reduction is computed in $2\,200 \pm 14$ ms. In comparison, the AE-HNN method spends 452 ± 20 ms, which is five times faster. On the contrary, the PSD method spends $2\,382 \pm 12$ ms, which is a bit slower than the reference solution. This results from the evaluation of the gradient of the Hamiltonian in the non-reduced dimension. Hyper-reduction technic like Hamiltonian DEIM would accelerate the computations [8]. However, this could also deteriorate the precision, and one should then consider larger reduced dimensions.

3.4.2 1D shallow water system

We consider the one-dimensional shallow water system under the following formulation:

$$\begin{cases} \partial_t \chi(x,t;\mu) + \partial_x \left((1+\chi(x,t;\mu))\, \partial_x \phi(x,t;\mu) \right) = 0, & \text{in } \Omega_2 \times (0,T], \\ \partial_t \phi(x,t;\mu) + \frac{1}{2} (\partial_x \phi(x,t;\mu))^2 + \chi(x,t;\mu) = 0, & \text{in } \Omega_2 \times (0,T], \\ \chi(x,0;\mu) = \chi_{\text{init}}(x;\mu), & \text{in } \Omega_2, \\ \phi(x,0;\mu) = \phi_{\text{init}}(x;\mu), & \text{in } \Omega_2, \end{cases}$$

where $\chi(x,t;\mu)$ denotes the perturbation of the equilibrium and $\phi(x,t;\mu)$ is the scalar velocity potential and $\Omega_2 = [-1,1]$. Periodic boundary conditions are considered. This system admits a Hamiltonian function given by:

$$\mathcal{H}_{\text{cont}}(\chi,\phi) = \frac{1}{2} \int_{-1}^{1} \left((1+\chi)(\partial_x \phi)^2 + \chi^2 \right) dx.$$

Like the wave equation, we consider a spatial discretization of this model, still using a uniform mesh grid with N nodes $(x_i)_{i\in\{0,\dots,,N-1\}}$ on Ω_2. The approximate solution is denoted $\boldsymbol{\chi} = (\chi_i)_{i\in\{0,\dots,,N-1\}}, \boldsymbol{\phi} = (\phi_i)_{i\in\{0,\dots,,N-1\}}$. We then introduce the discrete Hamiltonian function:

$$\mathcal{H}(\boldsymbol{\chi},\boldsymbol{\phi}) = \frac{\Delta x}{2} \sum_{i=1}^{N} (1+\chi_i) \left(\frac{\phi_{i+1} - \phi_{i-1}}{\Delta x} \right)^2 + \chi_i^2, \tag{3.19}$$

and the resulting discrete shallow water equation:

$$\begin{cases} \dfrac{d}{dt}\boldsymbol{\chi}(t;\mu) = -D_x^2 \boldsymbol{\phi}(t;\mu) - D_x(\boldsymbol{\chi}(t;\mu) \odot D_x \boldsymbol{\phi}(t;\mu)), \\ \dfrac{d}{dt}\boldsymbol{\phi}(t;\mu) = -\dfrac{1}{2}(D_x \boldsymbol{\phi}(t;\mu)) \odot (D_x \boldsymbol{\phi}(t;\mu)) - \boldsymbol{\chi}(t;\mu), \\ \chi_i(0;\mu) = \chi_{\text{init}}(x_i;\mu), \\ \phi_i(0;\mu) = \phi_{\text{init}}(x_i;\mu), \end{cases} \tag{3.20}$$

where $\odot$ denotes the element-wise vector multiplication and D_x the centered second-order finite difference matrix with periodic boundary condition

$$D_x = \frac{1}{2\Delta x} \begin{pmatrix} 0 & 1 & & -1 \\ -1 & & \ddots & \\ & \ddots & & 1 \\ 1 & & -1 & 0 \end{pmatrix}.$$

We note that Hamiltonian (3.19) is not separable. As a consequence, the numerical resolution of (3.20) with the Störmer-Verlet scheme is implicit.Therefore, applying a reduction is all the more attractive.

3.4.2.1 Reduction of the system

The number of discretization points is taken equal to $N = 1024$, the final time equal to $T = 0.5$ and the time step $\Delta t = 2 \times 10^{-4}$. The initial condition is parameterized with 2 parameters $\mu = (c,\sigma) \in [0, 0.2] \times [0.2, 0.05]$

$$\chi_{\text{init}}(x;\mu) = \frac{0.02}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{x-c}{\sigma} \right)^2 \right), \quad \phi_{\text{init}}(x;\mu) = 0.$$

For the training data, we use 20 different solution parameters (c, σ): we take regularly spaced values in the segment $[(0, 0.2), (0.2, 0.05)]$. Let us observe the influence of the parameters on the solutions at initial and final time on Figure 3.11. Nonlinear patterns appear for small values of the standard deviation σ.

The set of hyper-parameters is almost identical to the previous test cases except for the HNN architecture, activation functions and the watch duration, which here is taken equal to 48. They are gathered in Table 3.1.

As in the previous sections, we inspect the validation loss functions and obtain value given in Table 3.2.

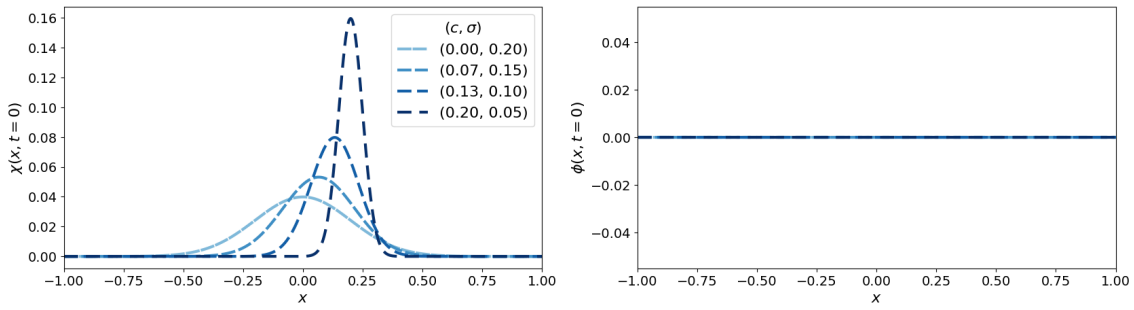
We choose three different sets of parameters to test the AE-HNN method:

$$(c, \sigma) = (0.105, 0.11), \qquad \text{(test 1)}$$
$$(c, \sigma) = (0.195, 0.053), \qquad \text{(test 2)}$$
$$(c, \sigma) = (0.21, 0.045). \qquad \text{(test 3)}$$

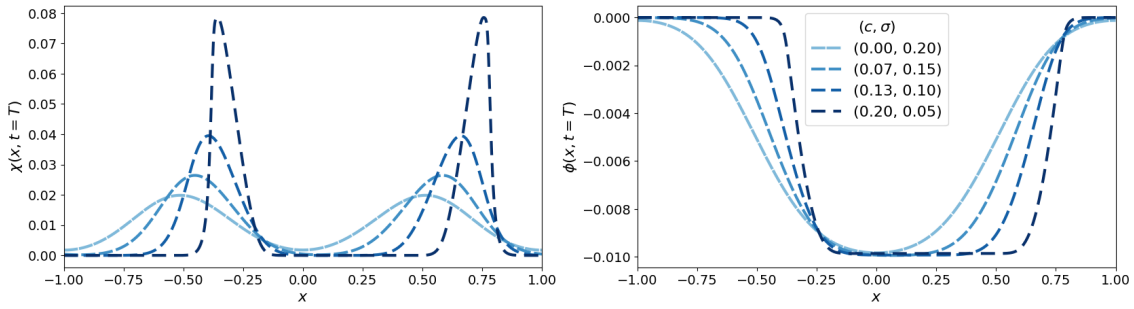
As in the previous test cases, we compute the relative errors of the AE-HNN method with respect to the reference solution (see Table 3.6) and compare them to the results obtained with the PSD and POD methods for different values of K.

The AE-HNN method achieves a mean error of about 3×10^{-2} with a reduced dimension of $K = 4$ only. To achieve a similar performance, the PSD requires a reduced dimension of $K = 24$ and the POD need a value larger than $K = 32$. Table 3.6 also shows that the AE-HNN method has a different behavior with respect to K than the POD and PSD methods. For the latter, increasing K improves accuracy (at the expense of computation time). With the AE-HNN method, increasing K improves the encoder-decoder performance but makes it more difficult to learn the reduced dynamics for the HNN neural network. Therefore, the HNN performance requires a balance between an adequate compression and a low-dimensional reduced model.

Then, we compare the solutions obtained with the AE-HNN method and the PSD in Figure 3.12. For a given reduced dimension $K = 4$, the AE-HNN method solution remains close to the reference as expected (Fig. 3.12a) while the PSD solution oscillates and stays far from the reference solution, even for the initial condition (Fig. 3.12b). When increasing the dimension of the reduced model to $K = 24$ for the PSD method (Fig. 3.12c), we recover similar results as the ones obtained with the AE-HNN.



(a) $t = 0$

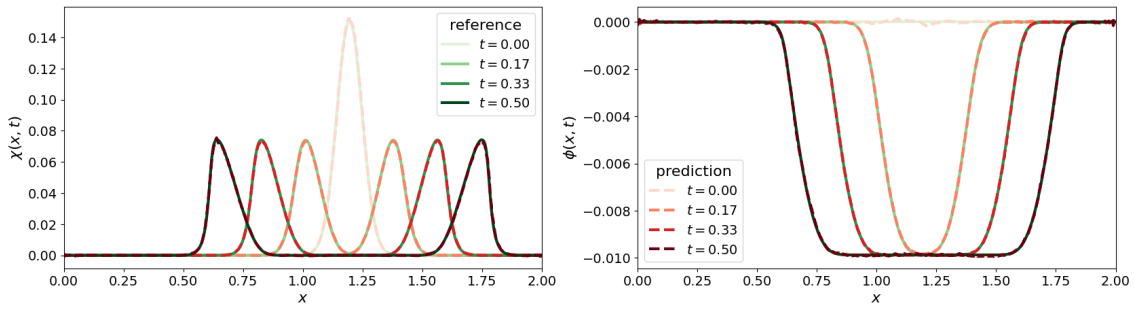


(b) $t = 0.5$

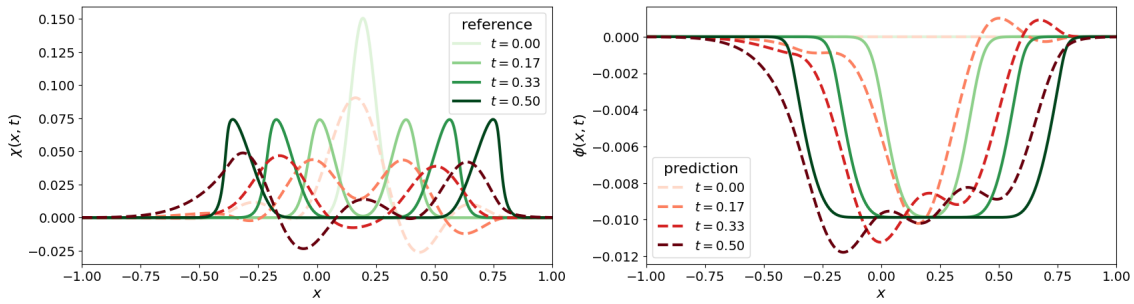
Figure 3.11: (Shallow water 1D) Solutions $(\boldsymbol{\chi}, \boldsymbol{\phi})$ at initial time $t = 0$ and final time $t = 0.5$ for various parameters (c, σ).

		test 1		test 2		test 3	
		error χ	error ϕ	error χ	error ϕ	error χ	error ϕ
AE-HNN	$K = 4$	5.86×10^{-2}	2.89×10^{-2}	1.34×10^{-2}	5.00×10^{-3}	8.12×10^{-2}	2.27×10^{-2}
	$K = 6$	8.46×10^{-2}	5.17×10^{-2}	2.07×10^{-2}	7.77×10^{-3}	9.70×10^{-2}	2.80×10^{-2}
	$K = 8$	6.26×10^{-2}	3.54×10^{-2}	2.21×10^{-2}	1.20×10^{-2}	1.45×10^{-1}	4.99×10^{-2}
PSD	$K = 10$	7.11×10^{-2}	1.71×10^{-2}	1.64×10^{-1}	2.74×10^{-2}	3.08×10^{-1}	5.89×10^{-2}
	$K = 14$	2.67×10^{-2}	5.36×10^{-3}	7.98×10^{-2}	1.00×10^{-2}	2.08×10^{-1}	3.17×10^{-2}
	$K = 24$	1.19×10^{-2}	3.43×10^{-3}	2.69×10^{-2}	4.20×10^{-3}	9.56×10^{-2}	1.06×10^{-2}
POD	$K = 14$	1.13×10^{-1}	4.14×10^{-2}	1.22×10^{-1}	4.10×10^{-2}	6.98×10^{-1}	2.99×10^{-1}
	$K = 24$	4.22×10^{-2}	9.91×10^{-3}	3.31×10^{-2}	7.81×10^{-3}	3.10×10^{-1}	7.96×10^{-2}
	$K = 32$	8.70×10^{-3}	1.67×10^{-3}	1.32×10^{-2}	2.31×10^{-3}	1.35×10^{-1}	2.43×10^{-2}

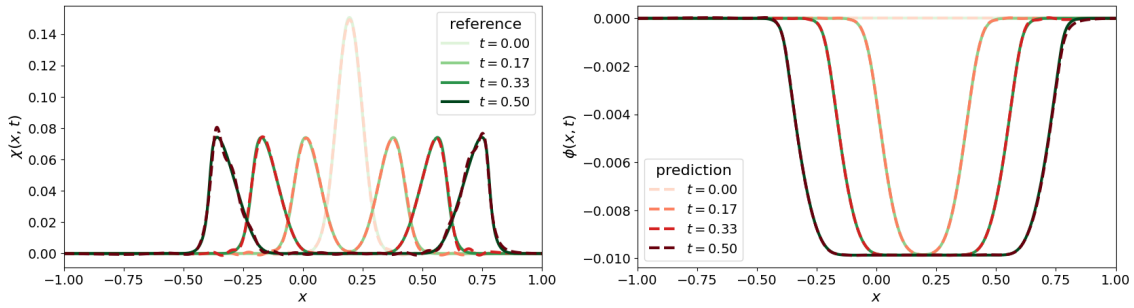
Table 3.6: (Shallow water 1D) Relative L^2 errors for different reduced dimensions K. Blue cells correspond to POD and PSD simulations with lower errors than the corresponding AE-HNN simulation in yellow.



(a) AE-HNN, $K = 4$



(b) PSD, $K = 4$



(c) PSD, $K = 24$

Figure 3.12: (Shallow water 1D) $(\boldsymbol{\chi}, \boldsymbol{\phi})$ as a at different times on test 2 and $K = 8$, reference solution (full lines) and prediction (dashed lines).

3.4.2.2 Comparison with a non Hamiltonian reduction

In this section, we perform a study similar to that performed in Section 3.4.1.3 to show the importance of having a Hamiltonian AE-HNN reduction instead of a classical AE-Flow reduced model. We consider the same test case as in Section 3.4.2.1 with the same hyper-parameters described in Table 3.1. We stop both AE-HNN and AE-Flow training after reaching a validation loss value of 3×10^{-5}. The obtained validation loss functions for the AE-Flow are as follows

$$\mathcal{L}^s_{\text{pred}} = 7.02\times10^{-5}, \quad \mathcal{L}_{\text{AE}} = 6.97\times10^{-5}, \quad \mathcal{L}^s_{\text{Flow}} = 2.85\times10^{-7}.$$

They are of comparable magnitude to that of the AE-HNN method in Table 3.2. Relative errors are provided in Table 3.7 for the different test values. The AE-HNN method is about 4 times more precise than the AE-Flow method. Figure 3.13 shows the time evolution of the L^2 errors for test case 2 and Figure 3.14 depicts the solution at different times. The AE-Flow drifts away from the reference solution while the AE-HNN remains close to it. Finally, Figure 3.15 shows that the AE-HNN method preserves the Hamiltonian more effectively than the AE-Flow method.

	test 1		test 2		test 3	
	error χ	error ϕ	error χ	error ϕ	error χ	error ϕ
AE-HNN	5.86×10^{-2}	2.89×10^{-2}	1.34×10^{-2}	5.00×10^{-3}	8.12×10^{-2}	2.27×10^{-2}
AE-Flow	6.63×10^{2}	3.66×10^{-2}	1.05×10^{-1}	3.61×10^{-2}	1.73×10^{-1}	5.12×10^{-2}

Table 3.7: (Shallow water 1D) Relative L^2 errors for the AE-Flow and the AE-HNN

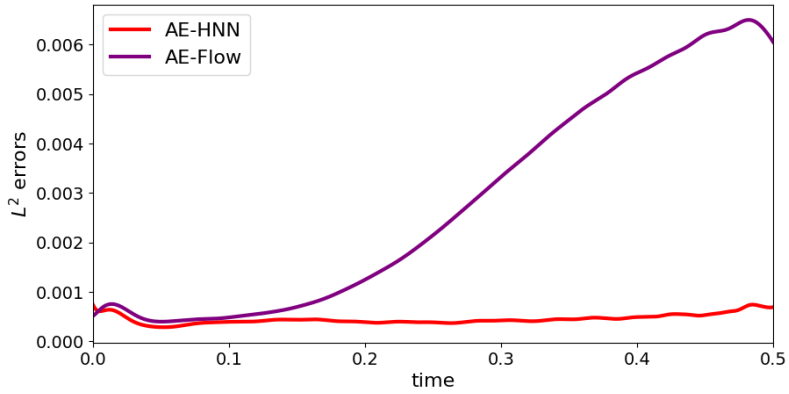


Figure 3.13: (Shallow water 1D) L^2 errors on test 2 as a function of time for the AE-Flow (purple) and the AE-HNN (red)

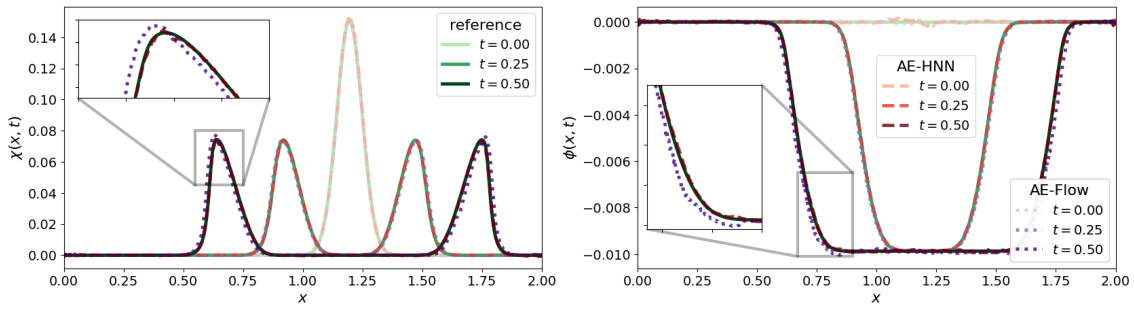


Figure 3.14: (Shallow water 1D) $(\boldsymbol{\chi}, \boldsymbol{\phi})$ for different times on test 2, reference solution (green), AE-HNN solution (red) and AE-Flow solution (purple)

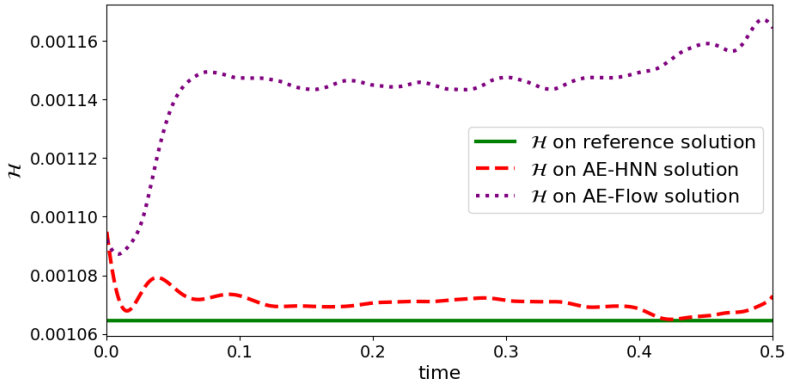


Figure 3.15: (Shallow water 1D) Time evolution of the Hamiltonian on test 2 for the reference solution (green), the AE-HNN solution (red) and the AE-Flow solution (purple).

3.4.3 2D shallow water system

We consider the two-dimensional shallow water system on a square $\Omega_3 = [-1, 1]^2$ under the following formulation:

$$\begin{cases} \partial_t \chi(\mathbf{x}, t; \mu) + \nabla \cdot ((1 + \chi(\mathbf{x}, t; \mu)) \nabla \phi(\mathbf{x}, t; \mu)) = 0, & \text{in } \Omega_3 \times (0, T], \\ \partial_t \phi(\mathbf{x}, t; \mu) + \frac{1}{2} |\nabla \phi(\mathbf{x}, t; \mu)|^2 + \chi(\mathbf{x}, t; \mu) = 0, & \text{in } \Omega_3 \times (0, T], \\ \chi(\mathbf{x}, 0; \mu) = \chi_{\text{init}}(\mathbf{x}; \mu), & \text{in } \Omega_3, \\ \phi(\mathbf{x}, 0; \mu) = \phi_{\text{init}}(\mathbf{x}; \mu), & \text{in } \Omega_3, \end{cases} \tag{3.21}$$

where $\chi(\mathbf{x}, t; \mu)$ denotes the perturbation of the equilibrium and $\phi(\mathbf{x}, t; \mu)$ is the scalar velocity potential. Periodic boundary conditions are considered. This system admits a Hamiltonian function given by:

$$\mathcal{H}_{\text{cont}}(\chi, \phi) = \frac{1}{2} \int_{\Omega_3} \left((1 + \chi) |\nabla \phi|^2 + \chi^2 \right).$$

We consider a spatial discretization of the domain Ω with a regular mesh of $N = M^2$ cells of size $\Delta x = 2/(M - 1)$ in each direction. The discrete Hamiltonian function is

$$\mathcal{H}(\boldsymbol{\chi}, \boldsymbol{\phi}) = \frac{1}{2} \sum_{i,j=0}^{M-1} \left((1 + \chi_{i,j}) \left[\left(\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x} \right)^2 + \left(\frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta y} \right)^2 \right] + \chi_{i,j}^2 \right)$$

with $\chi_{i,j}(t;\mu) \approx \chi(\mathbf{x}_{i,j},t;\mu)$ (resp. $\phi_{i,j}(t;\mu) \approx \phi(\mathbf{x}_{i,j},t;\mu)$), with $\mathbf{x}_{i,j} = (-1,-1) + (i\Delta x, j\Delta x)$. The resulting discrete system reads

$$\begin{cases} \dfrac{d}{dt}\boldsymbol{\chi}(t;\mu) = -D_x\left([1+\boldsymbol{\chi}(t;\mu)]\odot D_x\boldsymbol{\phi}(t;\mu)\right) - D_y\left([1+\boldsymbol{\chi}(t;\mu)]\odot D_y\boldsymbol{\phi}(t;\mu)\right), \\ \dfrac{d}{dt}\boldsymbol{\phi}(t;\mu) = -\frac{1}{2}\left[(D_x\boldsymbol{\phi}(t;\mu))^2 + (D_y\boldsymbol{\phi}(t;\mu))^2\right] - \boldsymbol{\chi}(t;\mu), \\ \chi_m(0;\mu) = \chi_{\text{init}}(\mathbf{x}_m;\mu), \\ \phi_m(0;\mu) = \chi_{\text{init}}(\mathbf{x}_m;\mu), \end{cases}$$

where D_x and D_y are respectively the centered finite difference operators along the x and y axis, and $m = jM + i$.

3.4.3.1 Reduction of the system

We consider $M = 64$ cells per direction, The final time is set to $T = 15$ and the time step to $\Delta t = 1\times 10^{-3}$. In this test case, we choose to use an implicit midpoint scheme [5]. The initial condition, parameterized with two parameters $\mu = (\alpha,\beta) \in [0.2, 0.5]\times[1, 1.7]$, is chosen equal to

$$\chi_{\text{init}}(\mathbf{x};\mu) = \alpha\exp\left(-\beta\,\mathbf{x}^T\mathbf{x}\right), \quad \phi_{\text{init}}(\mathbf{x};\mu) = 0.$$

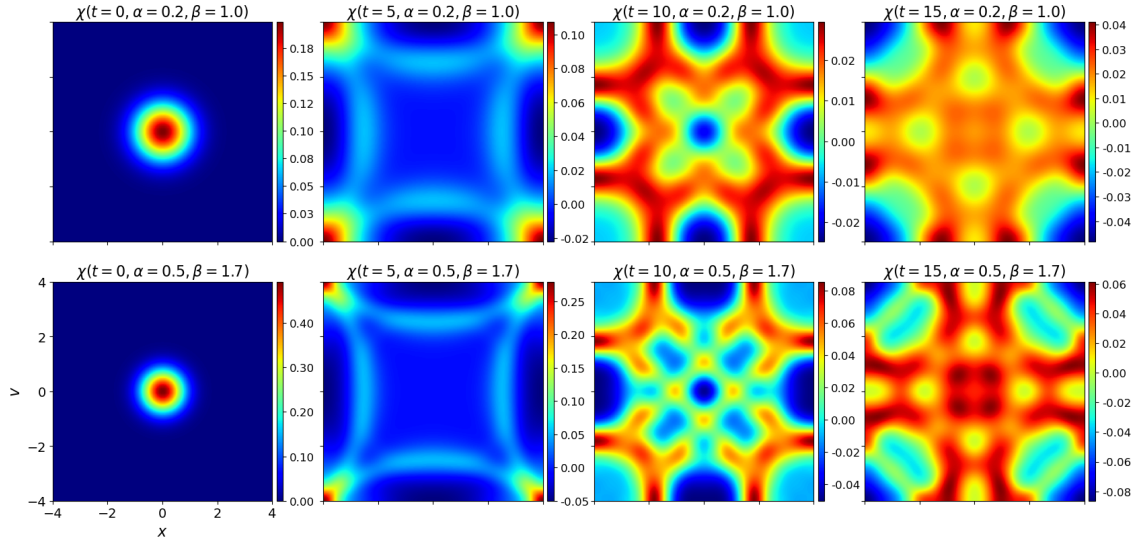
For the training data, we use 20 different couples of parameter (α,β) regularly spaced in the segment $[(0.2,1),(0.5,1.7)]$. Figure 3.15 shows the time evolution for two couples of parameters: $(\alpha,\beta) = (0.2,1)$ and $(\alpha,\beta) = (0.5,1.7)$.

As data are two-dimensional, we use a 2D variant of the convolutional AE: convolutionlayers are two-dimensional, up and down-sampling are extended in 2D. All the neural network hyper-parameters are given in Table 3.1.

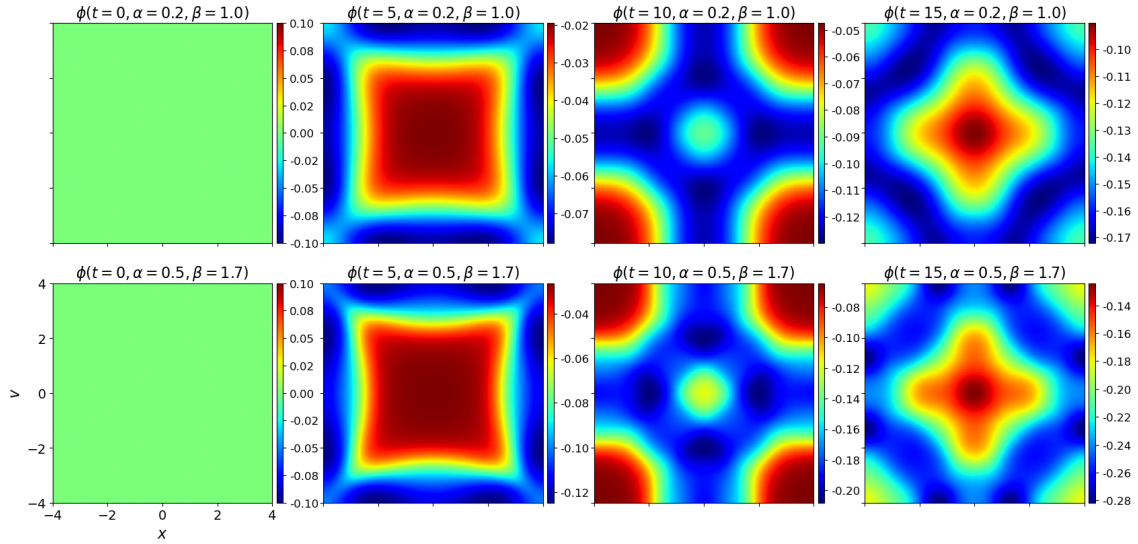
We choose three different sets of parameters to test the AE-HNN method

$$\begin{aligned} &(\alpha,\beta) = (0.35, 1.35), && \text{(test 1)} \\ &(\alpha,\beta) = (0.41, 1.49), && \text{(test 2)} \\ &(\alpha,\beta) = (0.51, 1.72). && \text{(test 3)} \end{aligned}$$

As in the previous test cases, we compute relative errors of the AE-HNN method with respect to the reference solution and compare them to the results obtained with the PSD and POD methods for different values of K in Table 3.8. Regarding the AE-HNN method, a fixed reduced dimension of $K = 4$ if sufficient to obtain a precise reduced model while, for the same reduced dimension, the PSD solution is far from the reference solution as it can be observed in Fig. 3.16. When increasing the reduced dimension to $K = 30$, the PSD produces similar results as the ones obtained with the AE-HNN with only $K = 4$. Regarding the POD reduced model, even a reduced dimension of $K = 35$ does not provide the targeted precision. Indeed, the non-symplecticity of the reduced model produces some instabilities as it can be observed on test 3 in Fig. 3.17.



(a) $\chi(t;\mu)$



(b) $\phi(t;\mu)$

Figure 3.15: (Shallow water 2D) Solutions (χ, ϕ) at different times $t \in \{0, 5, 10, 15\}$ for various parameters $(\alpha, \beta) \in \{(0.2, 1), (0.5, 1.8)\}$.

		test 1		test 2		test 3	
		error χ	error ϕ	error χ	error ϕ	error χ	error ϕ
AE-HNN	$K=4$	4.68×10^{-2}	1.37×10^{-2}	1.73×10^{-2}	5.03×10^{-3}	3.33×10^{-2}	8.92×10^{-3}
	$K=5$	2.32×10^{-2}	6.25×10^{-3}	7.38×10^{-2}	1.62×10^{-2}	1.39×10^{-1}	2.98×10^{-2}
PSD	$K=6$	3.48×10^{-1}	3.96×10^{-2}	3.82×10^{-1}	4.50×10^{-2}	4.33×10^{-1}	5.41×10^{-2}
	$K=20$	5.05×10^{-2}	3.39×10^{-3}	6.52×10^{-2}	4.55×10^{-3}	9.49×10^{-2}	7.13×10^{-3}
	$K=30$	1.37×10^{-2}	9.20×10^{-4}	1.87×10^{-2}	1.19×10^{-3}	3.09×10^{-2}	2.01×10^{-3}
POD	$K=10$	4.51×10^{-1}	3.69×10^{-2}	4.81×10^{-1}	4.47×10^{-2}	5.33×10^{-1}	6.02×10^{-2}
	$K=16$	1.19×10^{-1}	6.31×10^{-3}	4.04×10^{-1}	1.76×10^{-2}	1.01×10^{0}	4.52×10^{-2}
	$K=35$	3.94×10^{-2}	1.83×10^{-3}	5.74×10^{-2}	2.67×10^{-3}	5.05×10^{-1}	2.52×10^{-2}

Table 3.8: (Shallow water 2D) Relative L^2 errors for different reduced dimensions K. Blue cells correspond to POD and PSD simulations with lower errors than the corresponding AE-HNN simulation in yellow.

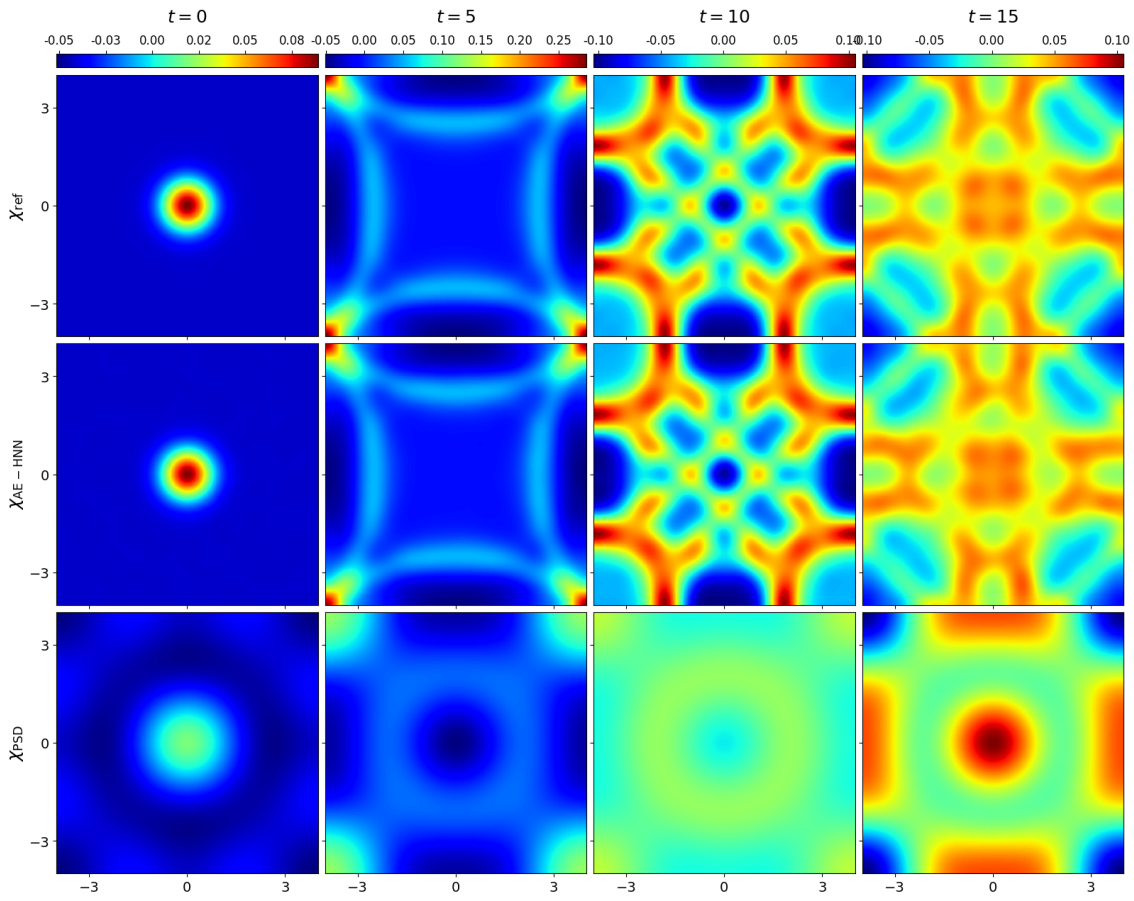


Figure 3.16: (Shallow water 2D) Solutions $\boldsymbol{\chi}(t;\mu)$ at different times $t \in \{0,5,10,15\}$ on test 3 with $K=4$, reference solution (top line), AE-HNN solution (middle line) and PSD solution (bottom line).

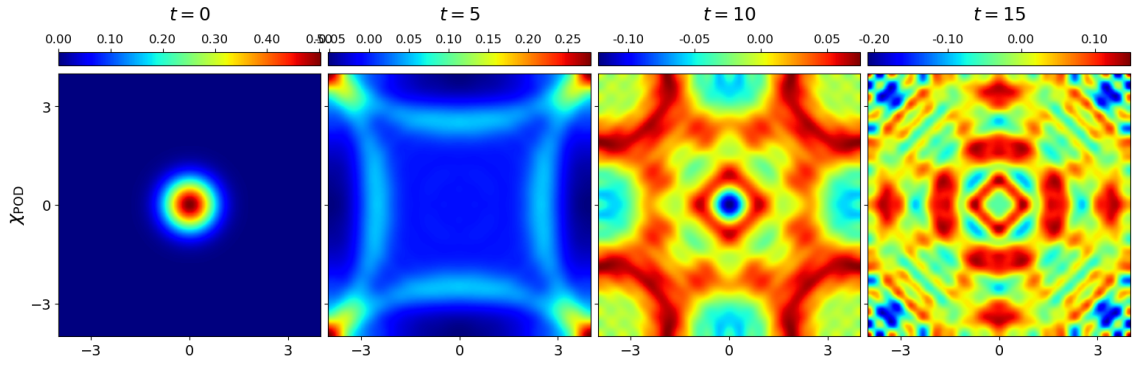


Figure 3.17: (Shallow water 2D) POD solution $\boldsymbol{\chi}(t;\mu)$ at different times $t \in \{0, 5, 10, 15\}$ on test 3 with $K = 35$.

3.4.3.2 Comparison with a symplectic DEIM hyper-reduction

While the AE-HNN would require a smaller reduced dimension than the PSD for a given targeted precision, practical efficiency still need to be compared. As the PSD still requires to come back to the original $2N$ dimension for nonlinear models, we also compare the AE-HNN with the Discrete Empirical Interpolation Method (DEIM) version of the PSD as proposed in [2]: it relies on an interpolation of m given components (among the $2N$ ones). For the sake of completeness, the method is well presented in Chapter 1, Sec. 2.3.3.

First, we test the precision of this DEIM approximation as a function of m. We consider the test-case of the previous subsection, with reduced dimension equal to $K = 30$. We compute the discrete L^2 error on the 3 test cases:

$$\overline{\text{err}}_u^n = \frac{1}{3} \sum_{\mu \in \text{ test}} \left(\sum_{i=1}^{N} \Delta x \, (u_{\text{ref},i}^{\mu,n} - u_{\text{pred},i}^{\mu,n})^2 \right).$$

where $\overline{u}_{\text{ref},i}$ refers to the reference solution computed with the original model and $u_{\text{pred},i}^{\mu,n}$ the one computed with the reduced one. We observe the time evolution of this error for different values of m on Fig 3.18. As expected, the DEIM method deteriorates the reduced model solution with respect to the sole PSD and increasing m decreases the error. In practice, the value of m is set so that the DEIM does not deteriorate the solution too much.

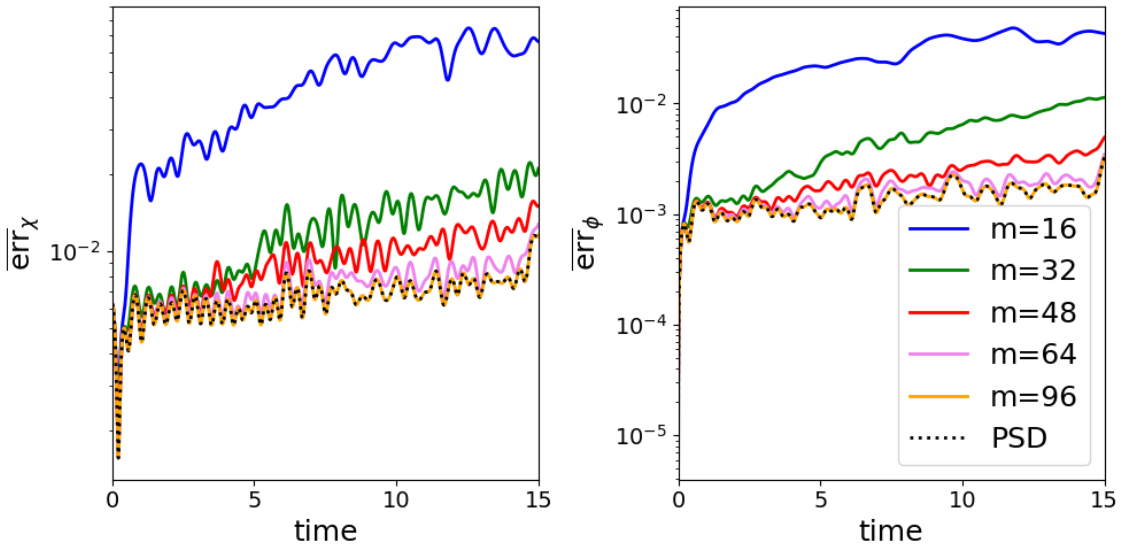


Figure 3.18: (Shallow water 2D) $\overline{\text{err}}$ for solutions $\boldsymbol{\chi}(t;\mu)$ (left) and $\boldsymbol{\phi}(t;\mu)$ (right) for different $m \in \{16, 32, 48, 64, 96\}$ with $K = 30$.

We then compare the computational times of the AE-HNN method with respect to the PSD algorithm with and without symplectic DEIM hyper-reduction. We set $K = 4$ and $m = 32$ so that the DEIM error is smaller than the PSD error. Numerical integration are performed with an implicit midpoint scheme as above-mentioned. The original model and the PSD without DEIM are solved on an Intel Xeon CPU while the AE-HNN method is executed on a NVIDIA Tesla T4 GPU. We also use the Strang splitting method [5] where the linear part is solved explicitly and the nonlinear one is solved with numerical integration except for the AE-HNN method which cannot be split.

The computational time equals 101.0 s for the original model, 107.0 s for the PSD and 57.4 s for the DEIM-PSD. As expected, the PSD is not time efficient in a nonlinear case due to the

decompression-compression of the solution at each time step. An efficient DEIM-PSD implementation allows a satisfactory speed-up of a factor 1.76. Regarding the AE-HNN model, it is solved in 26.5 s, which corresponds to a speed up of 3.81. This could be further improved. Indeed, the neural networks are executed on a GPU but the nonlinear solver used in the implicit part of the scheme is executed on a CPU, which slows down the reduced order model. In practice, a SciPy [33] implementation of the nonlinear solver [34] is used. More than 62% of the nonlinear solver computation time takes place on the CPU.

3.5 Conclusion

We have developed a new Hamiltonian reduction method. It is based on an autoencoder (AE) to transform initial variables into reduced variables and vice versa, and on a Hamiltonian Neural Network (HNN) to learn the Hamiltonian reduced dynamics. Using a set of coupled loss functions, we are able to learn a reduced model (AE-HNN), which has an Hamiltonian structure. It already has better reduction properties than the PSD in linear test-cases, but the gain is much larger in nonlinear test-cases, as expected. Due to its Hamiltonian structure, the reduced model also shows good stability properties. Two-dimensional test-cases show that the AE-HNN method has better computational performance than the PSD-DEIM method.

The question remains of how to improve the quality of the approximation. Indeed, unlike the PSD method, increasing the reduced dimension K does not always provide better results. The quality of the approximation could rather be increased by modifying the architecture of the neural networks (increasing the number of layers and the number of neurons per layer), which implies increasing the number of trainable parameters. However, up to our knowledge, there is unfortunately a lack of theoretical results about a systematic way to improve such approximations. Further studies need to be carried out.

Obviously, the results will have to be extended to partial differential equations in three spatial dimension. As convolution layers are used in the autoencoder, the increase of dimension should not require too large an increase in the size of the neural networks. In consequence, we expect that the computational gain will be even larger for such dynamics. Other extensions would be to consider time-dependent reductions as in [8] and adapt the method for the reduction of large Hamiltonian differential equations that do not have spatial structures [8].

Acknowledgements. This research was funded in part by l'Agence Nationale de la Recherche (ANR), project ANR-21-CE46-0014 (Milk).

References

[1] R. Côte et al. "Hamiltonian reduction using a convolutional auto-encoder coupled to a Hamiltonian neural network". In: *Commun. Comput. Phys.* 37.2 (2025), pp. 315–352. ISSN: 1815-2406,1991-7120. DOI: 10.4208/cicp.OA-2023-0300.

[2] L. Peng and K. Mohseni. "Symplectic model reduction of Hamiltonian systems". In: *SIAM J. Sci. Comput.* 38.1 (2016), A1–A27. ISSN: 1064-8275,1095-7197. DOI: 10.1137/140978922.

[3] J. S. Hesthaven, C. Pagliantini, and G. Rozza. "Reduced basis methods for time-dependent problems". In: *Acta Numer.* 31 (2022), pp. 265–345. ISSN: 0962-4929,1474-0508. DOI: 10.1017/S0962492922000058.

[4] T. Tyranowski and M. Kraus. "Symplectic model reduction methods for the Vlasov equation". In: *Contrib. Plasma Phys.* 63.5-6 (2023), e202200046. ISSN: 0863-1042. DOI: 10.1002/ctpp.202200046.

[5] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration*. Second. Vol. 31. Springer Series in Computational Mathematics. Structure-preserving algorithms for ordinary differential equations. Springer-Verlag, Berlin, 2006, pp. xviii+644. ISBN: 978-3-540-30663-4.

[6] B. M. Afkham and J. S. Hesthaven. "Structure preserving model reduction of parametric Hamiltonian systems". In: *SIAM J. Sci. Comput.* 39.6 (2017), A2616–A2644. ISSN: 1064-8275,1095-7197. DOI: 10.1137/17M1111991.

[7] C. Pagliantini. "Dynamical reduced basis methods for Hamiltonian systems". In: *Numer. Math.* 148.2 (2021), pp. 409–448. ISSN: 0945-3245. DOI: 10.1007/s00211-021-01211-w.

[8] J. S. Hesthaven, C. Pagliantini, and N. Ripamonti. "Adaptive symplectic model order reduction of parametric particle-based Vlasov-Poisson equation". In: *Math. Comp.* 93.347 (2024), pp. 1153–1202. ISSN: 0025-5718,1088-6842. DOI: 10.1090/mcom/3885.

[9] B. Sonday et al. "Manifold learning techniques and model reduction applied to dissipative PDEs". In: *arXiv e-prints* (2010), arXiv–1011. DOI: 10.48550/arXiv.1011.5197.

[10] S. Bhattacharjee and K. Matouš. "A nonlinear manifold-based reduced order model for multiscale analysis of heterogeneous hyperelastic materials". In: *J. Comput. Phys.* 313 (2016), pp. 635–653. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2016.01.040.

[11] J. B. Tenenbaum, V. de Silva, and J. C. Langford. "A Global Geometric Framework for Nonlinear Dimensionality Reduction". In: *Science* 290.5500 (2000), pp. 2319–2323. DOI: 10.1126/science.290.5500.2319.

[12] M. Belkin and P. Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation". In: *Neural computation* 15.6 (2003), pp. 1373–1396. DOI: 10.1162/089976603321780317.

[13] R. R. Coifman and S. Lafon. "Diffusion maps". In: *Appl. Comput. Harmon. Anal.* 21.1 (2006), pp. 5–30. ISSN: 1063-5203,1096-603X. DOI: 10.1016/j.acha.2006.04.006.

[14] Y. Bengio et al. "Learning eigenfunctions links spectral embedding and kernel PCA". In: *Neural computation* 16.10 (2004), pp. 2197–2219. DOI: 10.1162/0899766041732396.

[15] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[16] K. Lee and K. T. Carlberg. "Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders". In: *J. Comput. Phys.* 404 (2020), pp. 108973, 32. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2019.108973.

[17] Y. Kim et al. “A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder”. In: *J. Comput. Phys.* 451 (2022), Paper No. 110841, 29. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2021.110841.

[18] F. Romor, G. Stabile, and G. Rozza. “Non-linear manifold reduced-order models with convolutional autoencoders and reduced over-collocation method”. In: *J. Sci. Comput.* 94.3 (2023), Paper No. 74, 39. ISSN: 0885-7474,1573-7691. DOI: 10.1007/s10915-023-02128-2.

[19] P. Buchfink, S. Glas, and B. Haasdonk. “Symplectic model reduction of Hamiltonian systems on nonlinear manifolds and approximation with weakly symplectic autoencoder”. In: *SIAM J. Sci. Comput.* 45.2 (2023), A289–A311. ISSN: 1064-8275,1095-7197. DOI: 10.1137/21M1466657.

[20] S. Chaturantabut and D. C. Sorensen. “Nonlinear model reduction via discrete empirical interpolation”. In: *SIAM J. Sci. Comput.* 32.5 (2010), pp. 2737–2764. ISSN: 1064-8275,1095-7197. DOI: 10.1137/090766498.

[21] C. Pagliantini and F. Vismara. “Fully adaptive structure-preserving hyper-reduction of parametric Hamiltonian systems”. In: *SIAM J. Sci. Comput.* 47.1 (2025), A124–A152. ISSN: 1064-8275,1095-7197. DOI: 10.1137/23M1601225.

[22] C. Pagliantini and F. Vismara. “Gradient-preserving hyper-reduction of nonlinear dynamical systems via discrete empirical interpolation”. In: *SIAM J. Sci. Comput.* 45.5 (2023), A2725–A2754. ISSN: 1064-8275,1095-7197. DOI: 10.1137/22M1503890.

[23] R. Maulik, B. Lusch, and P. Balaprakash. “Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders”. In: *Phys. Fluids* 33.3 (2021), p. 037106. DOI: 10.1063/5.0039986.

[24] S. Fresca, L. Dede’, and A. Manzoni. “A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs”. In: *J. Sci. Comput.* 87.2 (2021), Paper No. 61, 36. ISSN: 0885-7474,1573-7691. DOI: 10.1007/s10915-021-01462-7.

[25] Q. Wang, N. Ripamonti, and J. S. Hesthaven. “Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori-Zwanzig formalism”. In: *J. Comput. Phys.* 410 (2020), pp. 109402, 32. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2020.109402.

[26] S. Greydanus, M. Dzamba, and J. Yosinski. “Hamiltonian neural networks”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates Inc., 2019. DOI: 10.48550/arXiv.1906.01563.

[27] H. Yu et al. “OnsagerNet: Learning stable and interpretable dynamics using a generalized Onsager principle”. In: *Phys. Rev. Fluids* 6.11 (2021). ISSN: 2469-990X. DOI: 10.1103/physrevfluids.6.114402.

[28] Z. Zhang, Y. Shin, and G. E. Karniadakis. “GFINNs: GENERIC formalism informed neural networks for deterministic and stochastic dynamical systems”. In: *Philos. Trans. Roy. Soc. A* 380.2229 (2022), Paper No. 20210207, 21. ISSN: 1364-503X,1471-2962. DOI: 10.1098/rsta.2021.0207.

[29] X. Chen et al. “Constructing custom thermodynamics using deep learning”. In: *Nat. Comput. Sci.* 4.1 (2024), pp. 66–85. ISSN: 2662-8457. DOI: 10.1038/s43588-023-00581-5.

[30] M. Flaschel, S. Kumar, and L. De Lorenzis. “Automated discovery of generalized standard material models with EUCLID”. In: *Comput. Methods Appl. Mech. Engrg.* 405 (2023), Paper No. 115867, 26. ISSN: 0045-7825,1879-2138. DOI: 10.1016/j.cma.2022.115867.

[31] N. Thuerey et al. “Physics-based deep learning”. In: *arXiv preprint arXiv:2109.05237* (2021). DOI: 10.48550/arXiv.2109.05237.

[32] D. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv preprint arXiv:1412.6980* (2014). DOI: 10.48550/arXiv.1412.6980.

[33] P. Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[34] W. La Cruz, J. M. Martínez, and M. Raydan. “Spectral residual method without gradient information for solving large-scale nonlinear systems of equations”. In: *Math. Comp.* 75.255 (2006), pp. 1429–1448. ISSN: 0025-5718,1088-6842. DOI: 10.1090/S0025-5718-06-01840-0.

Chapter 4

Reduced Particle in Cell method for the Vlasov-Poisson system using autoencoder and Hamiltonian neural networks

This chapter is based on a preprint currently under review *(submitted for publication)*.

E. Franck, L. Navoret, V. Vigon, R. Côte, and G. Steimer. "Reduced Particle in Cell method for the Vlasov-Poisson system using auto-encoder and Hamiltonian neural". working paper or preprint. June 2025. URL: https://hal.science/hal-05116555

Abstract

Hamiltonian particle-based simulations of plasma dynamics are inherently computationally intensive, primarily due to the large number of particles required to obtain accurate solutions. This challenge becomes even more acute in many-query contexts, where numerous simulations must be conducted across a range of time and parameter values. Consequently, it is essential to construct reduced order models from such discretizations to significantly lower computational costs while ensuring validity across the specified time and parameter domains. Preserving the Hamiltonian structure in these reduced models is also crucial, as it helps maintain long-term stability.

In this paper, we introduce a nonlinear, non-intrusive, data-driven model order reduction method for the 1D-1V Vlasov–Poisson system, discretized using a Hamiltonian Particle-In-Cell scheme. Our approach relies on a two-step projection framework: an initial linear projection based on the Proper Symplectic Decomposition, followed by a nonlinear projection learned via an autoencoder neural network. The reduced dynamics are then modeled using a Hamiltonian neural network. The offline phase of the method is split into two stages: first, constructing the linear projection using full-order model snapshots; second, jointly training the autoencoder and the Hamiltonian neural network to simultaneously learn the encoder-decoder mappings and the reduced dynamics. We validate the proposed method on several benchmarks, including Landau damping and two-stream instability. The results show that our method has better reduction properties than standard linear Hamiltonian reduction methods.

Chapter's contents

4.1 Introduction . . . 88
4.2 Particle discretization of the Vlasov-Poisson equation . . . 91
4.2.1 The Vlasov-Poisson equation . . . 91
4.2.2 Hamiltonian particle-based discretization . . . 91
4.2.3 Time discretization and initialization . . . 93
4.3 A Hamiltonian reduction with Proper Symplectic Decompostion pre-reduction . . . 94
4.3.1 PSD-AE-HNN reduction method . . . 95
4.3.2 PSD reduction . . . 97
4.3.3 AE-HNN reduction . . . 98
4.3.4 Hyperparameters tuning . . . 99
4.4 Numerical results . . . 101
4.4.1 Linear Landau damping . . . 102
4.4.2 Nonlinear Landau damping . . . 105
4.4.3 Two-stream instability . . . 111
4.4.4 Computation gain . . . 114
4.5 Conclusion . . . 118
References . . . 119

4.1 Introduction

Plasma are gases made of charged particles interacting through long-range Coulomb interactions. A standard kinetic approach for characterizing collisionless plasma dynamics is based on the Vlasov–Maxwell equations, which describe the time evolution of the particle distribution functions in position-velocity phase space and the dynamics of the self-consistent electromagnetic fields. In this work, we focus on the electrostatic limit of these equations, namely the Vlasov–Poisson system, where particle interactions are driven by a self-consistent electric field satisfying the Poisson equation.

Simulating the Vlasov–Poisson system numerically presents significant challenges, and a wide range of particle-based methods have been developed to address them. These methods represent the charged particles distribution using a large set of macro particles, whose trajectories are evolved according to the characteristics of the kinetic equation. To compute the self-induced electric field, the computational domain is discretized into a mesh on which the Poisson equation is solved. The particle distribution is projected onto this mesh to get the charge density, the electric field is computed there, and then interpolated back to the particle positions. The particles are subsequently moved under the influence of the resulting Lorentz force. This approach is known as the Particle-In-Cell (PIC) method [2, 3].

Over time, PIC methods have been refined to preserve important physical invariants, such as total energy (see [4, 5] and references therein). Notably, the Vlasov–Poisson system admits a Hamiltonian formulation [6], which ensures conservation of total energy and the system's underlying symplectic structure. Preserving this Hamiltonian structure in the discretized PIC framework is essential for maintaining long-term numerical stability. As a consequence, several Hamiltonian PIC methods have been developed, including the Hamiltonian PIC scheme [7], as well as canonical and non-canonical symplectic PIC methods [8, 9], and the Geometric Electromagnetic PIC (GEMPIC) method [10].

Given the nonlinear dynamics and multiscale phenomena of the Vlasov–Poisson system, along with the need to employ a very large number of particles to achieve accurate convergence to the solution [11], PIC simulations present a significant numerical challenge. This makes the use of Hamiltonian model order reduction techniques particularly compelling. In real time or many query contexts—such as control processes, optimization, or uncertainty quantification—reduced order models (ROMs) can be crucial. Starting from a particle-based discretization of the Vlasov–Poisson system, referred to as the full order model (FOM), model reduction seeks to construct a smaller dynamical system that provides accurate approximations over a specified range of times and parameters, while substantially lowering computational cost. Crucially, preserving the Hamiltonian structure in the reduced model contributes to its long-term robustness and stability [12].

Over the recent years, considerable efforts have been devoted to constructing reduced models for the Vlasov–Poisson dynamics. These surrogate models aim to reduce the computational cost associated with plasma simulations. By allowing for a small approximation error, reduced models enable significantly faster solution evaluations. Broadly speaking, three main families of model order reduction techniques have been applied to the Vlasov–Poisson system. These approaches are well reviewed in [13] within a more general framework.

The first family comprises projection-based model order reduction methods [14, 15, 16]. These approaches assume that the solution manifold lies close to a low-dimensional subspace, which is approximated using a collection of solution snapshots obtained thanks to Proper Orthogonal Decomposition (POD) or greedy approach. The reduced model is then derived using a Galerkin projection onto this subspace. However, projection alone often does not suffice to reduce computational cost: to compute the reduced system's vector field, the reduced state must typically be lifted back to the full physical space. To mitigate this costly round trip, various strategies have been introduced. For example, the Discrete Empirical Interpolation Method (DEIM) [17] selects a small set of spatial points at which to evaluate the nonlinear terms, while Dynamic Mode Decomposition (DMD) [18, 19] extracts relevant modes from the data and models their evolution with a simplified linear system. The second family concerns sparse approximation methods, in which solutions are approximated by selecting a small number of basis functions, based on prior knowledge about the structure of the solution. This has been explored for the Vlasov-Poisson dynamics in [20], where the dynamics is computed on a sparse grid using a semi-Lagrangian solver. The third family is made of low-rank approximation methods, where the solutions are expressed as a sum of low rank tensors. This has been applied for plasma dynamics in [21, 22, 23]. For instance, [22] proposes a continuous low-rank representation of the distribution function, followed by discretization using a conservative dynamical low-rank scheme that preserves key physical quantities such as mass, momentum, and energy.

While these techniques have proven effective for continuous problems in Eulerian or semi-Lagrangian frameworks, the model reduction of particle-based discretizations of the Vlasov–Poisson system poses a distinct challenge. In parallel with the development of these methods, several studies have highlighted the potential of machine learning—particularly neural networks—to assist with or even automate aspects of model order reduction. Examples include manifold learning techniques [24, 25], as well as data-driven reduced order models for PDEs derived from mesh-based discretizations [26, 27].

In addition, preserving the Hamiltonian structure in the reduced model is a supplementary challenge. Using reduction techniques which do not preserve structure, such as POD, often leads to numerically unstable models: their dynamics can diverge significantly from the underlying physical behavior. Fortunately, several reduction techniques can be adapted to retain structural properties within the reduced model. For example, the DEIM method can be modified to preserve the first moments of an operator, as shown in [28]. Specifically for Hamiltonian systems, struc-

ture preservation can be ensured through the Proper Symplectic Decomposition (PSD) [29], a symplectic counterpart to POD, in which the projection onto the reduced space is constrained to be symplectic. For the PIC model that motivates our work, [16] introduces a dynamic, projection-based model order reduction framework. The projection evolves in time and, with additional constraints, can be made symplectic—thus ensuring that the reduced model remains Hamiltonian. To further enhance computational efficiency, their approach also integrates a DMD-DEIM method to reduce the number of particles effectively. Machine learning techniques have also been extended to account for Hamiltonian structures. In [30], for instance, the authors replace the linear PSD mapping with a neural network that is weakly constrained to be symplectic.

In this paper, we focus on the 1D-1V Vlasov–Poisson system discretized using a Hamiltonian PIC scheme. This discretization yields a high-dimensional ODE with a Hamiltonian structure. However, relying solely on a PSD to construct a reduced model is insufficient to capture small-scale dynamics and nonlinear behaviors with a small reduced dimension. As a result, such an approach would offer limited computational speedup.

To address this issue, we present a strategy inspired by [31], where the authors perform an initial projection onto an intermediate subspace using a Proper Orthogonal Decomposition (POD), followed by the construction of a reduced model through deep learning techniques. Our approach, which we refer to as the PSD-AE-HNN method, similarly employs a two-step projection. It combines PSD with the AE-HNN method introduced in [32].

Starting from the full state variables of the PIC model, we first apply a PSD-based projection onto a symplectic subspace of intermediate dimension. Due to the symplectic nature of the PSD, the intermediate variables follows a Hamiltonian dynamic. Next, we perform a second, nonlinear projection using an autoencoder (AE) neural network [33] to map the system onto a lower dimensional space. While this second mapping is not explicitly symplectic, we enforce Hamiltonian structure in the reduced dynamics by training a Hamiltonian Neural Network (HNN) [34] and incorporating tailored loss functions to constrain the training process.

The motivation behind this two-step mapping lies in the nature of the PIC discretization: particles are neither ordered nor regularly spaced, precluding the use of convolutional neural networks. Furthermore, a large number of particles (e.g. 10^5 in 1D–1V) are typically required to achieve accurate convergence, making the direct use of dense neural networks impractical. The PSD thus acts as a symplectic preconditioner, enabling the AE-HNN method to learn dynamics from a Hamiltonian intermediate representation of reasonable size (e.g. 10^2).

The structure of this paper is as follows: In the first section, we recall the Vlasov-Poisson equation and its Hamiltonian PIC discretization. In the second section, we present our model order reduction technique, referred to as the PSD-AE-HNN method. In the third section, we apply this method to several classic numerical test cases, including linear and nonlinear Landau damping and the two-stream instability. We then provide a comparison of computational times before concluding.

4.2 Particle discretization of the Vlasov-Poisson equation

In this section, we present the full order model (FOM) of interest. It is a particle-based discretization of the Vlasov-Poisson equation which possesses a Hamiltonian structure.

4.2.1 The Vlasov-Poisson equation

We consider a parametric 1D-1V Vlasov-Poisson equation, that gives the dynamics of the particle distribution function $f(t, x, v; \mu)$ which depends on time $t \in [0, T]$ with $T > 0$, position $x \in \Omega_x$, a periodic domain of size $|\Omega_x|$, velocity $v \in \Omega_v \subset \mathbb{R}$, and parameters $\mu \in \Gamma \subset \mathbb{R}^d$ with $d > 0$. The equation reads

$$\begin{cases} \partial_t f(t,x,v;\mu) + v\,\partial_x f(t,x,v;\mu) + \dfrac{q}{m} E(t,x;\mu)\,\partial_v f(t,x,v;\mu) = 0, & \text{in } [0,T]\times\Omega_x\times\Omega_v\times\Gamma, \\ \partial_x E(t,x;\mu) = q\displaystyle\int_{\Omega_v} f(t,x,v;\mu)\,dv - \rho_0, & \text{in } [0,T]\times\Omega_x\times\Gamma, \\ f(0,x,v;\mu) = f_{\text{init}}(x,v;\mu), & \text{in } \Omega_x\times\Omega_v\times\Gamma, \end{cases} \tag{4.1}$$

where $E(t, x; \mu) \in \mathbb{R}$ is the electric field, q is the individual charge of the particles, m their individual mass and $f_{\text{init}}(x, v; \mu) \in \mathbb{R}$ is a given initial condition. Defining the charge density $\rho(t, x; \mu) = q \int_{\mathbb{R}} f(t, x, v; \mu)\, dv$ and the electric potential $\phi(t, x; \mu) \in \mathbb{R}$ such that $E(t, x; \mu) = -\partial_x \phi(t, x; \mu)$, the Poisson equation rewrites

$$-\partial_{xx}\phi(t,x;\mu) = \rho(t,x;\mu) - \rho_0, \quad \text{in } [0,T]\times\Omega_x\times\Gamma. \tag{4.2}$$

The variable ρ_0 corresponds to the average global charge density initially and remains constant over time:

$$\rho_0 = \frac{1}{|\Omega_x|}\int_{\Omega_x} \rho(t,x;\mu)\,dx. \tag{4.3}$$

This quantity is subtracted from the charge density ρ in the right-hand side of the Poisson equation to ensure that the system is well posed with periodic boundary conditions.

The Vlasov-Poisson equation given in Eq. (4.1) admits a Hamiltonian formulation with a Lie-Poisson bracket [35], and a Hamiltonian which corresponds to the sum of the kinetic and potential energies of the system

$$H(f;\mu) = \frac{m}{2}\int_{\Omega_x\times\Omega_v} v^2 f(t,x,v;\mu)\,dxdv + \frac{1}{2}\int_{\Omega_x} |E(t,x;\mu)|^2\,dx.$$

4.2.2 Hamiltonian particle-based discretization

We consider a Particle-In-Cell (PIC) discretization of Eq. (4.1) that preserves the Hamiltonian structure of the equations. Namely, we use a particularization of the GEMPIC algorithm [10], as considered in [16]. The distribution function f is approximated with a set of $N \in \mathbb{N}$ macro-particles and the electric field is obtained by solving the Poisson equation with a finite element discretization resulting in an approximate electric field $E_h(t, x, \mu)$. More precisely, we approximate f with a sum of Dirac delta distributions, located at position $(x_k(t;\mu), v_k(t;\mu))$ in phase space:

$$f_N(t,x,v;\mu) = \sum_{k=1}^{N} \omega\,\delta\left(x - x_k(t;\mu)\right)\delta\left(v - v_k(t;\mu)\right),$$

where ω is the weight of each particle assumed to be identical for all particles and set equals to $|\Omega_x|\rho_0/(qN)$ to ensure the charge density to be normalized (Eq. (4.3)). To satisfy the Vlasov equation of Eq. (4.1), the dynamics of N particles have to satisfy the following system of differential equations:

$$\begin{cases} \dfrac{d}{dt}\mathbf{x}(t;\mu) = \mathbf{v}(t;\mu), & \text{in } [0,T], \\ \dfrac{d}{dt}\mathbf{v}(t;\mu) = \dfrac{q}{m}\mathbf{E}_h(t,\mathbf{x}(t;\mu);\mu), & \text{in } [0,T], \\ \mathbf{x}(0;\mu) = \mathbf{x}_{\text{init}}(\mu), \\ \mathbf{v}(0;\mu) = \mathbf{v}_{\text{init}}(\mu), \end{cases} \tag{4.4}$$

where $\mathbf{x}(t;\mu) = (x_k(t;\mu)), \mathbf{v}(t;\mu) = (v_k(t;\mu)) \in \mathbb{R}^N$ denotes the vectors of positions and velocities and $\mathbf{E}_h(t,\mathbf{x};\mu) = (E_h(t,x_k;\mu)) \in \mathbb{R}^N$ the approximate electric field evaluated at each particle position. To obtain this approximate electric field, an approximate charge density $\rho_h(t,x;\mu)$ is computed on the finite element mesh from the particles distribution (deposition step), the Poisson equation is solved and then the electric field is evaluated at particle positions (interpolation step). We thus need deposition and interpolation steps such that the resulting system is still Hamiltonian.

In detail, we introduce a uniform grid of Ω_x, denoted $X_h = \{ih, i \in \{1,\cdots,n_x\}\}$, where h is the cell length. We consider a H^1-conforming finite element discretization of the Poisson Eq. (4.2) in the space $\mathcal{P}_1\Lambda^0(\Omega_x)$ of piecewise linear functions. As in [16], $\left(\lambda_i^0(x)\right)_{i\in\{1,\cdots,n_x\}}$ denotes the basis, which satisfies $\lambda_i^0(jh) = \delta_{i,j}$ with $\delta_{i,j}$ the Kronecker delta. Then, we define the particle-to-grid mapping $\Lambda^0(\mathbf{x}) \in \mathcal{M}_{N,n_x}(\mathbb{R})$:

$$\left(\Lambda^0(\mathbf{x})\right)_{k,i} = \lambda_i^0(x_k), \qquad k \in \{1,\cdots,N\}, i \in \{1,\cdots,n_x\},$$

and the matrix of the Poisson problem $L \in \mathcal{M}_{n_x,n_x}(\mathbb{R})$ by

$$L_{i,j} = \langle d_x\lambda_i^0, d_x\lambda_j^0\rangle_{L^2(\Omega_x)}, \qquad i,j \in \{1,\cdots,n_x\},$$

where d_x is the derivative with respect to x and $\langle\cdot,\cdot\rangle_{L^2(\Omega_x)}$ is the standard $L^2(\Omega_x)$ scalar product, which in our case equals the standard one-dimensional discrete Laplacian matrix (up to factor $1/h$):

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

With these notations, the computation of the approximate electric field can be written as follows. From the particles positions, we compute a discrete charge density

$$\text{(deposition step)} \qquad \boldsymbol{\rho}_h = q\omega\,\Lambda^0(\mathbf{x})^T\mathbb{1}_N = \left(q\omega\sum_{k=1}^{N}\lambda_i^0(x_k)\right)_i \in \mathbb{R}^{n_x}.$$

where $\mathbb{1}_N \in \mathbb{R}^N$ is a vector of ones. Then, the approximate potential is computed by solving the discrete Poisson equation

$$-L\boldsymbol{\phi}_h = \boldsymbol{\rho}_h - h\rho_0\mathbb{1}_{n_x},$$

with $\boldsymbol{\phi}_h \in \mathbb{R}^{n_x}$. Finally, the discrete electric field is defined by $E_h(t,x;\mu) = -\sum_{i=1}^{n_x} d_x\lambda_i^0(x)(\boldsymbol{\phi}_h)_i$, and can be evaluated at particles positions:

$$\text{(interpolation step)} \qquad \mathbf{E}_h = -\nabla\Lambda^0(\mathbf{x})\,\boldsymbol{\phi}_h = \left(-\sum_{i=1}^{n_x} d_x\lambda_i^0(x_k)(\boldsymbol{\phi}_h)_i\right)_k \in \mathbb{R}^N,$$

with $\nabla\Lambda^0(\mathbf{x}) = (d_x\lambda_i^0(x_k))_{k,i} \in \mathcal{M}_{N,n_x}(\mathbb{R})$. We note that we recover the deposition and interpolation steps of the standard PIC method [2].

The resulting system has a Hamiltonian structure. Indeed, introducing the discrete Hamiltonian function

$$\mathscr{H}\left(\mathbf{x}(t;\mu),\mathbf{v}(t;\mu)\right) = \frac{1}{2}\|\mathbf{v}(t;\mu)\|^2 + \mathcal{U}\left(\mathbf{x}(t;\mu)\right), \tag{4.5}$$

with

$$\mathcal{U}\left(\mathbf{x}(t;\mu)\right) = \frac{1}{2m\omega}\left(q\omega\Lambda^0\left(\mathbf{x}(t;\mu)\right)^T\mathbb{1}_N - h\rho_0\mathbb{1}_{n_x}\right)^T L^{-1}\left(q\omega\Lambda^0\left(\mathbf{x}(t;\mu)\right)^T\mathbb{1}_N - h\rho_0\mathbb{1}_{n_x}\right), \tag{4.6}$$

and the variable $\mathbf{u}(t;\mu) = (\mathbf{x}(t;\mu),\mathbf{v}(t;\mu))$, the full order dynamics Eq. (4.4) rewrite as a Hamiltonian system:

$$\begin{cases} \dfrac{d}{dt}\mathbf{u}(t;\mu) = J_{2N}\nabla_{\mathbf{u}}\mathscr{H}\left(\mathbf{u}(t;\mu)\right), \quad \text{in } [0,T] \\ \mathbf{u}(0;\mu) = \mathbf{u}_{\text{init}}(\mu) \end{cases} \tag{4.7}$$

with the Hamiltonian gradient given by:

$$\nabla_{\mathbf{u}}\mathscr{H}\left(\mathbf{u}(t;\mu)\right) = \begin{pmatrix}\nabla_{\mathbf{x}}\mathcal{U}(\mathbf{x}(t;\mu))\\ \mathbf{v}(t;\mu)\end{pmatrix} = \begin{pmatrix}\frac{q}{m}\nabla\Lambda^0\left(\mathbf{x}(t;\mu)\right)L^{-1}\left(q\omega\Lambda^0\left(\mathbf{x}(t;\mu)\right)^T\mathbb{1}_N - h\rho_0\mathbb{1}_{n_x}\right)\\ \mathbf{v}(t;\mu)\end{pmatrix} \tag{4.8}$$

and J_{2N} referring to the canonical symplectic matrix

$$J_{2N} = \begin{pmatrix} 0_N & I_N \\ -I_N & 0_N \end{pmatrix},$$

with I_N and 0_N, respectively, the identity and null matrices of size N. Equations (4.7)-(4.8) will be referred to as the Hamiltonian FOM system.

4.2.3 Time discretization and initialization

We recall that the flow $\phi_t : \mathbb{R}^{2N} \to \mathbb{R}^{2N}$ of a differential equation is a mapping from the initial state to the state at any time t

$$\phi_t\left(\mathbf{u}_{\text{init}}(\mu)\right) := \mathbf{u}(t;\mu).$$

A key property of Hamiltonian systems, as defined in Eq. (4.7), is that the associated flow is symplectic, meaning that it satisfies the relation

$$\left(\nabla_{\mathbf{u}}\phi_t\left(\mathbf{u}_{\text{init}}(\mu)\right)\right)^T J_{2N}\left(\nabla_{\mathbf{u}}\phi_t\left(\mathbf{u}_{\text{init}}(\mu)\right)\right) = J_{2N}, \qquad \forall t\in[0,T], \mu\in\Gamma.$$

One consequence is that the Hamiltonian $\mathscr{H}$ is preserved along the flow

$$\mathscr{H}\left(\mathbf{u}(t;\mu)\right) = \mathscr{H}\left(\mathbf{u}_{\text{init}}(\mu)\right), \qquad \forall t\in(0,T], \mu\in\Gamma.$$

This is particularly important when considering physical systems. To preserve the symplectic structure at the discrete level, we consider the Störmer-Verlet scheme, which is a symplectic time integrator [36]. It is second order accurate and is explicit in the case of a separable Hamiltonian, which is the case in the problem under consideration. Indeed, the Hamiltonian (4.5) writes as the sum of a discrete kinetic energy, depending only on $\mathbf{v}$, and a discrete potential energy, depending only on $\mathbf{x}$:

$$\mathscr{H}(\mathbf{u}) = \mathscr{H}^{\text{kin}}(\mathbf{v}) + \mathscr{H}^{\text{pot}}(\mathbf{x}).$$

with

$$\mathscr{H}^{\text{kin}}(\mathbf{v}) = \frac{1}{2}\|\mathbf{v}\|^2, \quad \mathscr{H}^{\text{pot}}(\mathbf{x}) = \mathcal{U}(\mathbf{x}).$$

Introducing a time step Δt, and denoting $\mathbf{u}^n = (\mathbf{x}^n, \mathbf{v}^n)$ the numerical solution at time $t^n = n\Delta t$, the Störmer-Verlet scheme reads

$$\begin{aligned} \mathbf{v}^{n+\frac{1}{2}} &= \mathbf{v}^n - \frac{\Delta t}{2}\nabla_{\mathbf{x}}\mathcal{U}(\mathbf{x}^n), \\ \mathbf{x}^{n+1} &= \mathbf{x}^n + \Delta t\, \mathbf{v}^{n+\frac{1}{2}}, \\ \mathbf{v}^{n+1} &= \mathbf{v}^{n+1} - \frac{\Delta t}{2}\nabla_{\mathbf{x}}\mathcal{U}(\mathbf{x}^{n+1}), \end{aligned} \tag{4.9}$$

where the expression of $\nabla_{\mathbf{x}}\mathcal{U}$ is given in Eq. (4.8).

The numerical simulation starts by initializing the particle positions, $\mathbf{x}^0 = \mathbf{x}_{\text{init}}(\mu)$, and velocities, $\mathbf{v}^0 = \mathbf{v}_{\text{init}}(\mu)$, based on the initial distribution $f_{\text{init}}(x, v; \mu)$. A common approach is to use inverse sampling, which may require to empirical estimate the inverse cumulative distribution function. This method depends on a random number generator, which introduces noise that can degrade the accuracy of the solution [37, 11]. To avoid this issue, we instead use a non-random number generator based on a Hammersley sequence [38], which effectively reduces simulation noise. This method is known as a quiet start.

4.3 A Hamiltonian reduction with Proper Symplectic Decompostion prereduction

Taking into consideration that the number of particles N is generally large, the numerical resolution of the Hamiltonian FOM, given in Eq. (4.7), requires significant computational resources and time. Hence, obtaining solutions for various parameters $\mu \in \Gamma$ and times t can become computationally intractable. As a consequence, we aim at building a reduced order model, much smaller in size, that captures the main dynamics for various times t and parameters $\mu \in \Gamma$ and that is more affordable to compute. This reduced order model must also have a Hamiltonian structure.

First, we define the solution manifold

$$\mathcal{M} = \{\mathbf{u}(t;\mu) \,\text{with}\, t \in [0, T], \mu \in \Gamma\} \subset \mathbb{R}^{2N}$$

formed by the values taken by the solutions of the ODE Eq. (4.7). The manifold structure results from the Cauchy-Lipschitz (Picard-Lindelöf) theorem with parameters under some regularity assumptions of the Hamiltonian. We assume that $\mathcal{M}$ is well approximated by a trial manifold $\widehat{\mathcal{M}}$ that reads

$$\widehat{\mathcal{M}} = \left\{\mathcal{D}\left(\bar{\mathbf{u}}(t;\mu)\right) \text{ with } \bar{\mathbf{u}}(t;\mu) \in \mathbb{R}^{2K}\right\} \subset \mathbb{R}^{2N},$$

with a decoding operator $\mathcal{D} : \mathbb{R}^{2K} \to \mathbb{R}^{2N}$. In addition, we consider its pseudo-inverse operator $\mathcal{E} : \mathbb{R}^{2N} \to \mathbb{R}^{2K}$, called the encoder, which satisfies the relation

$$\mathcal{E} \circ \mathcal{D} = \text{Id}_{\mathbb{R}^{2K}}.$$

In other words, we search for a reduced model that is a $2K$-dimensional ODE of solution $\bar{\mathbf{u}}(t;\mu)$. To do so, we have to determine $\mathcal{D}$ and $\mathcal{E}$, we therefore ask for the projection operator $\mathcal{D}\circ\mathcal{E}$ onto $\widehat{\mathcal{M}}$ to be close to the identity on a data set $U\subset\mathcal{M}$:

$$\forall \mathbf{u}\in U,\quad \mathcal{D}\circ\mathcal{E}(\mathbf{u})\approx\mathbf{u}.$$

The data set U is composed of snapshots of the solutions at different times and various parameters, obtained with time integration; it writes

$$U=\left\{\mathbf{u}_{\mu_1}^0,\dots,\mathbf{u}_{\mu_1}^{n_T},\dots,\mathbf{u}_{\mu_P}^0,\dots,\mathbf{u}_{\mu_P}^{n_T}\right\}\in\mathcal{M}_{2N,(n_T+1)P}(\mathbb{R}), \tag{4.10}$$

where $\mathbf{u}_{\mu_p}^k\approx\mathbf{u}(t_k;\mu_p)$ it the numerical solution at time step k and parameters μ_p, $n_T+1>0$ is the total number of time steps and $P>0$ is the number of sampled parameters. In practice, parameters are uniformly sampled across Γ. We denote this sample $\Gamma^{\text{train}}:=\{\mu_p\}_{p\in\{1,\dots,P\}}$.

In addition, we constrain the reduced variables $\bar{\mathbf{u}}(t;\mu)$ to follow the reduced Hamiltonian dynamics

$$\begin{cases}\dfrac{d}{dt}\bar{\mathbf{u}}(t;\mu)=J_{2K}\nabla_{\bar{\mathbf{u}}}\bar{\mathscr{H}}\left(\bar{\mathbf{u}}(t;\mu)\right), & \text{in } [0,T],\\ \bar{\mathbf{u}}(0;\mu)=\mathcal{E}\left(\mathbf{u}_{\text{init}}(\mu)\right),\end{cases} \tag{4.11}$$

where $\bar{\mathscr{H}}:\mathbb{R}^{2K}\to\mathbb{R}$ is a reduced Hamiltonian, to be built.

In the following, we present our strategy to construct the encoder and decoder. It is based on the coupling of the Proper Symplectic Decomposition (PSD), introduced in [29], and the AE-HNN method proposed in [32], which combines an AutoEncoder (AE) [33] and a Hamiltonian Neural Network (HNN) [34]. The method will be referred to as PSD-AE-HNN.

4.3.1 PSD-AE-HNN reduction method

The PSD-AE-HNN is a three-step reduction method.

First, the Hamiltonian FOM, which evolves in a $2N$-dimensional phase space, is projected onto an intermediate $2M$-dimensional symplectic subspace with $M\ll N$ using the PSD. Let $A\in\mathcal{M}_{2N,2M}(\mathbb{R})$ denote the symplectic matrix obtained from the PSD algorithm, and $A^+\in\mathcal{M}_{2M,2N}(\mathbb{R})$ be its symplectic inverse, so that $A^+A=I_{2M}$. Together, A and A^+ serve as projection and reconstruction operators between the full and intermediate reduced phase space. Further details are provided in Sec. 4.3.2.

Second, we further reduce the intermediate $2M$-dimensional representation to a low-dimensional $2K$-dimensional subspace, with $K\ll M$, using an AE. This neural network consists of an encoder $\mathcal{E}_{\theta_e}:\mathbb{R}^{2M}\to\mathbb{R}^{2K}$ and a decoder $\mathcal{D}_{\theta_d}:\mathbb{R}^{2K}\to\mathbb{R}^{2M}$, where θ_e and θ_d denote the respective parameters of the encoder and decoder. Additional details on the network architectures and training setup are provided in Sec. 4.3.4. The autoencoder is trained to approximate the identity mapping, i.e. $\mathcal{D}_{\theta_d}\circ\mathcal{E}_{\theta_e}\approx\mathrm{id}$. These networks thus act as nonlinear projectors, mapping data from the intermediate subspace to the low-dimensional space and back. Consequently, the full encoder and decoder are defined by

$$\mathcal{E}=\mathcal{E}_{\theta_e}\circ A^+,\quad \mathcal{D}=A\circ\mathcal{D}_{\theta_d},$$

where A^+ (resp. A) is identified with the map $\mathbf{u}\mapsto A^+\mathbf{u}$ (resp. $\mathbf{u}\mapsto A\mathbf{u}$).

Third, the dynamics of the reduced model is then captured by a third neural network, the HNN, denoted $\bar{\mathscr{H}}_{\theta_h}:\mathbb{R}^{2K}\to\mathbb{R}$, where θ_h represents its trainable parameters. It is trained such that Eq. (4.11) holds when evaluated on the reduced variables:

$$\begin{cases}\dfrac{d}{dt}\mathcal{E}(\mathbf{u}(t;\mu))\approx J_{2K}\nabla_{\bar{\mathbf{u}}}\bar{\mathscr{H}}_{\theta_h}\left(\mathcal{E}(\mathbf{u}(t;\mu))\right), & \text{in } [0,T]\\ \bar{\mathbf{u}}(0;\mu)=\mathcal{E}\left(\mathbf{u}_{\text{init}}(\mu)\right)\end{cases} \tag{4.11}$$

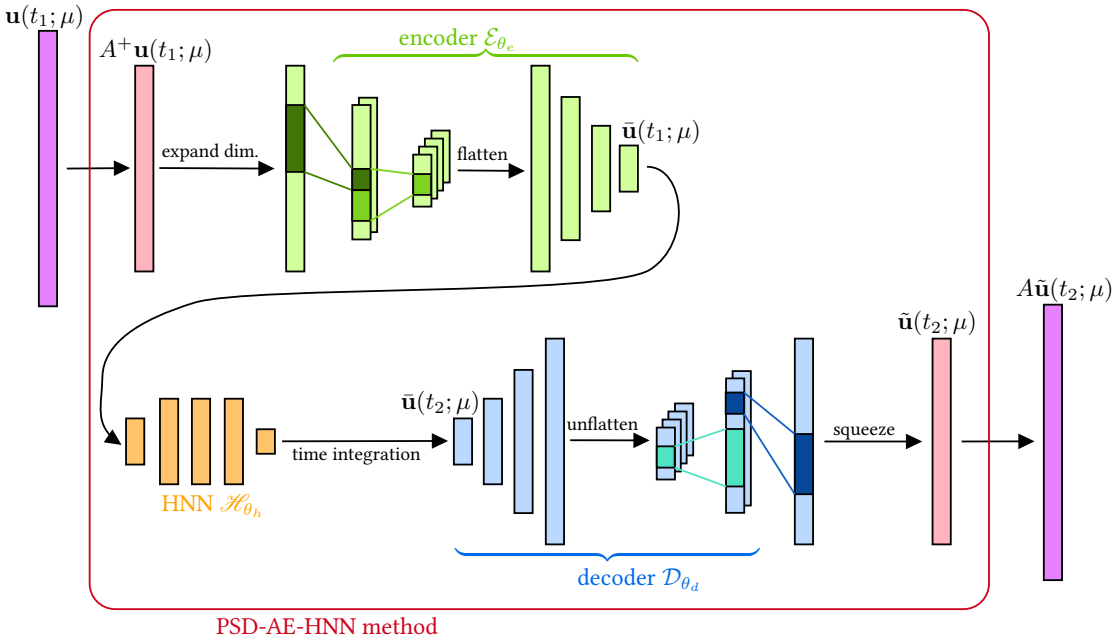


Figure 4.1: PSD-AE-HNN architecture: from FOM solution $\mathbf{u}(t_1;\mu)$, a PSD intermediate reduced variable $A^+\mathbf{u}(t_1;\mu)$ is computed, followed by the reduced state $\bar{\mathbf{u}}(t_1;\mu) = \mathcal{E}_{\theta_e}(A^+\mathbf{u}(t_1;\mu))$. Next, time integration $\bar{\mathbf{u}}(t_2;\mu)$ is performed with the HNN gradient, the final state $A\tilde{\mathbf{u}}(t_2;\mu) = A\mathcal{D}_{\theta_d}(\bar{\mathbf{u}}(t_2;\mu))$ is recovered with decoder and PSD successive decompression.

A more detailed description of this component is provided in Sec. 4.3.3.

The online process for applying the reduced model is schematized in Fig. 4.1. We start with a full order solution $\mathbf{u}(t_1;\mu) \in \mathbb{R}^{2N}$ at time t_1. Our goal is to approximate the full order solution at time $t_2 > t_1$, using the reduced model. We first apply the symplectic projection to an intermediate reduced variable

$$A^+\mathbf{u}(t_1;\mu) \in \mathbb{R}^{2M}.$$

The encoder then maps this intermediate representation to a low-dimensional reduced state

$$\bar{\mathbf{u}}(t_1;\mu) = \mathcal{E}_{\theta_e}(A^+\mathbf{u}(t_1;\mu)) \in \mathbb{R}^{2K}.$$

Since $\bar{\mathbf{u}}(t;\mu)$ evolves according to a Hamiltonian system defined by the HNN, we employ the Störmer-Verlet integrator described in Eq. (4.9) to advance the solution in time up to time t_2. The required gradients of the learned Hamiltonian are computed via backpropagation, allowing us to obtain the reduced state $\bar{\mathbf{u}}(t_2;\mu)$ at time t_2. Finally, the decompression step is performed to recover an approximation of the full-order solution. The reduced state $\bar{\mathbf{u}}(t_2;\mu)$ is first decoded to the intermediate space via

$$\tilde{\mathbf{u}}(t_2;\mu) = \mathcal{D}_{\theta_d}(\bar{\mathbf{u}}(t_2;\mu)).$$

Finally, we apply the symplectic lift to reconstruct the full-order approximation

$$A\tilde{\mathbf{u}}(t_2;\mu) \approx \mathbf{u}(t_2;\mu).$$

There are two main motivations for combining the PSD with the AE-HNN. First, although the AE-HNN is an efficient data-driven model reduction technique, its computational cost scales with its input dimension. For large N, this results in neural networks that are too large to train effectively. Second, since the inputs correspond to particles in phase space, they are inherently unstructured and may contain noise. The prior reduction via PSD projects the dynamics onto a

lower-dimensional symplectic subspace, resulting in a more structured and compact representation. This intermediate reduced variable is both easier to learn for the autoencoder and HNN while also preserving the underlying Hamiltonian structure.

The offline stage of the method, to construct the different elements of the reduced order model, consists of three main steps:

(i) snapshot generation: we compute a collection of full order solutions at various times and for different parameter values;

(ii) symplectic basis construction: we apply the PSD algorithm to build the reduced symplectic basis A;

(iii) neural network training: following the approach of [32], we simultaneously train the second stage of the encoder, $\mathcal{E}_{\theta_e}$, the first stage of the decoder, $\mathcal{D}_{\theta_d}$, and the HNN, $\mathscr{H}_{\theta_h}$. These networks are trained using the FOM snapshots projected onto the intermediate subspace via A^+.

We dive into both PSD and AE-HNN functioning in the following sections.

4.3.2 PSD reduction

In this section, we briefly introduce the Proper Symplectic Decomposition (PSD) [29]. The goal of PSD is to approximate the manifold $\mathcal{M} \subset \mathbb{R}^{2N}$ of full order states with a $2M$-dimensional linear subspace. To preserve the Hamiltonian structure of the dynamics, we require the projection to be symplectic. Under this constraint, the intermediate reduced variable $\widetilde{\mathbf{u}}(t;\mu) \in \mathbb{R}^{2M}$ is defined by

$$\widetilde{\mathbf{u}}(t;\mu) = A^+\mathbf{u}(t;\mu), \tag{4.12}$$

where A^+ denotes the symplectic inverse of a matrix $A \in \mathrm{Sp}_{2M,2N}(\mathbb{R})$. This set denotes the symplectic Stiefel manifold, which consists of all $2N \times 2M$ matrices A satisfying the symplectic condition

$$A^T J_{2N} A = J_{2M}.$$

For any matrix $A \in \mathrm{Sp}_{2M,2N}(\mathbb{R})$, its symplectic inverse A^+ is given by

$$A^+ = J_{2M}^T A^T J_{2N}, \tag{4.13}$$

which satisfies $A^+A = I_{2M}$.

The symplectic matrix A is computed by minimizing the reconstruction error over a set of training snapshots. That is, A is obtained as the solution to the following optimization problem

$$\min_{A \in \mathrm{Sp}_{2M,2N}(\mathbb{R})} \left\| U - AA^+U \right\|_F, \tag{4.14}$$

where $\|X\|_F := \sqrt{\sum_{i,j} |x_{i,j}|^2}$ is the Frobenius norm and U is the snapshot matrix defined in Eq. (4.10). A direct solution of Eq. (4.14) cannot be obtained. However, with additional assumptions outlined in Chap. 2, Sec. 2.3.1, we construct the matrix A using the Complex Singular Value Decomposition (SVD) algorithm [29]. Since A is a symplectic transformation, it can be checked that the intermediate reduced variable $\widetilde{\mathbf{u}}$ evolves according to the Hamiltonian dynamics with Hamiltonian function $\mathscr{H} \circ A$:

$$\begin{cases} \dfrac{d}{dt}\widetilde{\mathbf{u}}(t;\mu) = J_{2M}\nabla_{\tilde{\mathbf{u}}}(\mathscr{H} \circ A)\left(\widetilde{\mathbf{u}}(t;\mu)\right) = J_{2M}A^T\nabla_{\mathbf{u}}\mathscr{H}\left(A\bar{\mathbf{u}}(t;\mu)\right), \\ \widetilde{\mathbf{u}}(0;\mu) = A^+\mathbf{u}_{\text{init}}(\mu). \end{cases} \tag{4.15}$$

As observed in Sec. 4.4, the value of M required to achieve satisfactory precision is often too large, which reduces the efficiency of a reduced model based solely on the PSD. Additionally, the evaluation of the reduced Hamiltonian gradients, $A^T\nabla_{\mathbf{u}}\mathscr{H}(A\,\cdot)$, still depends on the gradient of the original Hamiltonian function. This results in a computational cost that is higher than that of the FOM itself. To address this issue, hyper-reduction techniques have been proposed, as in [16].

4.3.3 AE-HNN reduction

This section provides a brief overview of the AE-HNN method introduced in [32]. The method consists of training simultaneously an auto-encoder, composed of $\mathcal{E}_{\theta_e}, \mathcal{D}_{\theta_d}$, and a Hamiltonian Neural Network, $\bar{\mathscr{H}}_{\theta_h}$. The AE consists of a pair of convolutional neural networks, with convolutional layers followed by dense layers, while the HNN is implemented as a dense neural network [39, 33]. The neural network parameters, $(\theta_e, \theta_d, \theta_h) \in \Theta$, are determined by solving an optimization problem of the form

$$\underset{(\theta_e,\theta_d,\theta_h)\in\Theta}{\operatorname{argmin}} \quad \mathcal{L}(\theta_e, \theta_d, \theta_h),$$

where the loss function $\mathcal{L}$ is computed using the training dataset $\widetilde{U}$, composed of the snapshots U projected onto the intermediate subspace

$$\widetilde{U} = \left\{\widetilde{\mathbf{u}}^0_{\mu_1}, \ldots, \widetilde{\mathbf{u}}^{n_T}_{\mu_P}\right\} = \left\{A^+\mathbf{u}^0_{\mu_1}, \ldots, A^+\mathbf{u}^{n_T}_{\mu_P}\right\}.$$

A gradient descent algorithm is used to determine optimal parameters.

In the AE-HNN method, the loss function is composed of four different loss terms. The first term, $\mathcal{L}_{\text{AE}}$, forces the AE to be close to the identity map, i.e. $\mathcal{D}_{\theta_d} \circ \mathcal{E}_{\theta_e} \approx \text{id}$, on the training dataset:

$$\mathcal{L}_{\text{AE}}(\theta_e, \theta_d) = \sum_{\widetilde{\mathbf{u}}\in\widetilde{U}} \left\|\widetilde{\mathbf{u}} - \left(\mathcal{D}_{\theta_d} \circ \mathcal{E}_{\theta_e}\right)(\widetilde{\mathbf{u}})\right\|_2^2. \tag{4.16}$$

In practice, a split AE is employed where both the encoder and decoder are made of two neural networks. The first network processes the generalized positions, while the second network processes the generalized velocities. Specifically, the encoder and decoder are structured as follows

$$\mathcal{E}_{\theta_e} = \begin{pmatrix}\mathcal{E}_{\theta_{e,1}}\\ \mathcal{E}_{\theta_{e,2}}\end{pmatrix}, \quad \mathcal{D}_{\theta_d} = \begin{pmatrix}\mathcal{D}_{\theta_{d,1}}\\ \mathcal{D}_{\theta_{d,2}}\end{pmatrix}.$$

The reduced state is then given by

$$\bar{\mathbf{u}}(t;\mu) = \begin{pmatrix}\mathbf{x}(t;\mu)\\ \mathbf{v}(t;\mu)\end{pmatrix} = \begin{pmatrix}\mathcal{E}_{\theta_{e,1}}(\widetilde{\mathbf{x}}(t;\mu))\\ \mathcal{E}_{\theta_{e,2}}(\widetilde{\mathbf{v}}(t;\mu))\end{pmatrix}$$

and conversely for the decoded state.

The second loss term is defined to constrain the reduced trajectories $\bar{\mathbf{u}}(t;\mu)$ to be close to those of a Hamiltonian system, as described in Eq. (4.11). In practice, these reduced dynamics are defined through a time discretization. We therefore introduce the prediction operator

$$\mathcal{P}_s\left(\bar{\mathbf{u}}, \bar{\mathscr{H}}_{\theta_h}\right)$$

which consists in performing $s \in \mathbb{N}^*$ iterations of the Störmer-Verlet algorithm (4.9), starting from the reduced state $\bar{\mathbf{u}}$ and using the reduced Hamiltonian $\bar{\mathscr{H}}_{\theta_h}$. The number of iterations considered s, also called the watch duration, is a hyperparameter, which must be set. With this prediction operator, the second loss function, $\mathcal{L}^s_{\overline{\text{pred}}}$, constrains the HNN to accurately capture the reduced dynamics between the n-th and $(n+s)$-th time steps

$$\mathcal{L}^s_{\overline{\text{pred}}}(\theta_e, \theta_h) = \sum_{\widetilde{\mathbf{u}}^n, \widetilde{\mathbf{u}}^{n+s}\in\widetilde{U}} \left\|\bar{\mathbf{u}}^{n+s} - \mathcal{P}_s\left(\bar{\mathbf{u}}^n; \bar{\mathscr{H}}_{\theta_h}\right)\right\|_2^2, \tag{4.17}$$

where $\widetilde{\mathbf{u}}^n, \widetilde{\mathbf{u}}^{n+s} \in \widetilde{U}$ denotes the sampling of random pairs on the dataset $\widetilde{U}$. Since the full order Hamiltonian in Eq. (4.5) is separable, the reduced Hamiltonian $\bar{\mathscr{H}}_{\theta_h}$ is also assumed to be separable:

$$\bar{\mathscr{H}}_{\theta_h}(\bar{\mathbf{u}}) = \bar{\mathscr{H}}^{\text{kin}}_{\theta_h}(\mathbf{v}) + \bar{\mathscr{H}}^{\text{pot}}_{\theta_h}(\mathbf{x}),$$

which allows for the explicit formulation of the time integrator $\mathcal{P}_s$.

The third part of the loss function aims at ensuring that the reduced trajectories, generated by the encoder $\mathcal{E}_{\theta_e}$, preserve the reduced Hamiltonian. The loss function, $\mathcal{L}^s_{\overline{\text{stab}}}$ writes:

$$\mathcal{L}^s_{\overline{\text{stab}}}(\theta_e, \theta_h) = \sum_{\bar{\mathbf{u}}^n, \bar{\mathbf{u}}^{n+s} \in \bar{U}} \left\| \bar{\mathscr{H}}_{\theta_h}\left(\bar{\mathbf{u}}^{n+s}\right) - \bar{\mathscr{H}}_{\theta_h}\left(\bar{\mathbf{u}}^{n}\right) \right\|_2^2 . \tag{4.18}$$

Finally, the three neural networks are strongly coupled using a loss function, $\mathcal{L}^s_{\text{pred}}$, which encapsulates the full prediction from the n-th time step to the $n+s$-th time step, using the encoder at the beginning, the decoder at the end and the prediction operator associated with the reduced model:

$$\mathcal{L}^s_{\text{pred}}(\theta_e, \theta_d, \theta_h) = \sum_{\widetilde{\mathbf{u}}^n, \widetilde{\mathbf{u}}^{n+s} \in \widetilde{U}} \left\| \widetilde{\mathbf{u}}^{n+s} - \mathcal{D}_{\theta_d}\left(\mathcal{P}_s\left(\mathcal{E}_{\theta_e}(\widetilde{\mathbf{u}}^n); \bar{\mathscr{H}}_{\theta_h}\right)\right) \right\|_2^2 . \tag{4.19}$$

To summarize, four different loss functions, given in Eqs. (4.16) to (4.19), are used to train the AE and the HNN. More precisely, the parameters are determined such as to minimize the following weighted sum

$$\begin{aligned}\mathcal{L}(\theta_e, \theta_d, \theta_h) =& \omega_{\text{AE}}\, \mathcal{L}_{\text{AE}}(\theta_e, \theta_d) + \omega_{\overline{\text{pred}}}\, \mathcal{L}^s_{\overline{\text{pred}}}(\theta_e, \theta_h) \\ &+ \omega_{\overline{\text{stab}}}\, \mathcal{L}^s_{\overline{\text{stab}}}(\theta_e, \theta_h) + \omega_{\text{pred}}\, \mathcal{L}^s_{\text{pred}}(\theta_e, \theta_d, \theta_h),\end{aligned}$$

where $\omega_{\text{AE}}, \omega_{\overline{\text{pred}}}, \omega_{\overline{\text{stab}}}$ and ω_{pred} are positive weights. The networks are thus jointly trained, with potentially adversarial goals. Ultimately, $\mathcal{L}^s_{\text{pred}}$ serves as the primary loss function to measure the performance of the AE-HNN reduction. The other loss functions act as auxiliary functions to drive the training process.

4.3.4 Hyperparameters tuning

This section specifies the hyperparameters of the models and describes how they are selected. They are chosen based on two main criteria. First, the reduced model must closely approximate the full model, with the difference measured by the losses, while minimizing the reduced dimension K. Second, the networks, particularly the HNN, must remain lightweight in terms of the number of parameters to ensure fast computation. Note that the AE is less critical in terms of size, as it is only called once during the online phase.

Regarding the PSD part, the main hyperparameter is the intermediate subspace dimension M. A smaller value of M results in a more significant reduction and reduces the computation time, while a larger value of M provides a richer subspace for the subsequent training the AE-HNN, but with increased computation time. In practice, we select an M value such that the PSD reconstruction error is slightly less than the target accuracy of the reduced model. In the following test cases, a typical value is $M = 121$ for a final time $T = 20$ and $M = 256$ when $T = 40$.

Secondly, the AE-HNN part involves hyperparameters for defining the architecture of the neural networks. As explained in Sec. 4.3.3, the encoder consists of two convolutional neural networks when considering a split AE. Each network starts with an input of size M, which is fed through a series of 1D convolutional layers with a stride of 3, a kernel size of 3, and valid padding each. The number of filters is progressively multiplied by the stride between layers, starting with 12 filters. The final output is flattened and passed through a series of dense layers, whose sizes gradually decrease, ultimately leading to a single dense layer of output size K. The activation function is applied throughout the encoder, except for the output layer, which uses a linear activation function. The decoder is designed as a mirror image of the encoder, where

the 1D convolutions are replaced with 1D transposed convolutions. Lastly, the HNN is a simple multi-layer perceptron with an input size of $2K$ and an output size of 1. The activation function in the HNN may differ from that used in the AE.

The AE-HNN also requires some hyperparameters to be fixed for the training. The chosen optimization method is the Adam algorithm [40], which is an adaptive stochastic gradient descent method. The learning rate follows the rule

$$\rho_k = (0.99)^{k/150}\,\rho_0,$$

where the division operator denotes integer division and k is the training step. Additionally, we can reset the decay, i.e. set $k = 0$, if the loss function reaches a plateau. The purpose of this reset strategy is to escape poor local minima by introducing a sudden, larger learning rate. In most cases, we start training with a large $\rho_0 = 10^{-3}$ to accelerate the convergence. Then, we diminish it to $\rho_0 = 5\times10^{-4}$ or so for fine-tuning. The training process depends on the watch duration s. It is be set to $s = 8$ and then be reasonably increased up to $s = 32$ to improve predictions. Finally,training is divided in two stages. First, the AE is trained alone by setting

$$\omega_{\text{AE}} = 1, \quad \omega_{\overline{\text{pred}}} = \omega_{\overline{\text{stab}}} = \omega_{\text{pred}} = 0.$$

Then, after the loss has reached a value in the range $[5\times10^{-3}, 1\times10^{-2}]$, the AE and the HNN are trained together by setting

$$\omega_{\text{AE}} = 1, \quad \omega_{\overline{\text{pred}}} = 10, \quad \omega_{\overline{\text{stab}}} = 1\times10^{-4}, \quad \omega_{\text{pred}} = 1.$$

Table 4.1 recapitulates the hyperparameters used for the different test cases of the next section.

		linear Landau damping	nonlinear Landau damping	two stream instability
AE	nb of convolution blocks (encoder)	2	2	2
	dense layers (encoder)	$[150, 100, 50, 25]$	$[250, 150, 100, 50, 25]$	$[150, 100, 50, 25]$
	activation functions	ELU	ELU	ELU
HNN	dense layers	$[48, 24, 24, 24, 12]$	$[96, 48, 48, 48, 24]$	$[48, 24, 24, 24, 12]$
	activation functions	softplus	softplus	softplus
watch duration	s	16	$10 \to 22$	$16 \to 32$

Table 4.1: Hyper-parameters. Activation functions are used except for the last layer of the neural networks. ELU refers to the function $\text{elu}(x) = x1_{x>0} + (e^x - 1)1_{x<0}$ and softplus to the function $\text{softplus}(x) = \log(1 + e^x)$. For the autoencoder (AE), the number of convolution blocks and the sizes of the hidden of layers are those of the encoder. The decoder is constructed in a mirror way.

Remark. *In practice, pre-processing is applied to the neural network inputs. While such functions could be learned by the first layers of the network, manually selecting them significantly improves both the training and prediction processes. Considering the SVD of the snapshot matrix U defined in Eq.* (4.10)*,*

$$U = W\Sigma V^*,$$

with W and V unitary matrices, V^* is the conjugate transpose of V and Σ a diagonal matrix of diagonal values $(\sigma_k)_k$ sorted in descending order, the encoder input is pre-processed with the function

$$(\widetilde{\mathbf{u}})_k \mapsto \sigma_k^{-1/2}(\widetilde{\mathbf{u}})_k,$$

where $(\widetilde{\mathbf{u}})_k$ is the k-th coefficient of the intermediate reduced variable $\widetilde{\mathbf{u}}$. The idea is to balance the influence of each singular PSD vector in the intermediate reduced basis, thereby allowing the AE to capture the most important modes without overly neglecting the other modes.

4.4 Numerical results

In this section, the PSD-AE-HNN reduction of the PIC method is tested on three classical plasma physics dynamics: the linear Landau damping, the nonlinear Landau damping, and the two-stream instability test cases.

The parameterized initial distributions of the particles takes the following form

$$f_{\text{init}}(x, v; \mu) = f_{\text{init},x}(x; \alpha)\, f_{\text{init},v}(v; \sigma),$$

with parameters $\mu = (\alpha, \sigma)^T \in \Gamma \subset \mathbb{R}^2$. The initial position distribution is a perturbed uniform distribution

$$f_{\text{init},x}(x; \alpha) = \frac{k}{2\pi}\left(1 + \alpha \cos(k\, x)\right), \tag{4.20}$$

defined over $\Omega_x = \left[0, \frac{2\pi}{k}\right)$, where $k > 0$ is a fixed wave number. The parameter $\alpha > 0$ is the perturbation amplitude. The initial velocity distribution $f_{\text{init},v}(v; \sigma)$, defined over $\Omega_v = [-6, 6]$, is given by a Gaussian

$$f_{\text{init},v}(v; \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{v^2}{2\sigma^2}\right), \tag{4.21}$$

for the Landau test cases, and by the sum of two Gaussian

$$f_{\text{init},v}(v; \sigma) = \frac{1}{2\sigma\sqrt{2\pi}} \left[\exp\left(-\frac{(v-3)^2}{2\sigma^2}\right) + \exp\left(-\frac{(v+3)^2}{2\sigma^2}\right)\right], \tag{4.22}$$

for the two stream instability test case, where $\sigma > 0$ stands for the standard deviation of the Gaussian distributions.

The P training parameters $\mu \in \Gamma^{\text{train}}$ are selected on a $\sqrt{P} \times \sqrt{P}$ grid over Γ. The model is then evaluated on a fine 20×20 grid $\Gamma^{\text{test}} \subset \Gamma$. For each parameter μ, the reference FOM solution is denoted

$$X_\mu^{\text{ref}} = \left\{\mathbf{x}_\mu^0, \ldots, \mathbf{x}_\mu^{n_T}\right\}, \quad V_\mu^{\text{ref}} = \left\{\mathbf{v}_\mu^0, \ldots, \mathbf{v}_\mu^{n_T}\right\},$$

while the solution obtained by the PSD-AE-HNN method is denoted

$$X_\mu^{\text{test}} = \left\{\hat{\mathbf{x}}_\mu^0, \ldots, \hat{\mathbf{x}}_\mu^{n_T}\right\}, \quad V_\mu^{\text{test}} = \left\{\hat{\mathbf{v}}_\mu^0, \ldots, \hat{\mathbf{v}}_\mu^{n_T}\right\}.$$

We recall that it is obtained through the compression of the initial condition, its complete integration over $[0, T]$ using the HNN followed by its decompression. We measure the relative errors on a single parameter μ for all time steps

$$\text{err}_{X,\mu} = \frac{\left\|X_\mu^{\text{test}} - X_\mu^{\text{ref}}\right\|_F}{\left\|X_\mu^{\text{ref}}\right\|_F}, \quad \text{err}_{V,\mu} = \frac{\left\|V_\mu^{\text{test}} - V_\mu^{\text{ref}}\right\|_F}{\left\|V_\mu^{\text{ref}}\right\|_F},$$

and the mean relative errors at a single time t over all $\mu \in \Gamma^{\text{test}}$

$$\text{err}_{X,t}^{\text{mean}} = \text{mean}\left(\frac{\left\|\mathbf{x}_\mu^t - \hat{\mathbf{x}}_\mu^t\right\|_F}{\left\|\mathbf{x}_\mu^t\right\|_F}, \mu \in \Gamma^{\text{test}}\right), \quad \text{err}_{V,t}^{\text{mean}} = \text{mean}\left(\frac{\left\|\mathbf{v}_\mu^t - \hat{\mathbf{v}}_\mu^t\right\|_F}{\left\|\mathbf{v}_\mu^t\right\|_F}, \mu \in \Gamma^{\text{test}}\right).$$

In addition, we also compute the associated maximal and minimal errors.

4.4.1 Linear Landau damping

We first consider the linear Landau damping test cases, with initial distributions (4.20)-(4.21) and $k = 0.5$, $N = 10^5$ particles, $n_x = 48$ spatial discretization points. The final time equals $T = 20$ and the time step is set to $\Delta t = 2.5 \times 10^{-3}$. The parameter domain is taken equal to $\Gamma = [0.03, 0.06] \times [0.8, 1]$: the size of the perturbation is thus kept small. For the training dataset, we consider $P = 64$ parameters. The variation of the initial distribution and the electric energy damping as a function of μ is shown in Fig. 4.2. Each color represents a different parameter in Γ^{train}, and black lines are the envelopes of all the colored curves.

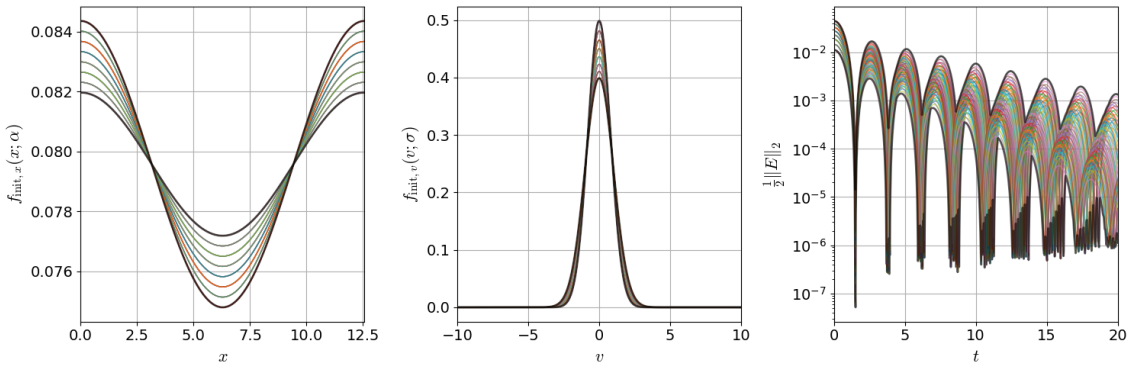


Figure 4.2: (Linear Landau damping) Initial distribution $f_{\text{init},x}(x;\alpha)$ (left), $f_{\text{init},v}(x;\sigma)$ (middle) and evolution of the electric energy $\frac{1}{2}\|E\|_2\left(\mathbf{x}(t;\mu);\mu\right))$ (right) for every $\mu \in \Gamma^{\text{train}}$.

The intermediate reduced variable size is set to $M = 121$ and the complex SVD algorithm is used to build the first linear mapping. After completion of the training process with the architecture from Tab. 4.1, we evaluate our model on $\mu \in \Gamma^{\text{test}}$. To begin with, we vary $K \in \{2, 3, 4\}$ and observe relative errors as a function of time in Fig. 4.3. A larger K leads to smaller errors. For instance, with $K = 2$, $\text{err}_{X,t}^{\text{mean}}$ is around 2×10^{-2}, while it is around 1×10^{-3} for $K = 4$. With this architecture, a reduced dimension $K = 3$ is satisfactory. To obtain more precise results, we would have to modify the architecture presented in Tab. 4.1 for a larger one. In the following, we set $K = 3$.

In Fig. 4.4, we then look at the relatives errors $\text{err}_{X,\mu}, \text{err}_{V,\mu}$ as a function of the parameters. The errors $\text{err}_{X,\mu}, \text{err}_{V,\mu}$ are of the order 6×10^{-3} and 3×10^{-2}, respectively. In this specific case, we note that the maximal error is obtained inside the parameters domain for the positions and on the boundary for the velocities.

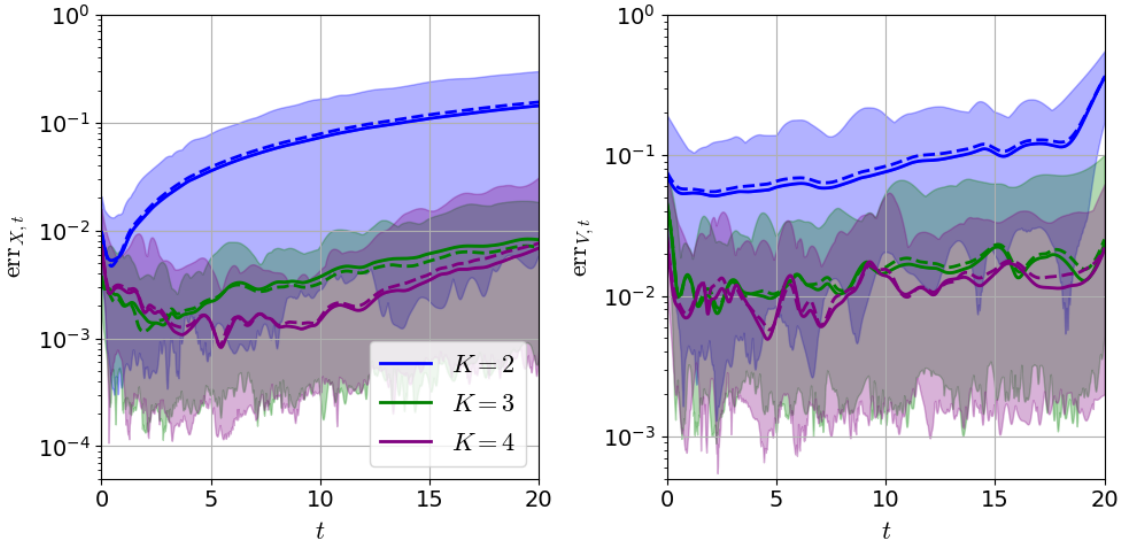


Figure 4.3: (Linear Landau damping) Mean error as a function of time $\mathrm{err}_{X,t}^{\mathrm{mean}}$ (left,solid line) and $\mathrm{err}_{V,t}^{\mathrm{mean}}$ (right, solid) for $\mu \in \Gamma^{\mathrm{test}}$. Each color stands for a value of $K \in \{2,3,4\}$, dashed lines are $\mathrm{err}_{X,t}^{\mathrm{mean}}, \mathrm{err}_{V,t}^{\mathrm{mean}}$ evaluated on the training set Γ^{train}, the envelopes represent minimal and maximal errors $\mathrm{err}_{X,t}^{\mathrm{min}}, \mathrm{err}_{X,t}^{\mathrm{max}}$ (left) and $\mathrm{err}_{V,t}^{\mathrm{min}}, \mathrm{err}_{V,t}^{\mathrm{max}}$.

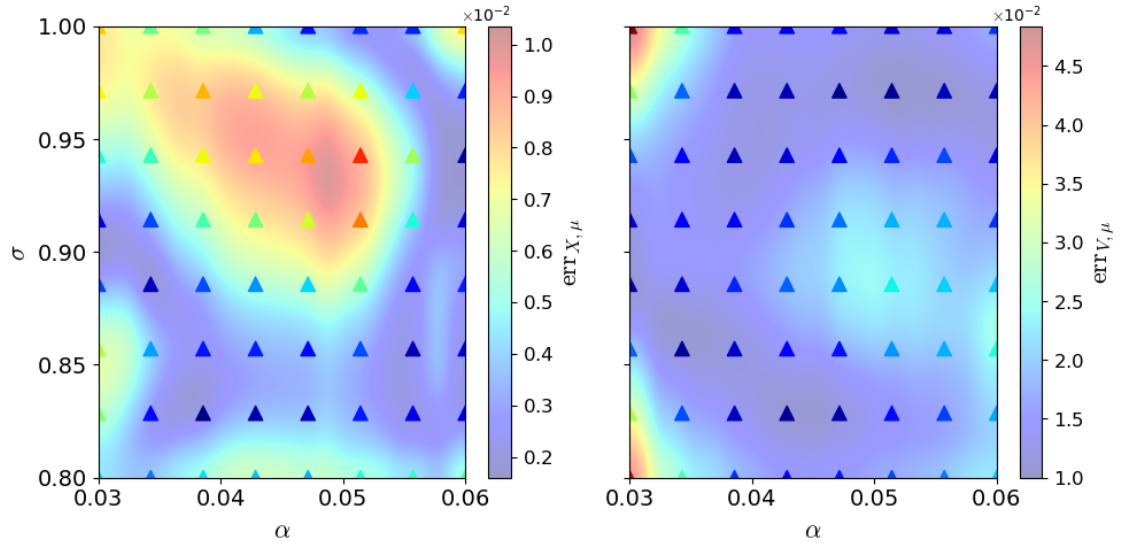


Figure 4.4: (Linear Landau damping) Errors as a function of the reduction parameters $\mathrm{err}_{X,\mu}$ (left) and $\mathrm{err}_{V,\mu}$ (right), triangular points represent the same error evaluated on the training set Γ^{train}.

Next, we investigate the correctness of the damping rate. In theory, the electric energy $\frac{1}{2}\|E\|_2(\mathbf{x}(t;\mu);\mu)$ decays exponentially in time with a constant damping rate, that depends on the standard deviation σ of the Maxwellian initial distribution $f_{\mathrm{init},v}$ and not on the amplitude α of the initial perturbation in space $f_{\mathrm{init},x}$ [41]. This property is captured by the reduced model, as observed in Fig. 4.5. Thus, damping rates predictions are precise with an absolute error of about 5×10^{-3}. In practice, the compression generates an error that causes the decay rate to fluctuate as a function of α, but this fluctuation remains very small. Similarly, we can see that the reduced model slightly underestimates decay rates.

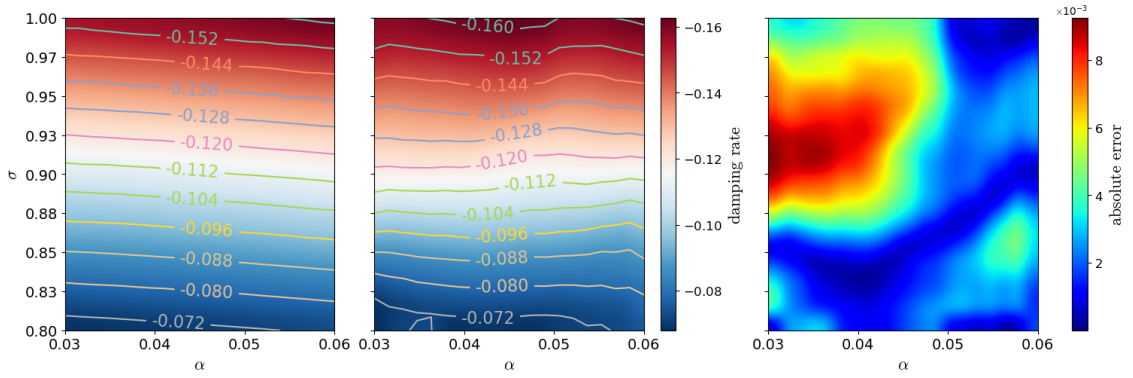
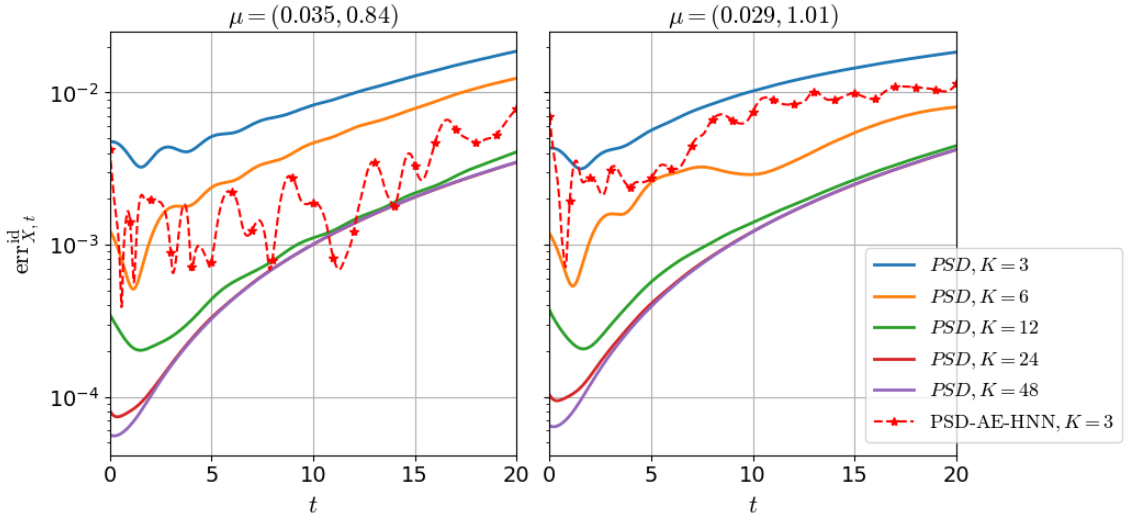
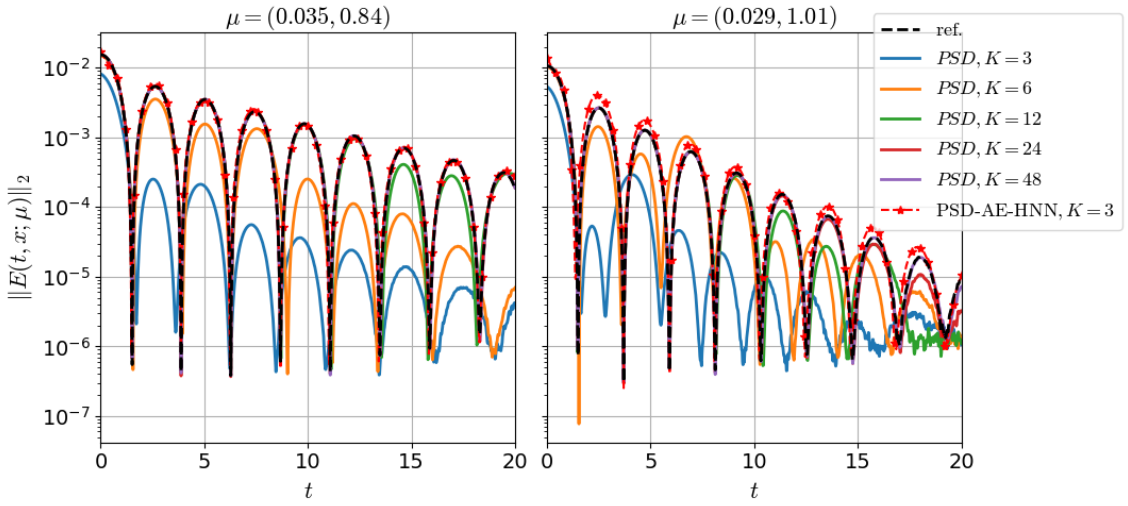


Figure 4.5: (Linear Landau damping) Electric energy $\frac{1}{2}\|E\|_2\left(\mathbf{x}(t;\mu);\mu\right), \mu \in \Gamma^{\text{test}}$ exponential damping rates of the FOM (left), the ROM (center) and absolute error (right).

We evaluate the performance of our method compared to the PSD-only approach in Fig. 4.6. We test both methods with $K \in \{3, 6, 12, 24, 48\}$, and evaluate them at two parameter sets: $\mu = (0.035, 0.84) \in [0.03, 0.06] \times [0.8, 1]$ and $\mu = (0.029, 1.01) \notin [0.03, 0.06] \times [0.8, 1]$. The damping rate, shown in Fig. 4.6b, indicates that $K \approx 30$ modes are required to match the performance of $K = 3$ modes in the PSD-AE-HNN approach. However, as illustrated in Fig. 4.6a, the relative error in particle positions does not show significant differences. This highlights that the PSD method struggles to capture small-scale dynamics, that are crucial for preserving electric energy oscillations and damping, although it still performs well for large-scale dynamics.



(a) Errors $\mathrm{err}^{\mathrm{id}}_{X,t}$



(b) Electric energies $\frac{1}{2}\|E\|_2\left(\mathbf{x}(t;\mu);\mu\right)$

Figure 4.6: (Linear Landau damping) Comparison of the PSD reduced model against our method with $K = 3$, $\mathrm{err}^{\mathrm{id}}_{X,t} = \left\|\mathbf{x}^t_\mu - \hat{\mathbf{x}}^t_\mu\right\|_F / \left\|\mathbf{x}^t_\mu\right\|_F$ for a given $\mu = (0.035, 0.84)$ (left) and $\mu = (0.029, 1.01)$ (right).

4.4.2 Nonlinear Landau damping

In this test case, we keep the same initial distribution as in the previous section but consider a parametric domain, $\Gamma = [0.46, 0.5] \times [0.96, 1]$, with larger spatial perturbation amplitudes. We consider $\times 10^5$ particles, $n_x = 64$ spatial discretization points. The final time and the time step are respectively set to $T = 40$ and $\Delta t = 2.5 \times 10^{-3}$. For the training dataset, we sample $(\alpha\ \sigma)^T \in \Gamma$ pairs over an 8×8 regular grid over Γ forming $P = 64$ pairs Γ^{train}. In Fig. 4.7, we plot the evolution of the initial distribution and the electric energy for all $\mu \in \Gamma^{\mathrm{train}}$. Each color represents a different value of μ and the envelopes are shown in black.

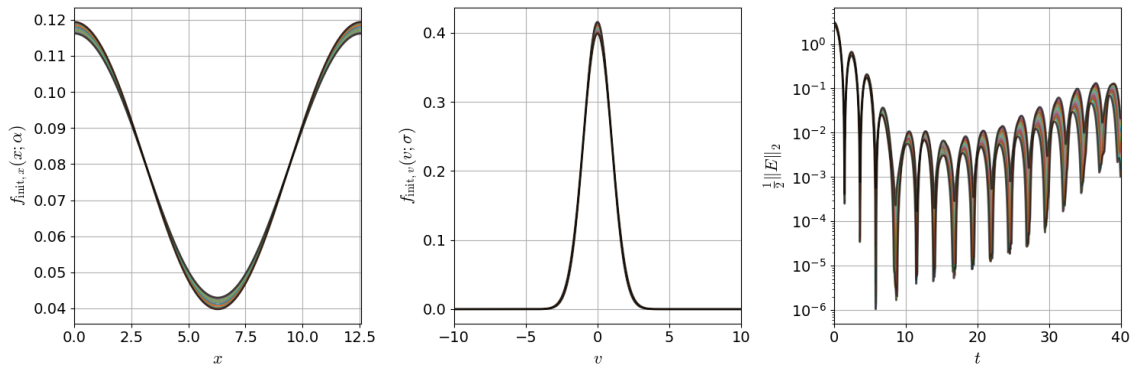


Figure 4.7: (Nonlinear Landau damping) Initial distribution $f_{\text{init},x}(x;\alpha)$ (left), $f_{\text{init},v}(x;\sigma)$ (middle) and evolution of the electric energy $\frac{1}{2}\|E\|_2\,(\mathbf{x}(t;\mu);\mu)$ (right) for every $\mu \in \Gamma^{\text{train}}$.

Given the increased complexity compared with the linear Landau damping, the intermediate reduced dimension is set to $M = 256$. The trained architecture is specified in Tab. 4.1. The reduced dimension is fixed equal to $K = 4$. The relative errors as a function of time are depicted in Fig. 4.8. The errors in positions and velocities are both on the order of 1×10^{-2}.

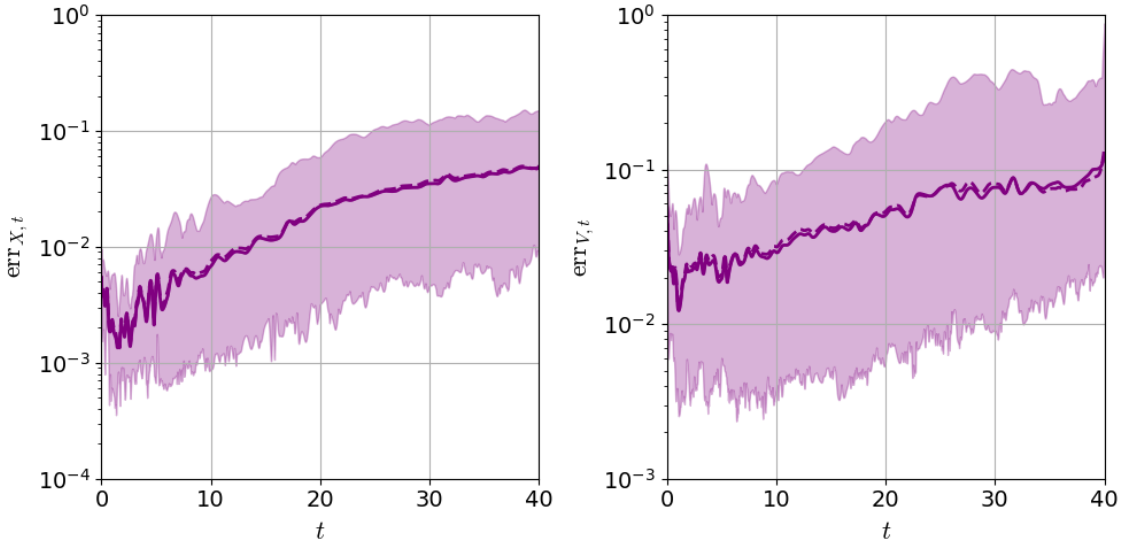


Figure 4.8: (Nonlinear Landau damping) Mean error as a function of time $\text{err}_{X,t}^{\text{mean}}$ (left,solid line) and $\text{err}_{V,t}^{\text{mean}}$ (right, solid) for $\mu \in \Gamma^{\text{test}}$ for $K = 4$. Dashed lines are $\text{err}_{X,t}^{\text{mean}}$, $\text{err}_{V,t}^{\text{mean}}$ evaluated on the training set Γ^{train}, the envelopes represents minimal and maximal errors $\text{err}_{X,t}^{\text{min}}$, $\text{err}_{X,t}^{\text{max}}$ (left) and $\text{err}_{V,t}^{\text{min}}$, $\text{err}_{V,t}^{\text{max}}$ (right).

Subsequently, the relative errors as a function of μ are shown in Fig. 4.9. The errors are around 2×10^{-2} for the positions and 5×10^{-2} for the velocities. As expected, errorr mainly concentrate on the boundary of the parameter domain.

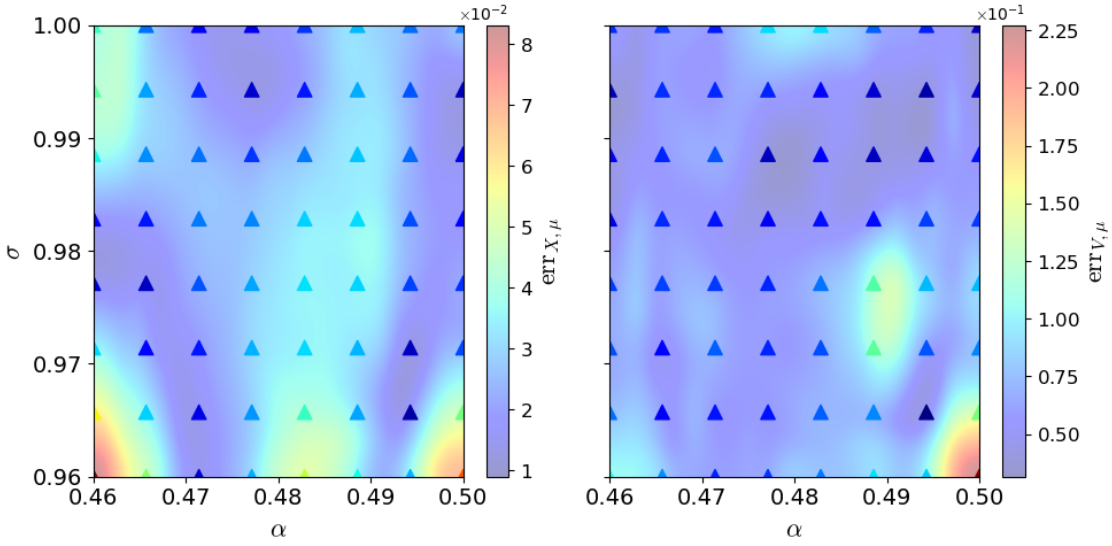
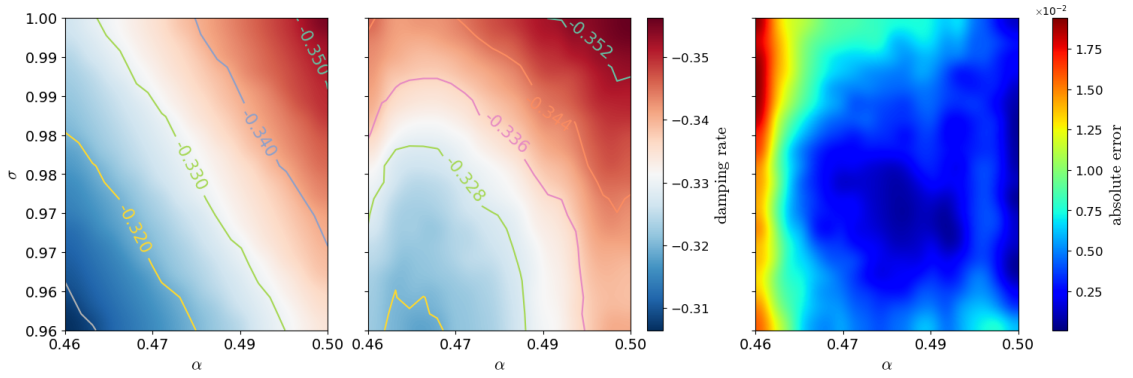
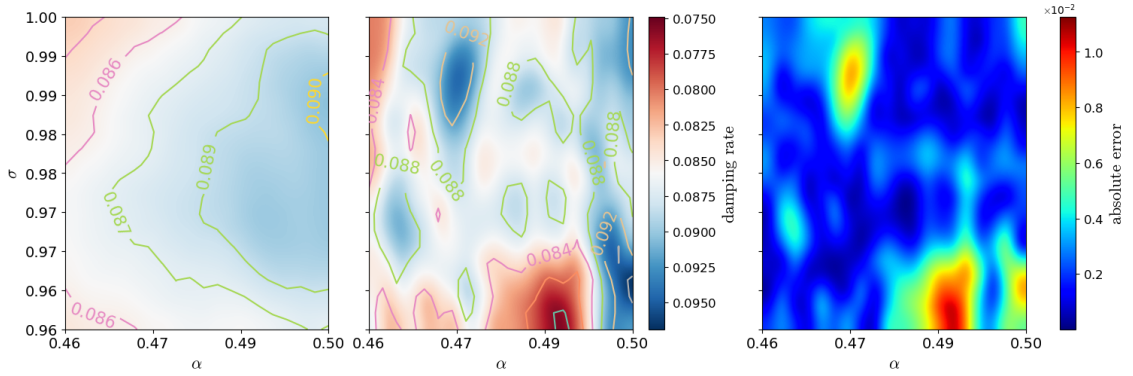


Figure 4.9: (Nonlinear Landau damping) Errors as a function of the reduction parameters $\mathrm{err}_{X,\mu}$ (left) and $\mathrm{err}_{V,\mu}$ (right), triangular points represent the same error evaluated on the training set Γ^{train}.

Then, in Fig. 4.10, we compare the exponential damping and growth rates of the electric energy in ROM with those in FOM.. As observed in Fig. 4.7, we expect a constant damping rate when $t < 10$ then a constant growth rate when $t > 20$. Fig. 4.10a shows that the mean error on the damping rate is about $7{\times}10^{-3}$ and the error is maximal for the smallest values of α. Fig. 4.10b shows that the mean error on the growth rate is about 3×10^{-3}. On the other hand, unlike the linear case, the dependency of the rates to the two parameters is less well captured.



(a) Damping rate.



(b) Growth rate.

Figure 4.10: (Nonlinear Landau damping) Electric energy $\frac{1}{2}\|E\|_2\left(\mathbf{x}(t;\mu);\mu\right), \mu \in \Gamma^{\text{test}}$ exponential damping rates (top) and growth rates (bottom) of the FOM (left), the ROM (center) and absolute error (right).

Next, we compare the evolution of the distribution $f(t, x, v; \mu)$ from the reference model with the predictions of its reduced model in Fig. 4.11. While small differences are observed, the overall dynamics are well captured.

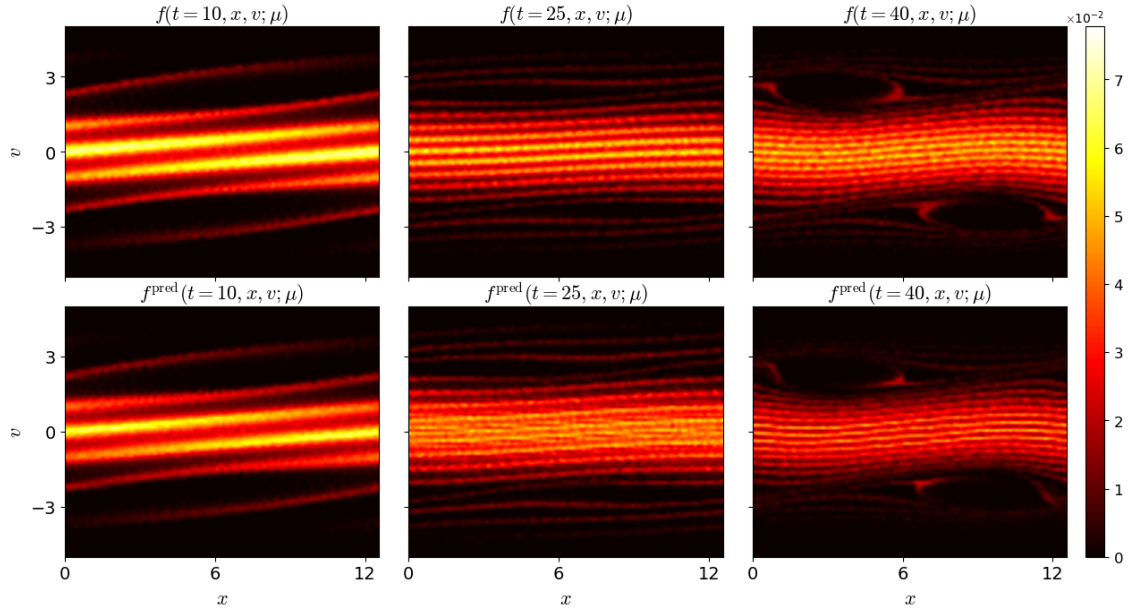
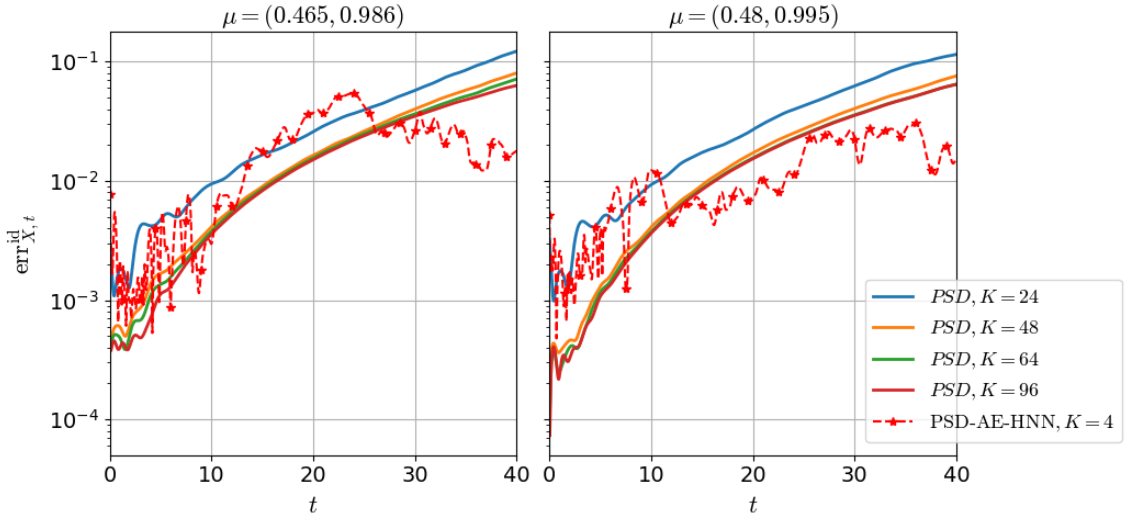
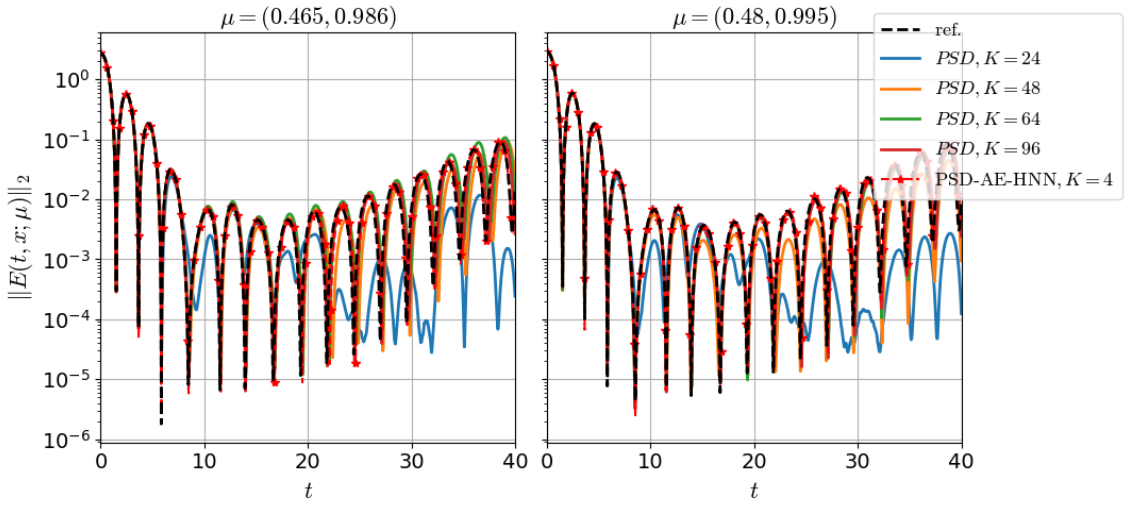


Figure 4.11: (Nonlinear Landau damping) Solution $f(t,x,v;\mu)$ (top) and $f^{\text{pred}}(t,x,v;\mu)$ (bottom) for $t \in \{10, 25, 40\}$ and $\mu = (0.465, 0.986)$.

In Fig. 4.12, we then compare the precision of the method with the PSD-only approach for two test parameters $\mu = (0.465, 0.986)$ and $\mu = (0.48, 0.995)$. To match the performance of our method, about $K = 100$ PSD modes are required, both for the particle distribution errors (Fig. 4.12a) and for the electric energy (Fig. 4.12b). In this test case, our approach is particularly effective for the positions and velocities associated with the last oscillations of the simulation.



(a) Errors $\mathrm{err}^{\mathrm{id}}_{X,t}$



(b) Electric energies $\frac{1}{2}\|E\|_2\left(\mathbf{x}(t;\mu);\mu\right)$

Figure 4.12: (Nonlinear Landau damping) Comparison of the PSD reduced model against our method with $K = 4$, $\mathrm{err}^{\mathrm{id}}_{X,t} = \left\|\mathbf{x}^t_\mu - \hat{\mathbf{x}}^t_\mu\right\|_F / \left\|\mathbf{x}^t_\mu\right\|_F$ for a given $\mu = (0.465, 0.986)$ (left) and $\mu = (0.48, 0.995)$ (right).

Finally, we assess the importance of using a HNN in our PSD-AE-HNN method, compared to a flux-approximating neural approach. For this, we replace the HNN in Sec. 4.3 with a standard neural network $\bar{\mathcal{F}}_{\theta_f}$, which approximates the flux of the reduced model:

$$\begin{cases} \dfrac{d}{dt}\bar{\mathbf{u}}(t;\mu) = \bar{\mathcal{F}}_{\theta_f}\left(\bar{\mathbf{u}}(t;\mu)\right), & \text{in } [0,T], \\ \bar{\mathbf{u}}(0;\mu) = \mathcal{E}\left(\mathbf{u}_{\text{init}}(\mu)\right). \end{cases}$$

We also replace, in the prediction operator $\mathcal{P}_s$, the symplectic Störmer-Verlet by the standard Runge-Kutta 2 scheme. The stability loss weight is set to $\omega_{\overline{\text{stab}}} = 0$ as it cannot be evaluated and the remaining parameters are unchanged. The trained model is tested on $\mu = (0.48, 0.995) \in \Gamma$, and the results are shown in Fig. 4.13. The errors increase strongly over time, and the electric energy growth is not accurately captured—unlike our PSD-AE-HNN method, which performs accurately in comparison. Unlike an approach based solely on PSD, where the reduced model

is Hamiltonian and the encoder and decoder are symplectic, here only the reduced model is Hamiltonian. This last case shows that simply preserving the structure in the reduced space is sufficient to improve long-term stability.

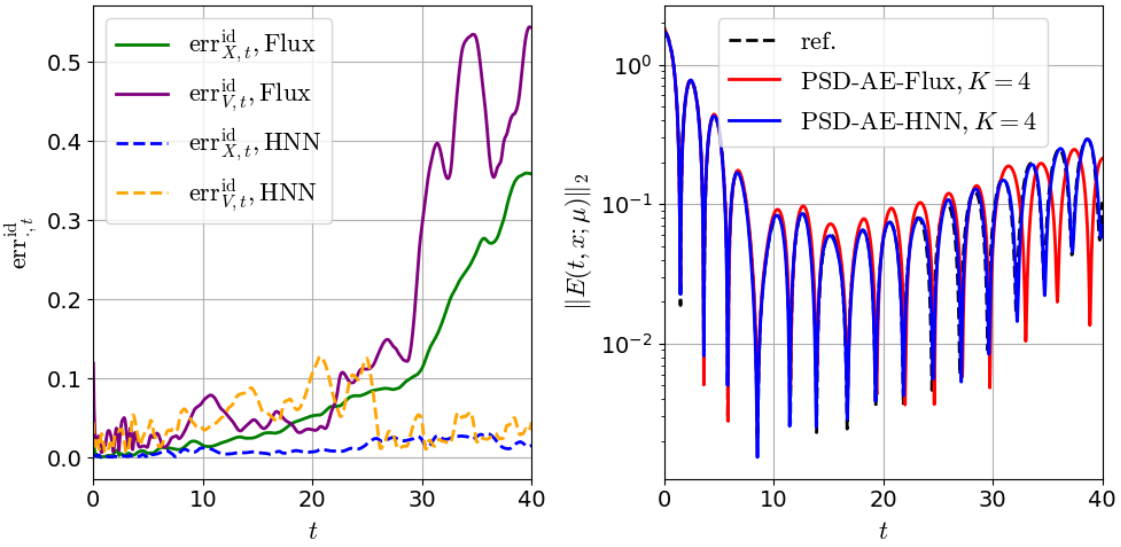


Figure 4.13: (Nonlinear Landau damping) PSD-AE-Flux prediction for a single test parameter μ compared to the PSD-AE-HNN method. Errors as a function of time $\mathrm{err}^{\mathrm{id}}_{X,t}, \mathrm{err}^{\mathrm{id}}_{V,t}$ (left) and predicted electric energy $\frac{1}{2}\|E\|_2\left(\mathbf{x}(t;\mu);\mu\right)$.

4.4.3 Two-stream instability

In this third test case, we consider the initial distributions (4.20)-(4.22), with the sum of two Gaussians in velocity, and $k = 0.2$. The number of particles equals $N = 1.5\times10^5$ and there are $n_x = 64$ spatial discretization points. The final time equals $T = 20$ and the time step $\Delta t = 2.5\times10^{-3}$. The parameter domain is taken equal to $\Gamma = [0.009, 0.011] \times [0.98, 1.02]$ and the training set is composed $P = 36$ distinct pairs. This training set is more scattered compared to the other test cases. For comparison purposes, the authors in [16] use 300 snapshots for the same test case. The variation in the initial distribution and electric energy is shown in Fig. 4.14.

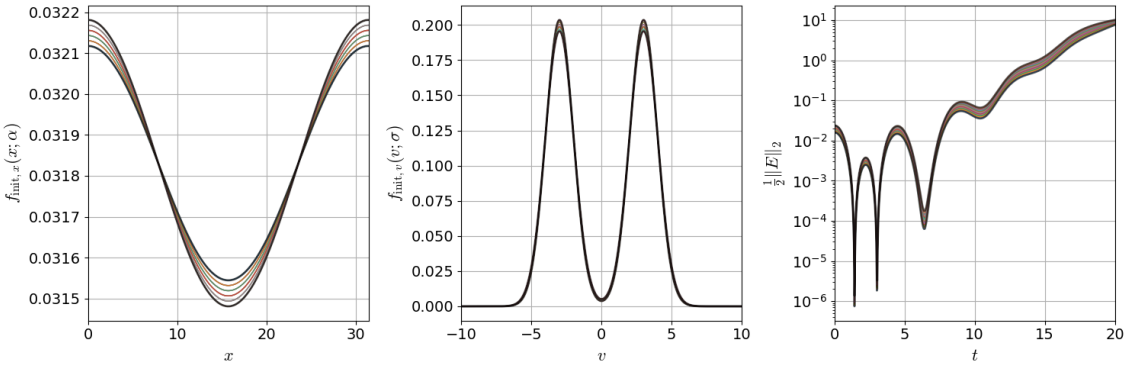


Figure 4.14: (Two stream instability) Initial distribution $f_{\mathrm{init},x}(x;\alpha)$ (left), $f_{\mathrm{init},v}(x;\sigma)$ (middle) and evolution of the electric energy $\frac{1}{2}\|E\|_2\left(\mathbf{x}(t;\mu);\mu\right)$ (right) for every $\mu \in \Gamma^{\mathrm{train}}$.

The intermediate reduced dimension and the final reduced dimensions are set to $M = 121$ and $K = 4$. The AE-HNN networks is trained with the architecture presented in Tab. 4.1. We

inspect relatives errors as a function of time in Fig. 4.15: $\text{err}_{X,t}^{\text{mean}}$ is about 2×10^{-3} and $\text{err}_{V,t}^{\text{mean}}$ is on the order of 1×10^{-2}.

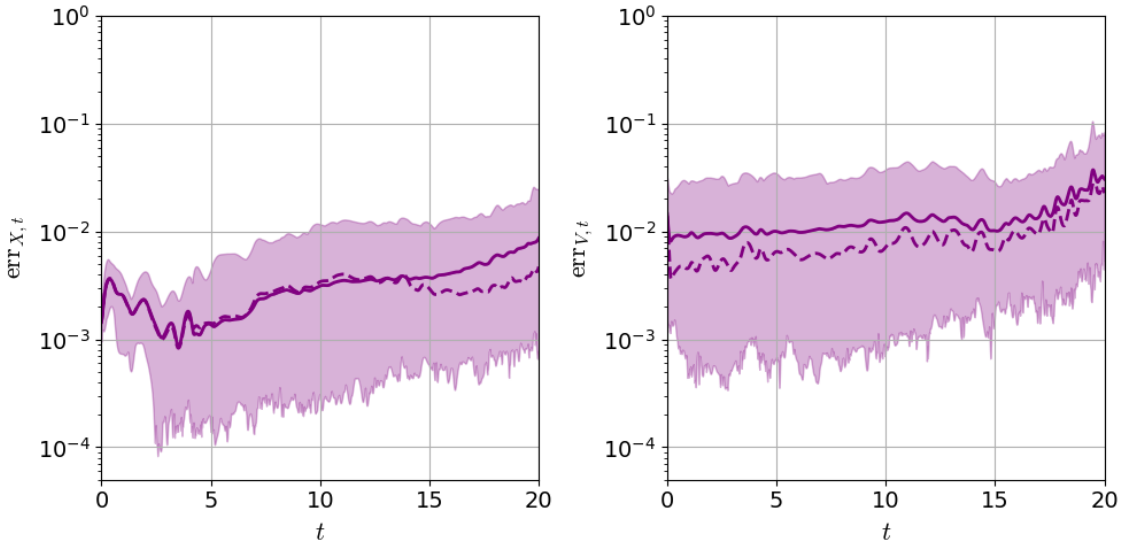


Figure 4.15: (Two stream instability) Mean error as a function of time $\text{err}_{X,t}^{\text{mean}}$ (left,solid line) and $\text{err}_{V,t}^{\text{mean}}$ (right, solid) for $\mu \in \Gamma^{\text{test}}$. Dashed lines are $\text{err}_{X,t}^{\text{mean}}, \text{err}_{V,t}^{\text{mean}}$ evaluated on the training set Γ^{train}, the envelopes represents minimal and maximal errors $\text{err}_{X,t}^{\text{min}}, \text{err}_{X,t}^{\text{max}}$ (left) and $\text{err}_{V,t}^{\text{min}}, \text{err}_{V,t}^{\text{max}}$.

Next, we observe the relative errors as a function of μ in Fig. 4.16. The error in positions, $\text{err}_{X,\mu}$, is approximately 5×10^{-3}, and the error in velocities, $\text{err}_{V,\mu}$, is around 2×10^{-2}. We notice only a slight increase in error as α increases, attributed to the sparsity of the training set. Overall, errors remain low, and the reduced dynamics are learned effectively.

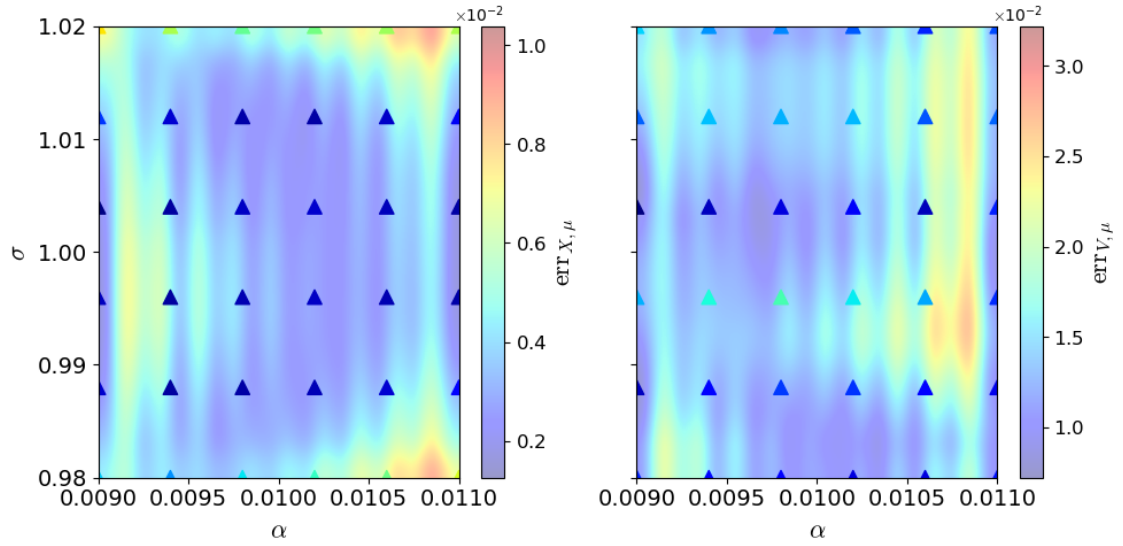


Figure 4.16: (Two stream instability) Errors as a function of the reduction parameters $\text{err}_{X,\mu}$ (left) and $\text{err}_{V,\mu}$ (right), triangular points represent the same error evaluated on the training set Γ^{train}.

Then, we plot the evolution of the distribution $f(t,x,v;\mu)$ of the FOM at different times in comparison with the ROM predicted distribution $f^{\text{pred}}(t,x,v;\mu)$ in Fig. 4.17. The dynamics are

correctly captured from the initial stream shearing to the development of a central vortex.

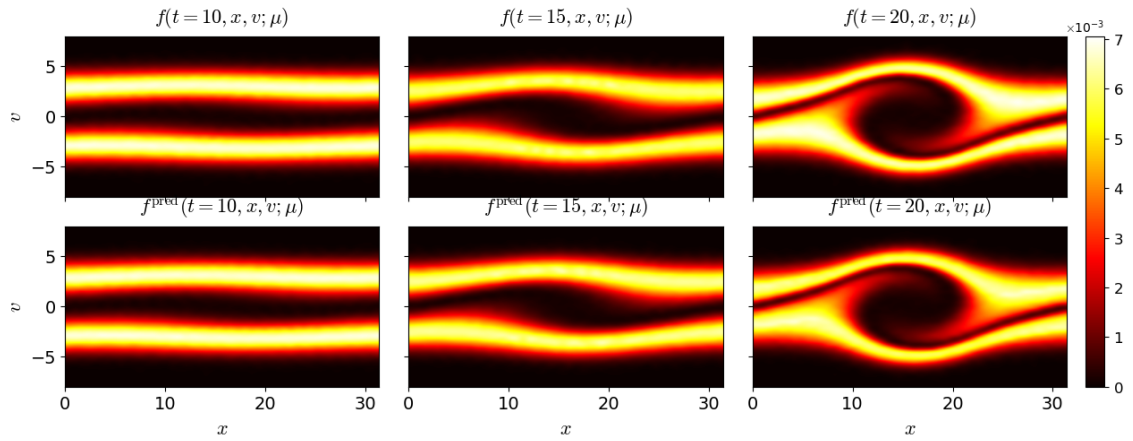
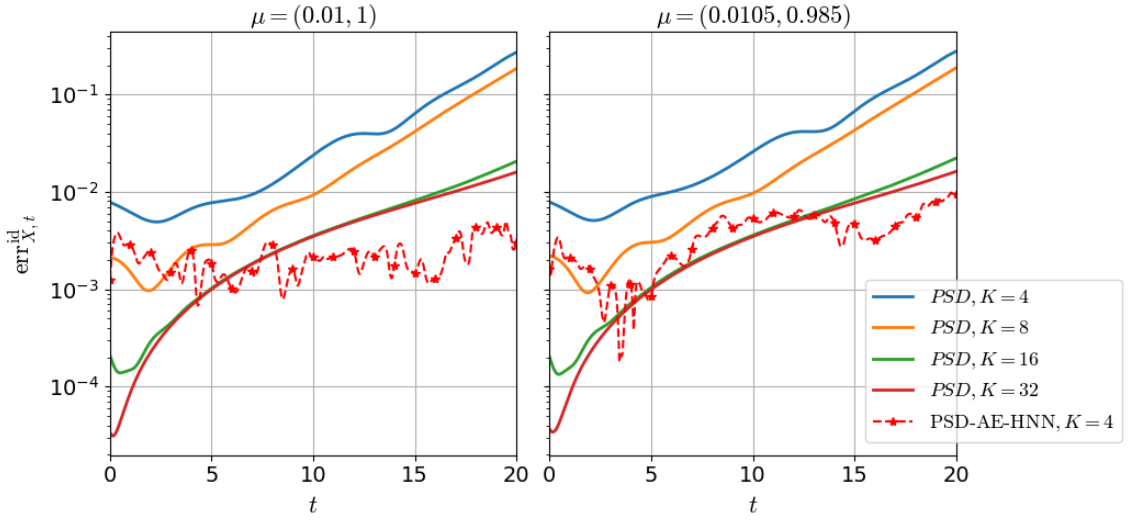
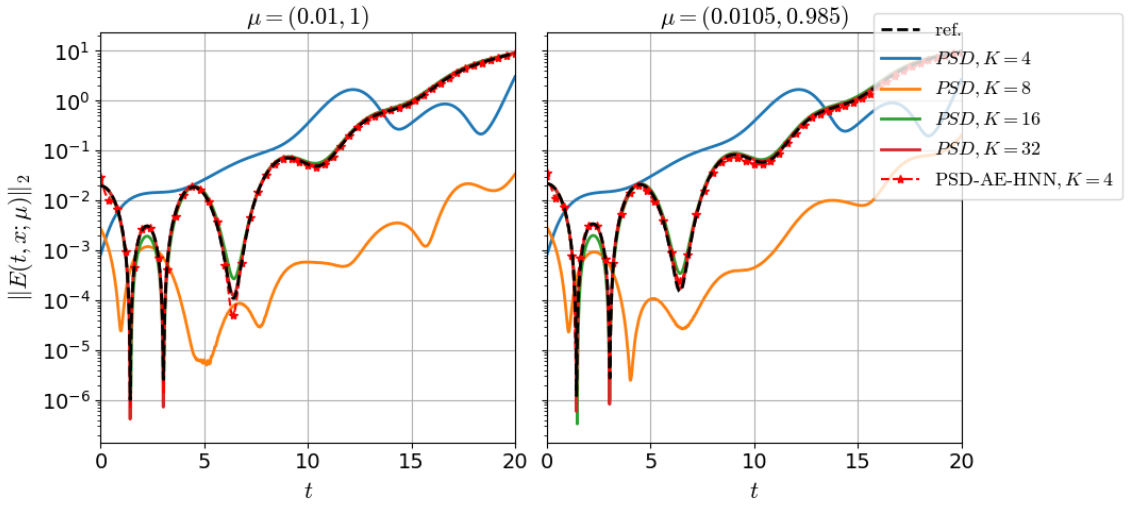


Figure 4.17: (Two stream instability) $f(t,x,v;\mu)$ (top) and $f^{\text{pred}}(t,x,v;\mu)$ (bottom) for $t \in \{10,15,20\}$ and $\mu = (0.0095, 0.99)$.

Eventually, we compare the method with $K = 4$ to the PSD with $K \in \{4,8,16,32\}$ in Fig. 4.18 for $\mu = (0.01, 1)$ (left) and $(0.0105, 0.985)$ (right). In Fig. 4.18b, we observe the electric energy $\frac{1}{2}\|E\|_2\left(\mathbf{x}(t;\mu);\mu\right)$, where its dynamics are well replicated with our method. We would need $K = 30$ modes with the PSD to obtain comparable results. In Fig. 4.18a, we study the relative errors $\text{err}^{\text{id}}_{X,t}$ and conclude that our method with a dimension of $K = 4$ achieves comparable results in terms of precision with the PSD and $K = 30$ modes.



(a) Errors $\mathrm{err}^{\mathrm{id}}_{X,t}$



(b) Electric energies $\frac{1}{2}\|E\|_2\left(\mathbf{x}(t;\mu);\mu\right)$

Figure 4.18: (Two stream instability) Comparison of the PSD reduced model against our method with $K = 4$, $\mathrm{err}^{\mathrm{id}}_{X,t} = \left\|\mathbf{x}^t_\mu - \hat{\mathbf{x}}^t_\mu\right\|_F / \left\|\mathbf{x}^t_\mu\right\|_F$ for a given $\mu = (0.01, 1)$ (left) and $\mu = (0.0105, 0.985)$ (right).

4.4.4 Computation gain

In this section, we evaluate the computation time for each model and test case from Secs. 4.4.1 to 4.4.3. To compare the performance of the reduced model, it is essential to identify a full model with equivalent accuracy. To achieve this, the number of particles will be varied. This approach will help us to assess that our method offers superior performance compared to a simple reduction in the number of particles. Therefore, we will not discuss the execution time of the PSD-only reduced model described in Eq. (4.15), as it is expected to result in a longer execution time without any hyper-reduction techniques.

In this study, we focus on a specific quantity of interest for each test case and evaluate the computation time for a single parameter, μ, across varying particle numbers, N. It is important to note that PIC simulations tend to exhibit noise when the particle count is low, leading to a non-monotonic relationship between the error and the particle number, N, with respect to the

quantity of interest.

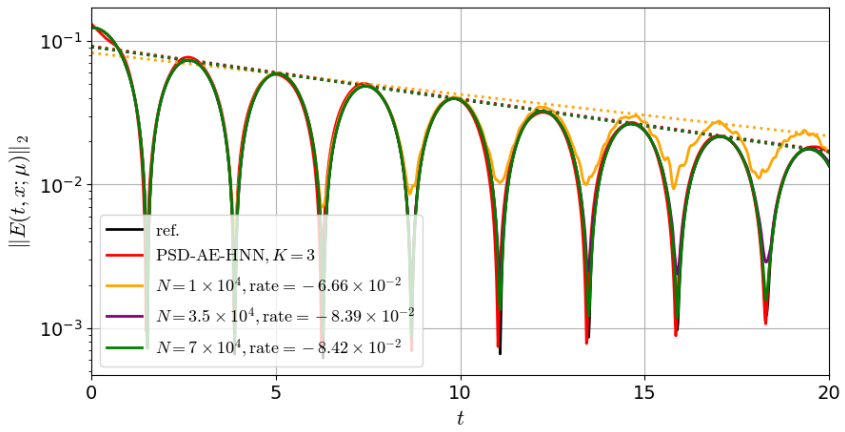
The code is implemented in Python, with the majority of operations utilizing the NumPy library. However, neural networks and training processes are implemented using the TensorFlow framework. A single AMD Ryzen 9 5900X CPU is employed for computation. It is important to note that this runtime test does not fully reflect the strengths of our method for two main reasons: (i) it is run on a CPU rather than a GPU, which significantly limits the efficiency of neural network evaluations; and (ii) it considers only a single initial condition, which prevents us from showcasing the neural network's ability to handle multiple inputs efficiently through vectorization.

For the linear Landau damping from Sec. 4.4.1, we set $\mu = (0.035, 0.84)$. We focus on the estimation of the damping rate to test our method. As shown in Fig. 4.19a, we estimate a PIC simulation with $N = 7 \times 10^4$ particles to be as precise as our PSD-AE-HNN method. From Fig. 4.20a, the PSD-AE-HNN is 4.63 times faster than the prior.

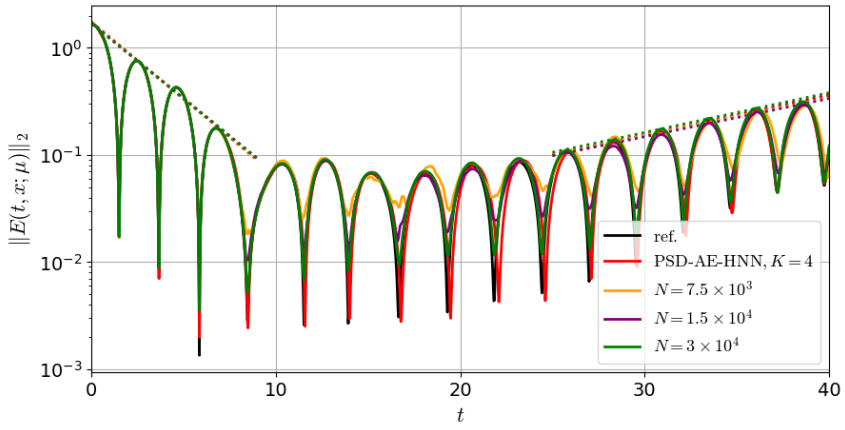
Then, we focus on the nonlinear Landau damping test case with $\mu = (0.465, 0.986)$. We consider the damping and growth rates as the quantities of interest. As shown in Fig. 4.19b, the equivalent PIC simulation requires $N = 3 \times 10^4$ particles and the speedup is about 1.95.

Finally, we are interested in the evolution of the electric energy for the two-stream instability. We set $\mu = (0.0105, 0.985)$ and observe that we need about $N = 3 \times 10^4$ particles for a comparable precision to our method in Fig. 4.19c giving an acceleration of 2.10.

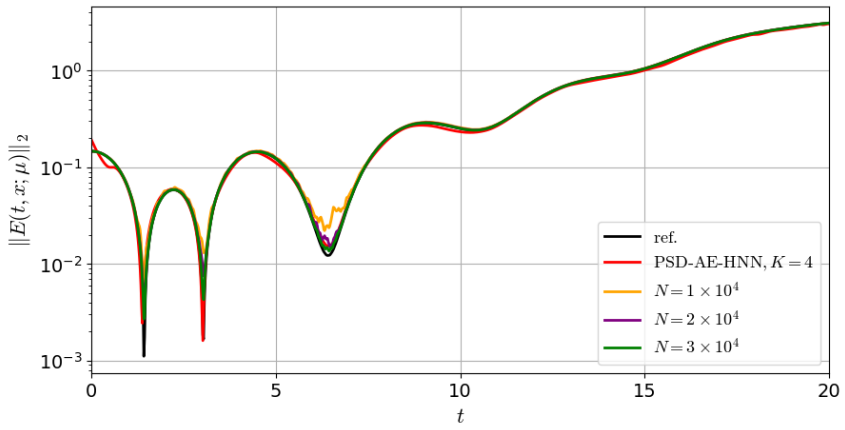
From a theoretical point of view and discarding the projection cost, one integration step requires $O(N + n_x^2)$ operations. From [32], our reduced model cost is about $O(\sum_{k=1}^{L} n^{(k-1)} n^{(k)})$ where $n^{(k)}$ is n-th layer width i.e. number of units of the HNN. In addition, as $n^{(k)}$ is of the order of K^2, the cost is $O(K^4)$ which makes it competitive as it does not depend on N nor n_x.



(a) (Linear Landau damping) $\mu = (0.035, 0.84)$. Dashed lines represent the linear damping.



(b) (Nonlinear Landau damping) $\mu = (0.465, 0.986)$. Dashed lines represent the linear damping and growth.



(c) (Two-stream instability) $\mu = (0.0105, 0.985)$.

Figure 4.19: Electric energy as a function of time for various N and the PSD-AE-HNN reduced model. Solid lines represent the energies.

	PIC				PSD-AE-HNN
N	1×10^5	7×10^4	3.5×10^4	1×10^4	
damping rate ($\times10^{-2}$)	-8.44	-8.42	-8.39	-6.66	-8.41
time (s)	25.05	11.40	6.13	2.00	2.46
speedup	0.46	1	1.86	5.70	4.63

(a) Linear Landau damping

	PIC				PSD-AE-HNN
N	1×10^5	3×10^4	1.5×10^4	7.5×10^3	
damping rate ($\times10^{-1}$)	-3.23	-3.23	-3.23	-3.26	-3.31
growth rate ($\times10^{-2}$)	8.55	8.55	8.22	7.89	8.60
time (s)	53.45	11.13	6.03	3.37	5.71
speedup	0.21	1	1.85	3.30	1.95

(b) Nonlinear Landau damping

	PIC				PSD-AE-HNN
N	1.5×10^5	3×10^4	2×10^4	1×10^4	
time (s)	59.30	5.34	3.68	2.08	2.54
speedup	0.09	1	1.45	2.57	2.10

(c) Two-stream instability

Figure 4.20: Computation time and numerical acceleration for each test case.

4.5 Conclusion

We have introduced a new Hamiltonian reduction method to reduce the number of particles in a particle-based discretization of the Vlasov-Poisson equation. This method uses a two-step mapping that combines Proper Symplectic Decomposition (PSD) for linear reduction with an autoencoder (AE) for additional nonlinear compression. PSD significantly reduces the dimensionality of the problem, while AE further compresses the data in a nonlinear manner. The reduced dynamics are then captured by a Hamiltonian neural network (HNN), ensuring the preservation of the Hamiltonian structure. The PSD-AE-HNN approach shows strong performance in linear and nonlinear test cases compared to the PSD method and offers good computational efficiency. Overall, the method is data-driven, non-intrusive, and highly adaptable.

One issue that arises is how to improve the quality of the approximation. In projection-based methods such as PSD, increasing the reduced dimension K leads to an increase in accuracy. There is no systematic process of this type for the proposed method. Instead, improvements can come from tuning the hyperparameters of the neural networks (e.g., the number of layers, size of the layers, activation functions) and refining the learning strategy.

The results should be extended to cover PIC simulations in two or three spatial and velocity dimensions. A substantial increase in network size should not be necessary for the AE-HNN component, as convolutional layers can be used. In contrast, the PSD may require further refinement. Additionally, future extensions could also involve integrating time-adaptive model reduction techniques as proposed in [15].

Acknowledgements. This research was funded in part by l'Agence Nationale de la Recherche (ANR), project ANR-21-CE46-0014 (Milk).

References

[1] E. Franck et al. “Reduced Particle in Cell method for the Vlasov-Poisson system using auto-encoder and Hamiltonian neural”. working paper or preprint. June 2025. URL: https://hal.science/hal-05116555.

[2] C. Birdsall and A. Langdon. *Plasma Physics via Computer Simulation*. Series in Plasma Physics and Fluid Dynamics. Taylor & Francis, 2018. ISBN: 9780750310253. DOI: 10.1201/9781315275048.

[3] P. L. Pritchett. “Particle-in-Cell Simulation of Plasmas— A Tutorial”. In: *Space Plasma Simulation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 1–24. ISBN: 978-3-540-36530-3. DOI: 10.1007/3-540-36530-3_1.

[4] H. R. Lewis. “Energy-Conserving Numerical Approximations for Vlasov Plasmas”. In: *J. Comput. Phys.* 1 (1970), pp. 136–141. DOI: 10.1016/0021-9991(70)90012-4.

[5] G. Chen, L. Chacón, and D. C. Barnes. “An energy- and charge-conserving, implicit, electrostatic particle-in-cell algorithm”. In: *J. Comput. Phys.* 230.18 (2011), pp. 7018–7036. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2011.05.031.

[6] J. E. Marsden and A. Weinstein. “The Hamiltonian structure of the Maxwell-Vlasov equations”. In: *Phys. D* 4.3 (1982), pp. 394–406. ISSN: 0167-2789,1872-8022. DOI: 10.1016/0167-2789(82)90043-4.

[7] Y. He et al. “Hamiltonian particle-in-cell methods for Vlasov-Maxwell equations”. In: *Phys. Plasmas* 23.9 (2016), p. 092108. ISSN: 1070-664X. DOI: 10.1063/1.4962573.

[8] H. Qin et al. “Canonical symplectic particle-in-cell method for long-term large-scale simulations of the Vlasov–Maxwell equations”. In: *Nucl. Fusion* 56.1 (2015), p. 014001. DOI: 10.1088/0029-5515/56/1/014001.

[9] J. Xiao et al. “Explicit high-order non-canonical symplectic particle-in-cell algorithms for Vlasov-Maxwell systems”. In: *Phys. Plasmas* 22.11 (2015), p. 112504. ISSN: 1070-664X. DOI: 10.1063/1.4935904.

[10] M. Kraus et al. “GEMPIC: geometric electromagnetic particle-in-cell methods”. In: *J. Plasma Phys.* 83.4 (2017). ISSN: 1469-7807. DOI: 10.1017/s002237781700040x.

[11] Y. Barsamian. “Pic-Vert : a particle-in-cell implementation for multi-core architectures”. Theses. Université de Strasbourg, Oct. 2018. URL: https://theses.hal.science/tel-02168151.

[12] T. Tyranowski and M. Kraus. “Symplectic model reduction methods for the Vlasov equation”. In: *Contrib. Plasma Phys.* 63.5-6 (2023), e202200046. ISSN: 0863-1042. DOI: 10.1002/ctpp.202200046.

[13] A. Nouy. “Low-rank tensor methods for model order reduction”. In: *Handbook of uncertainty quantification. Vol. 1, 2, 3*. Springer, Cham, 2017, pp. 857–882. ISBN: 978-3-319-12385-1. DOI: 10.1007/978-3-319-12385-1_21.

[14] C. Gräßle, M. Hinze, and S. Volkwein. “Model order reduction by proper orthogonal decomposition”. In: *Volume 2 Snapshot-Based Methods and Algorithms*. Ed. by P. Benner et al. Model Order Reduction. De Gruyter, 2021, pp. 47–96. ISBN: 9783110671490. DOI: 10.1515/9783110671490-002.

[15] J. S. Hesthaven, C. Pagliantini, and G. Rozza. “Reduced basis methods for time-dependent problems”. In: *Acta Numer.* 31 (2022), pp. 265–345. ISSN: 0962-4929,1474-0508. DOI: 10.1017/S0962492922000058.

[16] J. S. Hesthaven, C. Pagliantini, and N. Ripamonti. "Adaptive symplectic model order reduction of parametric particle-based Vlasov-Poisson equation". In: *Math. Comp.* 93.347 (2024), pp. 1153–1202. ISSN: 0025-5718,1088-6842. DOI: 10.1090/mcom/3885.

[17] S. Chaturantabut and D. C. Sorensen. "Nonlinear model reduction via discrete empirical interpolation". In: *SIAM J. Sci. Comput.* 32.5 (2010), pp. 2737–2764. ISSN: 1064-8275,1095-7197. DOI: 10.1137/090766498.

[18] G. Tissot et al. "Model reduction using Dynamic Mode Decomposition". en. In: *Comptes Rendus. Mécanique* 342.6-7 (2014), pp. 410–416. DOI: 10.1016/j.crme.2013.12.011.

[19] P. Schmid. "Dynamic mode decomposition of numerical and experimental data". In: *J. Fluid Mech.* 656 (2010), pp. 5–28. DOI: 10.1017/S0022112010001217.

[20] K. Kormann and E. Sonnendrücker. "Sparse grids for the Vlasov–Poisson equation". In: *Sparse Grids and Applications, 2014.* Ed. by Garcke, Jochen and Pflüger, Dirk. Lecture Notes in Computational Science and Engineering. Springer International Publishing, 2016, pp. 163–190. ISBN: 978-3-319-28262-6. DOI: 10.1007/978-3-319-28262-6_7.

[21] V. Ehrlacher and D. Lombardi. "A dynamical adaptive tensor method for the Vlasov-Poisson system". In: *J. Comput. Phys.* 339 (2017), pp. 285–306. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2017.03.015.

[22] L. Einkemmer and I. Joseph. "A mass, momentum, and energy conservative dynamical low-rank scheme for the Vlasov equation". In: *J. Comput. Phys.* 443 (2021), Paper No. 110495, 16. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2021.110495.

[23] L. Einkemmer and C. Lubich. "A low-rank projector-splitting integrator for the Vlasov-Poisson equation". In: *SIAM J. Sci. Comput.* 40.5 (2018), B1330–B1360. ISSN: 1064-8275,1095-7197. DOI: 10.1137/18M116383X.

[24] S. Bhattacharjee and K. Matous. "A nonlinear manifold-based reduced order model for multiscale analysis of heterogeneous hyperelastic materials". In: *J. Comput. Phys.* 313 (2016), pp. 635–653. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2016.01.040.

[25] B. Sonday et al. "Manifold learning techniques and model reduction applied to dissipative PDEs". In: *arXiv e-prints* (2010), arXiv–1011. DOI: 10.48550/arXiv.1011.5197.

[26] Y. Kim et al. "A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder". In: *J. Comput. Phys.* 451 (2022), Paper No. 110841, 29. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2021.110841.

[27] K. Lee and K. T. Carlberg. "Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders". In: *J. Comput. Phys.* 404 (2020), pp. 108973, 32. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2019.108973.

[28] E. Franck et al. "Hyperbolic reduced model for Vlasov-Poisson equation with Fokker-Planck collision". In: *ESAIM: ProcS* 77 (2024), pp. 213–228. DOI: 10.1051/proc/202477213.

[29] L. Peng and K. Mohseni. "Symplectic model reduction of Hamiltonian systems". In: *SIAM J. Sci. Comput.* 38.1 (2016), A1–A27. ISSN: 1064-8275,1095-7197. DOI: 10.1137/140978922.

[30] P. Buchfink, S. Glas, and B. Haasdonk. "Symplectic model reduction of Hamiltonian systems on nonlinear manifolds and approximation with weakly symplectic autoencoder". In: *SIAM J. Sci. Comput.* 45.2 (2023), A289–A311. ISSN: 1064-8275,1095-7197. DOI: 10.1137/21M1466657.

[31] S. Fresca and A. Manzoni. "POD-DL-ROM: enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition". In: *Comput. Methods Appl. Mech. Engrg.* 388 (2022), Paper No. 114181, 27. ISSN: 0045-7825,1879-2138. DOI: 10.1016/j.cma.2021.114181.

[32] R. Côte et al. "Hamiltonian reduction using a convolutional auto-encoder coupled to a Hamiltonian neural network". In: *Commun. Comput. Phys.* 37.2 (2025), pp. 315–352. ISSN: 1815-2406,1991-7120. DOI: 10.4208/cicp.OA-2023-0300.

[33] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[34] S. Greydanus, M. Dzamba, and J. Yosinski. "Hamiltonian neural networks". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates Inc., 2019. DOI: 10.48550/arXiv.1906.01563.

[35] F. Casas et al. "High-order Hamiltonian splitting for the Vlasov-Poisson equations". In: *Numer. Math.* 135.3 (2017), pp. 769–801. ISSN: 0029-599X,0945-3245. DOI: 10.1007/s00211-016-0816-z.

[36] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration*. Second. Vol. 31. Springer Series in Computational Mathematics. Structure-preserving algorithms for ordinary differential equations. Springer-Verlag, Berlin, 2006, pp. xviii+644. ISBN: 978-3-540-30663-4.

[37] J. Denavit and J. Walsh. "Nonrandom initializations of particle codes". In: *Plasma Phys. Control. Fusion* 6.6 (1981), pp. 209–223.

[38] J. M. Hammersley and D. C. Handscomb. "Random, Pseudorandom, and Quasirandom Numbers". In: *Monte Carlo Methods*. Dordrecht: Springer Netherlands, 1964, pp. 25–42. ISBN: 978-94-009-5819-7. DOI: 10.1007/978-94-009-5819-7_3.

[39] M. Kubat. "Neural networks: a comprehensive foundation by Simon Haykin, Macmillan, 1994, ISBN 0-02-352781-7." In: *Knowl. Eng. Rev.* 13.4 (1999), pp. 409–412. DOI: 10.1017/S0269888998214044.

[40] D. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv preprint arXiv:1412.6980* (2014). DOI: 10.48550/arXiv.1412.6980.

[41] G. Berge. "Landau damping in a plasma". Lectures given at The University of Bergen, Norway. 1969.

Chapter 5

Hyperbolic reduced model for Vlasov-Poisson equation with Fokker-Planck collision

This chapter has been published as a proceedings in *ESAIM: Proceedings and Surveys*, as outcome of a research project at the CEMRACS 2022 summer school on Transport in Physics, Biology and Urban Traffic.

E. Franck, I. Lannabi, Y. Nasseri, L. Navoret, G. Parasiliti Rantone, and G. Steimer. "Hyperbolic reduced model for Vlasov-Poisson equation with Fokker-Planck collision". In: *ESAIM: ProcS* 77 (2024), pp. 213–228. DOI: 10.1051/proc/202477213, entrytype=myverbose

Abstract

This paper proposes a reduced model to simulate the one-dimensional Vlasov-Poisson equation with the nonlinear Fokker-Planck operator. The model provides the space-time dynamics of a few macroscopic quantities constructed following the Reduced Order Method (ROM) in the velocity variable: the compression is thus applied to the semi-discretization of the Vlasov equation. To gain efficiency, a Discrete Empirical Interpolation Method (DEIM) is applied to the compressed nonlinear Fokker-Planck operator. The size of the resulting reduced model is chosen empirically according to the Knudsen number. Furthermore, we propose a correction to the reduced collision operator that ensures the reduced moments to satisfy an Euler-type system. Numerical simulations of the reduced model show that the model can capture the plasma dynamics in different collisional regimes and initial conditions at a low cost.

Chapter's contents

5.1 Introduction . . . 123
5.2 Reduced model in velocity for 1D Vlasov-Poisson-Fokker-Planck . . . 124
5.2.1 Vlasov-Poisson-Fokker-Planck model . . . 124
5.2.2 Semi-discretized model in velocity . . . 125
5.2.3 Reduced model . . . 127
5.3 Hyper-reduction and corrections of the reduced collision operator . . . 128
5.3.1 DEIM hyper-reduction . . . 128
5.3.2 Preservation of reduced moments . . . 128
5.3.3 Reduced Maxwellian distributions . . . 129
5.3.4 Reduced moment equations . . . 130
5.4 Numerical results . . . 131
5.4.1 Reduction for given parameters . . . 131
5.4.2 Preservation of Maxwellian distributions . . . 133
5.4.3 Generalization to other parameters . . . 134
5.5 Conclusion . . . 136
References . . . 136
A Numerical discretization details . . . 137
A.1 Discretization of the full model . . . 137
A.2 Discretization of the reduced model . . . 138
B Solution to the minimization problem . . . 138

5.1 Introduction

The Vlasov-Poisson-Fokker-Planck equation is a model for the transport of the distribution function of charged particles in the six-dimensional position-velocity phase space. The nonlinear Fokker-Planck operator describes the short-range binary interactions between charged particles, called collisions. The weight of collisions in the dynamics is measured by the dimensionless Knudsen number ε, the scaled mean-free path between collisions: a small Knudsen number corresponds to a collisional regime.

Simulations of such dynamics are very computationally demanding. Indeed, since phase space is of dimension six, simulations require a lot of memory and CPU resources. In addition, capturing collisional dynamics leads to numerical constraints on the time step and/or phase-space discretization. Indeed, the Fokker-Planck operator is a diffusive operator in the velocity variable: stability conditions for explicit numerical schemes have very stringent stability conditions when the Knudsen number ε is small. We refer to [2, 3] and references therein.

In order to avoid full model simulations, a classical strategy is to design reduced models that are valid in some parameter regimes. Typically, these reduced models provide the space-time dynamics of macroscopic quantities, which are integrated quantities over the velocity variables. For instance, the Euler system, satisfied by the density, the momentum, and the energy, can be obtained with the moment method and is valid in the strong collisional regime ($\varepsilon \ll 1$). The Navier-Stokes equations can be considered for lower Knudsen numbers with $\varepsilon < 10^{-2}$. Models with more moments have been designed in order to be valid at higher Knudsen number regimes.

Another possible approach is to consider reduced models based on data as proposed by the Reduced Order Modeling (ROM). The method consists in constructing an adapted basis in which

the solution can be well approximated with few components. The basis is computed through a Proper Orthogonal decomposition applied to samples of the unknown in the considered physical regime. The reduced model is then obtained with a projection Galerkin method. We refer to [4] for more references. This method has been applied for particle discretization of the Vlasov-Poisson system [5] with a reduction in both space and velocity. In order to efficiently tackle Eulerian discretization in the six-dimensional phase space, low-rank tensor bases have been proposed [6, 7], as well as their dynamical version [8].

Here we propose a mixed method where the ROM approach only compresses the dynamics in the velocity variable. Like moment models, the resulting reduced model provides the spatial dynamics of the reduced quantities. Therefore, the method provides a model independent of the spatial discretization of the computational domain. In order to provide a first assessment of the method, this study focuses on the one-dimensional dynamics.

The nonlinear collision operator leads to a reduced operator that would require to alternate between reduced and full dimensional data. To avoid these costly computations, we propose to use the Discrete Empirical Interpolation Method (DEIM) [9]. We also propose a correction of the reduced collision operator to ensure that it preserves reduced moments (like mass, momentum, and energy). Therefore, the reduced moments associated with the reduced data will satisfy an Euler-type equation.

This article is structured as follows. Sec. 5.2 introduces the Vlasov-Poisson-Fokker-Planck model, its semi-discretization in velocity, and the reduced model obtained after using the ROM approach in velocity. In Sec. 5.3, we recall the DEIM strategy to reduce the computational complexity for the nonlinear reduced collision operator. Then we present a correction to it such that it preserves the moments. Finally, in Sec. 5.4, we assess the capability of the reduced model in capturing nonlinear and linear dynamics. To this aim, we perform a Landau damping test case with different Knudsen numbers and perturbation amplitudes.

5.2 Reduced model in velocity for 1D Vlasov-Poisson-Fokker-Planck

5.2.1 Vlasov-Poisson-Fokker-Planck model

We consider the one-dimensional Vlasov-Poisson-Fokker-Planck dynamics that describes the evolution of a statistical distribution of charged particles in position-velocity phase-space, with long-range interactions through the self-induced electric field and short-range interactions. We define $f(t,x,v)$ as the distribution of particles at time $t \geqslant 0$ in the phase space $(x,v) \in [0,L]\times\mathbb{R}$ and the system reads:

$$\partial_t f(t,x,v) + v\,\partial_x f(t,x,v) + \partial_x \phi(t,x)\,\partial_v f(t,x,v) = \frac{1}{\varepsilon} Q(f)(t,x,v), \tag{5.1}$$

$$-\partial_x^2 \phi(t,x) = \frac{1}{L}\int_{[0,L]\times\mathbb{R}} f(t,y,v)\,dydv - \int_{\mathbb{R}} f(t,x,v)dv, \tag{5.2}$$

where $\phi(t,x)$ denotes the electric potential. The electric field is defined by the relation: $E(t,x) = -\partial_x\phi(t,x)$.

The right-hand side of the equation models collision interactions: $Q(f)$ models the short-range interactions, and the positive parameter ε represents the collision timescale. This collision operator is local in space and acts only on the velocity variable. Here, we consider a nonlinear Fokker-Planck collision operator given by:

$$Q(f) = \partial_v\Big((v-u_f)f + T_f\,\partial_v f\Big),$$

where $\rho_f(t,x)$, $u_f(t,x)$ and $T_f(t,x)$ denote the particle density (in space), the velocity, and the temperature, respectively, and are defined by:

$$\rho_f(t,x) = \int_{\mathbb{R}} f(t,x,v)\,dv, \quad \rho_f(t,x)u_f(t,x) = \int_{\mathbb{R}} v\, f(t,x,v)\,dv, \tag{5.3}$$

and

$$p_f(t,x) = \rho_f(t,x)T_f(t,x) = \int_{\mathbb{R}} (v-u_f(t,x))^2 f(t,x,v)dv. \tag{5.4}$$

The collision operator can be rewritten as:

$$Q(f) = T_f\,\partial_v\left(M_f\,\partial_v\left(\frac{f}{M_f}\right)\right), \tag{5.5}$$

where $M_f(v) = M_{\rho_f,u_f,T_f}(v)$ is the so-called Maxwellian distribution defined by:

$$M_{\rho,u,T}(v) = \frac{\rho}{\sqrt{2\pi T}} e^{-\frac{(v-u)^2}{2T}}. \tag{5.6}$$

This expression of Q shows that the equilibrium of the collision operator (satisfying $Q(f)=0$) is precisely the Maxwellian distribution.

To reduce the complexity of model (5.1)-(5.2), we want to project the velocity distribution function on a basis adapted to the desired physical parameters. For this purpose, we start by considering a finite-dimensional approximation of the dynamics.

5.2.2 Semi-discretized model in velocity

We perform a semi-discretization in velocity of the Vlasov-Poisson equations (5.1)-(5.2) using a finite difference method. We take a velocity interval $[-v_{\max}, v_{\max}]$ and an odd number of nodes N_v: $v_i = (i-(N_v-1)/2)\Delta v$ with $\Delta v = 2v_{\max}/(N_v-1)$ and $i \in \{0, N_v-1\}$. The unknown vector $\mathbf{f}(t,x) = (f(t,x,v_0),\ldots,f(t,x,v_{N_v-1}))$ satisfies the following semi-discretized Vlasov equation:

$$\partial_t \mathbf{f}(t,x) + A\,\partial_x \mathbf{f}(t,x) + (\partial_x\phi(t,x))\,D\,\mathbf{f}(t,x) = \frac{1}{\varepsilon} Q_{\Delta v}(\mathbf{f}(t,x)), \tag{5.7}$$

$$-\Delta\phi(t,x) = \frac{1}{L}\int_{[0,L]} \Delta v\,\mathbf{1}^T\mathbf{f}(t,x)\,dx - \Delta v\,\mathbf{1}^T\mathbf{f}(t,x), \tag{5.8}$$

where $A = \mathrm{Diag}(\mathbf{v})$ denotes the diagonal matrix of size N_v with the vector $\mathbf{v} = (v_0,\ldots,v_{N_v-1})$ on the diagonal. Matrix D refers to the matrix associated with the centered finite difference of the velocity derivative with Dirichlet boundary conditions:

$$D = \frac{1}{2\Delta v}\begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ -1 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & -1 & 0 \end{pmatrix}.$$

Symbol $\mathbf{1}$ denotes the vector $(1,\ldots,1)^T \in \mathbb{R}^{N_v}$. Eq. (5.7) is a hyperbolic system with a source term.

For the discretization of the collision operator, we consider the scheme proposed in [2] that is explicit, preserves the density, momentum, and energy, and dissipates entropy. Based on the following expression of the nonlinear Fokker-Planck collision operator,

$$Q(f) = T\,\partial_v\left(f\,\partial_v \log\left(\frac{f}{M_f}\right)\right),$$

the discretization can be written as:

$$Q_{\Delta v}(\mathbf{f})_j = \frac{\mathbf{f}_{j+\frac{1}{2}}\left(\log\left(\frac{\mathbf{f}_{j+1}}{(\mathbf{M_f})_{j+1}}\right) - \log\left(\frac{\mathbf{f}_j}{(\mathbf{M_f})_j}\right)\right)}{\Delta v^2} - \frac{\mathbf{f}_{j-\frac{1}{2}}\left(\log\left(\frac{\mathbf{f}_j}{(\mathbf{M_f})_j}\right) - \log\left(\frac{\mathbf{f}_{j-1}}{(\mathbf{M_f})_{j-1}}\right)\right)}{\Delta v^2},$$

where $\mathbf{f}_{\frac{1}{2}} = \mathbf{f}_{N_v-\frac{1}{2}} = 0$ and $\mathbf{f}_{j+\frac{1}{2}}$ is the entropic average:

$$\text{for } j = 1,\dots,N_v-2, \quad \mathbf{f}_{j+\frac{1}{2}} = \begin{cases} \frac{\mathbf{f}_{j+1}-\mathbf{f}_j}{\log(\mathbf{f}_{j+1})-\log(\mathbf{f}_j)}, & \text{if } \mathbf{f}_{j+1} \neq \mathbf{f}_j, \\ \mathbf{f}_j, & \text{otherwise .} \end{cases} \tag{5.9}$$

The discretization involves the discrete Maxwellian, which equals the evaluation of the Maxwellian at the velocity grid point, $\mathbf{M_f} = (M_{\rho_\mathbf{f},\tilde{u}_\mathbf{f},(\tilde{\rho}_\mathbf{f}\tilde{T}_\mathbf{f})/\rho_\mathbf{f}}(v_j))_j$, where $\tilde{\rho}_\mathbf{f}, \tilde{u}_\mathbf{f}, \tilde{T}_\mathbf{f}$ refer to the modified discrete moments:

$$\begin{aligned}
\tilde{\rho}_\mathbf{f} &= \sum_{j=1}^{N_v-2} \Delta v\,\mathbf{f}_{j+\frac{1}{2}}, \\
\tilde{\rho}_\mathbf{f}\tilde{u}_\mathbf{f} &= \sum_{j=1}^{N_v-2} \Delta v\,v_{j+\frac{1}{2}}\,\mathbf{f}_{j+\frac{1}{2}} + T_\mathbf{f}(\mathbf{f}_{N_v} - \mathbf{f}_1), \\
\tilde{\rho}_\mathbf{f}\tilde{T}_\mathbf{f} &= \sum_{j=1}^{N_v-2} \Delta v\,(v_{j+\frac{1}{2}} - \tilde{u}_\mathbf{f})^2\,\mathbf{f}_{j+\frac{1}{2}} + \tilde{\rho}_\mathbf{f}T_\mathbf{f}\big(\mathbf{f}_{N_v}(v_{N_v+\frac{1}{2}} - \tilde{u}_\mathbf{f}) + \mathbf{f}_1(v_{\frac{1}{2}} - \tilde{u}_\mathbf{f})\big),
\end{aligned}$$

and where $\rho_\mathbf{f}, u_\mathbf{f}, T_\mathbf{f}$ refer to the classical discrete moments associated with the discrete distribution $\mathbf{f}$ defined by:

$$\begin{aligned}
\rho_\mathbf{f} &= \Delta v\,\mathbf{1}^T\mathbf{f}, \\
\rho_\mathbf{f}\,u_\mathbf{f} &= \Delta v\,\mathbf{1}^T\mathrm{Diag}(\mathbf{v})\mathbf{f}, \\
\rho_\mathbf{f}\,u_\mathbf{f}^2 + \rho_\mathbf{f}T_\mathbf{f} &= \Delta v\,\mathbf{1}^T\mathrm{Diag}(\mathbf{v})^2\mathbf{f}.
\end{aligned}$$

We also define the moment matrix m:

$$m = \begin{bmatrix} m_\rho \\ m_{\rho u} \\ m_w \end{bmatrix} = \begin{bmatrix} \Delta v\,\mathbf{1}^T \\ \Delta v\,\mathbf{1}^T\mathrm{Diag}(\mathbf{v}) \\ \Delta v\,\mathbf{1}^T\mathrm{Diag}(\mathbf{v})^2 \end{bmatrix} \in M_{3,N_v}(\mathbb{R}).$$

As already said, the discrete moments $\rho_\mathbf{f}$ and $\rho_\mathbf{f}u_\mathbf{f}$ are conserved by the scheme. We also note that this discrete collision operator vanishes on the discrete Maxwellian with the modified mean velocity and temperature.

5.2.3 Reduced model

We now apply the ROM methodology. Using a Proper Orthogonal Decomposition (POD), we first generate a reduced basis of size $K \ll N_v$ from samples of the distribution function and define the linear decompression operator Φ, that we apply to a reduced vector $\hat{\boldsymbol{f}}$ to get the full vector $\boldsymbol{f}$, and the compression operator Φ^T, that we apply to the full vector $\boldsymbol{f}$ to get a reduced vector $\hat{\boldsymbol{f}}$:

$$\boldsymbol{f} \in \mathbb{R}^{N_v} \xrightarrow{\Phi^T} \hat{\boldsymbol{f}} \in \mathbb{R}^K \xrightarrow{\Phi} \tilde{\boldsymbol{f}} \in \mathbb{R}^{N_v}.$$

Then we perform a Galerkin projection to obtain the reduced model on the reduced quantity $\hat{\boldsymbol{f}}$.

Reduced basis and compression/decompression operators. We first consider N_s samples of the distribution in time and space obtained from the numerical simulations of the model of the previous section. Spatial discretization is done using finite volume schemes, and the time discretization is done using an explicit first-order scheme. We refer to Appendix Sec. A for more details. The sample matrix is then given by:

$$X = [\mathbf{f}_1(t_1, x_1), \dots, \mathbf{f}_n(t_n, x_n)] \in M_{N_v, N_s}(\mathbb{R}),$$

where $(\mathbf{f}_i)_i$ are obtained from N_s possible different initial data and different Knudsen numbers and (t_i, x_i) are chosen randomly. The linear decompression operator $\Phi \in M_{N_v,k}(\mathbb{R})$ is defined such that the compression-decompression error on the samples is minimized:

$$\min_{\substack{\Phi \in M_{N_v,K}(\mathbb{R}) \\ \Phi^T\Phi = \mathrm{Id}}} \left\| X - \Phi\Phi^T X \right\|_F^2, \tag{5.10}$$

where $\|A\|_F = \sqrt{\sum_{ij} A_{ij}^2}$ denotes the Frobenius norm for matrices. In practice, the columns of Φ are given by the first K eigenvectors of the matrix $X^T X \in M_{N_v}(\mathbb{R})$. These first K eigenvectors define the reduced basis.

Reduced model by Galerkin projection. Applying the Galerkin projection method to equations (5.7)-(5.8) consists in inserting the ansatz $\mathbf{f} = \Phi\hat{\boldsymbol{f}}$ into the equation, applying the projection Φ^T and then using the identity $\Phi^T\Phi = \mathrm{Id}$. We get the following model on the reduced variable $\hat{\boldsymbol{f}}$:

$$\partial_t \hat{\mathbf{f}}(t,x) + \hat{A}\,\partial_x \hat{\mathbf{f}}(t,x) + (\partial_x \phi(t,x))\,\hat{D}\,\hat{\mathbf{f}}(t,x) = \frac{1}{\varepsilon}\widehat{Q}_{\Delta v}(\hat{\mathbf{f}}), \tag{5.11}$$

$$-\Delta\phi(t,x) = \hat{m}_\rho \hat{\mathbf{f}}(t,x), \tag{5.12}$$

where matrices $\hat{A}, \hat{D}, m_\rho$ are respectively of size $K \times K$, $K \times K$ and $1 \times K$ and given by:

$$\hat{A} = \Phi^T \mathrm{Diag}(\mathbf{v})\Phi, \quad \hat{D} = \Phi^T D\,\Phi, \quad \hat{m}_\rho = \Delta v\,\mathbf{1}^T\Phi,$$

and the reduced collision operator is defined by:

$$\widehat{Q}_{\Delta v}(\hat{\mathbf{f}}) = \Phi^T Q_{\Delta v}(\Phi\hat{\mathbf{f}}(t,x)). \tag{5.13}$$

This system is a hyperbolic system of size K with two source terms: the first term is linear and comes from the transport in the velocity variable, while the other one is nonlinear and comes from the collision operator.

5.3 Hyper-reduction and corrections of the reduced collision operator

The nonlinear collision operator (5.13) requires to evaluate the collision operator $Q_{\Delta v}$ on the full vector $\Phi\hat{\mathbf{f}}$ of size N_v. Since this could be quite numerically expensive, several techniques called hyper-reduction have been developed. In the present work, we propose to use the classical Discrete Empirical Interpolation Method method [9].

The original discrete collision operator has been constructed to preserve mass, momentum, and energy. Such properties are crucial to capture the appropriate physical dynamics. We aim to ensure the same properties on the reduced collision operator. We thus define reduced moments and an associated corrected collision operator that preserves them.

5.3.1 DEIM hyper-reduction

Let us briefly describe the DEIM method. Using the Proper Orthogonal Decomposition on samples of the nonlinear collision operator $Q_{\Delta v}(\mathbf{f})$ (without reduction), the method first generates a basis made of K_Q vectors of size N_v: let Φ_Q the matrix of size $N_v \times K_Q$ that gathers these vectors. Then for any $\hat{\mathbf{f}}$, we are looking at coefficients $c(\hat{\mathbf{f}}) \in \mathbb{R}^{K_Q}$ such that:

$$Q_{\Delta v}(\Phi\hat{\mathbf{f}}) \approx \Phi_Q\, c(\hat{\mathbf{f}}), \tag{5.14}$$

The coefficients are chosen such that K_Q rows are indeed equalities (hence the name interpolation):

$$P_Q^T\, Q_{\Delta v}(\Phi\hat{\mathbf{f}}) = P_Q^T\, \Phi_Q\, c(\hat{\mathbf{f}}), \tag{5.15}$$

where P_Q is a selection matrix of size $N_v \times K_Q$, whose columns are made of canonical basis vectors. The selection of these K_Q rows is made iteratively with a greedy-like algorithm. In particular, P_Q is determined such that the matrix $P_Q^T\, \Phi_Q$ is invertible. Hence plugging back (5.15) into (5.14), we get the following approximation:

$$Q_{\Delta v}(\Phi\hat{\mathbf{f}}) \approx \Phi_Q(P_Q^T\Phi_Q)^{-1}P_Q^T Q_{\Delta v}(\Phi\hat{\mathbf{f}}),$$

and finally the operator $\Phi^T Q_{\Delta v}(\Phi\hat{\mathbf{f}})$ in expression (5.13) is replaced by

$$\widehat{Q}_{\Delta v}^{\text{DEIM}}(\hat{\mathbf{f}}) = \Phi^T\Phi_Q(P_Q^T\Phi_Q)^{-1}P_Q^T Q_{\Delta v}(\Phi\hat{\mathbf{f}}),$$

We refer to [9] for more details. The computational gain of this method comes from the fact that only the K_Q selected rows (among N_v) of $Q_{\Delta v}(\Phi\hat{\mathbf{f}})$ are actually computed.

5.3.2 Preservation of reduced moments

The reduced collision operator does not a priori satisfy the preservation of the reduced moments nor vanishes on discrete Maxwellians. We propose to slightly change the discrete collision operator to enforce these two properties, as already proposed in [10]. Similar corrections have been proposed in the context of spectral methods [11], discontinuous Galerkin methods [12] or discrete-velocity methods [13].

We first introduced the reduced moment operator as follows:

$$\hat{m} = \begin{bmatrix} \hat{m}_\rho \\ \hat{m}_{\rho u} \\ \hat{m}_w \end{bmatrix} = \begin{bmatrix} \Delta v\, \mathbf{1}^T \\ \Delta v\, \mathbf{1}^T\Phi(\Phi^T\text{Diag}(\mathbf{v})\Phi) \\ \frac{1}{2}\Delta v\, \mathbf{1}^T\Phi(\Phi^T\text{Diag}(\mathbf{v})\Phi)^2 \end{bmatrix} \in M_{3,K}(\mathbb{R}). \tag{5.16}$$

Note that this definition differs from the moment operator $m\Phi$, which would be the natural definition since it first decompresses up to the full size and then applies the classical moment

operator. Instead, the considered reduced moment operator first multiplies the reduced discrete distribution $\hat{\mathbf{f}} \in \mathbb{R}^K$ by the reduced velocity operator $(\Phi^T \mathrm{Diag}(\mathbf{v})\Phi)$, then decompresses and finally integrates. With such a definition, we have the relations:

$$\hat{m}_{\rho u} = \hat{m}_\rho \hat{A}, \quad \hat{m}_w = \frac{1}{2}\hat{m}_{\rho u}\hat{A}. \tag{5.17}$$

which will be useful in obtaining a good structure of the reduced moments equation.

We modify the reduced collision operator $\widehat{Q}_{\Delta v}$ by taking the closest vector $\widehat{Q}$ such that $\hat{m}\widehat{Q} = 0$. We thus define the corrected collision operator $\widehat{Q}^c_{\Delta v}$ as the solution to the minimizing problem:

$$\widehat{Q}^c_{\Delta v}(\Phi\hat{\mathbf{f}}) = \underset{\widehat{Q}\in\mathbb{R}^K \text{ s.t. } \hat{m}\widehat{Q}=0}{\operatorname{argmin}} \|\Phi^T\widehat{Q} - Q_{\Delta v}(\Phi\hat{\mathbf{f}})\|^2.$$

The solution to this minimization problem is given by:

$$\widehat{Q}^c_{\Delta v}(\Phi\hat{\mathbf{f}}) = \Phi^T Q_{\Delta v}(\Phi\hat{\mathbf{f}}) - \hat{m}^T(\hat{m}\hat{m}^T)^{-1}\hat{m}\Phi^T Q_{\Delta v}(\Phi\hat{\mathbf{f}}). \tag{5.18}$$

The proof can be found in Sec. B. Naturally, we can apply the DEIM strategy presented in the previous section to this corrected collision operator. Namely,

$$\widehat{Q}^{\mathrm{cDEIM}}_{\Delta v}(\Phi\hat{\mathbf{f}}) = \widehat{Q}^{\mathrm{DEIM}}_{\Delta v}(\hat{\mathbf{f}}) - \hat{m}^T(\hat{m}\hat{m}^T)^{-1}\hat{m}\widehat{Q}^{\mathrm{DEIM}}_{\Delta v}(\hat{\mathbf{f}}). \tag{5.19}$$

5.3.3 Reduced Maxwellian distributions

Given a set of moments $\mu \in \mathbb{R}^3$, we denote $\mathbf{M}(\mu) = \mathbf{M}_{\rho,u,T}$ the discrete Maxwellian distribution, where ρ, u, T are defined such that $\mu = (\rho, \rho u, \rho u^2/2 + \rho T/2)^T$. We then defined the reduced Maxwellian $\widehat{\mathbf{M}}(\mu)$ associated with reduced moments μ as the closest to the full Maxwellian $\mathbf{M}(\mu)$ after decompression:

$$\widehat{\mathbf{M}}(\mu) = \underset{\hat{\mathbf{f}}\in\mathbb{R}^K \text{ s.t. } \hat{m}\hat{\mathbf{f}}=\mu}{\arg\min} \|\Phi\hat{\mathbf{f}} - \mathbf{M}(\mu)\|^2. \tag{5.20}$$

As in the previous section, the solution to this minimization problem denoted is given by:

$$\widehat{\mathbf{M}}(\mu) = \Phi^T\mathbf{M}(\mu) + \hat{m}^T(\hat{m}\hat{m}^T)^{-1}\left(\mu - \hat{m}\Phi^T\mathbf{M}(\mu)\right).$$

If μ is associated with the reduced moments of $\hat{\mathbf{f}}$, then we get the following expression:

$$\widehat{\mathbf{M}}(\hat{m}\hat{\mathbf{f}}) = \Phi^T\mathbf{M}(\hat{m}\hat{\mathbf{f}}) + \hat{m}^T(\hat{m}\hat{m}^T)^{-1}\left(\hat{m}\hat{\mathbf{f}} - \hat{m}\Phi^T\mathbf{M}(\hat{m}\hat{\mathbf{f}})\right). \tag{5.21}$$

The reduced collision operator is not guaranteed to vanish for reduced Maxwellians. We thus propose to consider the following second correction:

$$\widehat{Q}^{c2}_{\Delta v}(\Phi\hat{\mathbf{f}}) = \left(\frac{\|\hat{\mathbf{f}} - \widehat{\mathbf{M}}(\hat{m}\hat{\mathbf{f}})\|}{\|\hat{\mathbf{f}} - \widehat{\mathbf{M}}(\hat{m}\hat{\mathbf{f}})\| + \delta}\right)\widehat{Q}^c_{\Delta v}(\Phi\hat{\mathbf{f}}), \tag{5.22}$$

with $\delta > 0$.

As for the collision operator, we can apply the DEIM method to avoid the computation of $\mathbf{M}(\mu)$ appearing in (5.21) and which is evaluated in the full space. We consider samples of the Maxwellian distribution and construct a reduced basis of size K_M and gather it into the matrix Φ_M of dimension $N_v \times K_M$. Then we consider the following reduced Maxwellian:

$$\begin{aligned}\widehat{\mathbf{M}}^{\mathrm{DEIM}}(\hat{m}\hat{\mathbf{f}}) = \Phi^T &\left(\Phi_M(P_M^T\Phi_M)^{-1})P_M^T\mathbf{M}(\hat{m}\hat{\mathbf{f}})\right) \\ &+ \hat{m}^T(\hat{m}\hat{m}^T)^{-1}\left(\hat{m}\hat{\mathbf{f}} - \hat{m}\Phi^T\left(\Phi_M(P_M^T\Phi_M)^{-1}P_M^T\mathbf{M}(\hat{m}\hat{\mathbf{f}})\right)\right),\end{aligned}$$

where P_M is a matrix of size $N_v \times K_M$ which selects K_M rows of $\mathbf{M}$. Then combining the above definition of the reduced Maxwellian with the second corrected reduced collisional operator (5.23), the reduced collision operator writes:

$$\widehat{Q}^{\text{c2DEIM}}_{\Delta v}(\Phi\hat{\mathbf{f}}) = \left(\frac{\|\hat{\mathbf{f}} - \widehat{\mathbf{M}}^{\text{DEIM}}(\hat{m}\hat{\mathbf{f}})\|}{\|\hat{\mathbf{f}} - \widehat{\mathbf{M}}^{\text{DEIM}}(\hat{m}\hat{\mathbf{f}})\| + \delta} \right) \widehat{Q}^{\text{cDEIM}}_{\Delta v}(\Phi\hat{\mathbf{f}}). \tag{5.23}$$

5.3.4 Reduced moment equations

In this part, we show that the equation on the reduced moments is an Euler-type system. Indeed, let us consider the semi-discretized reduced equation as

$$\partial_t \hat{\mathbf{f}} + \hat{A}\,\partial_x \hat{\mathbf{f}} + (\partial_x \phi)\,\hat{D}\,\hat{\mathbf{f}} = \frac{1}{\varepsilon}\widehat{Q}^{c}_{\Delta v}(\Phi\hat{\mathbf{f}}), \tag{5.24}$$

where $\widehat{Q}^{(2)}$ is the corrected reduced collision kernel defined in (5.23). As this operator has vanishing reduced moments, we have:

$$\partial_t(\hat{m}\hat{\mathbf{f}}) + \partial_x(\hat{m}\hat{A}\hat{\mathbf{f}}) + (\partial_x\phi)\,\hat{m}\hat{D}\,\hat{\mathbf{f}} = 0,$$

which is equivalent to the system of equations:

$$\begin{cases} \partial_t(\hat{m}_\rho\hat{\mathbf{f}}) + \partial_x(\hat{m}_\rho\hat{A}\hat{\mathbf{f}}) + (\partial_x\phi)\,\hat{m}_\rho\hat{D}\,\hat{\mathbf{f}} = 0, \\ \partial_t(\hat{m}_{\rho u}\hat{\mathbf{f}}) + \partial_x(\hat{m}_{\rho u}\hat{A}\hat{\mathbf{f}}) + (\partial_x\phi)\,\hat{m}_{\rho u}\hat{D}\,\hat{\mathbf{f}} = 0, \\ \partial_t(\hat{m}_w\hat{\mathbf{f}}) + \partial_x(\hat{m}_w\hat{A}\hat{\mathbf{f}}) + (\partial_x\phi)\,\hat{m}_w\hat{D}\,\hat{\mathbf{f}} = 0, \end{cases} \tag{5.25}$$

Next, thanks to relations (5.17), we get the following property:

Property 5.3.1. *Denoting $(\hat{\rho}, \hat{\rho}\hat{u}, \hat{w})^T = \hat{m}\hat{\mathbf{f}}$, system* (5.25) *becomes the following Euler-Poisson type system:*

$$\begin{cases} \partial_t\hat{\rho} + \partial_x(\hat{\rho}\hat{u}) = -(\partial_x\phi)\,\hat{m}_\rho\hat{D}\,\hat{\mathbf{f}}, \\ \partial_t(\hat{\rho}\hat{u}) + \partial_x(\hat{\rho}\hat{u}^2 + \hat{p}) = -(\partial_x\phi)\,\hat{m}_{\rho u}\hat{D}\,\hat{\mathbf{f}}, \\ \partial_t\hat{w} + \partial_x(\hat{w}\hat{u} + \hat{p}\hat{u} + \hat{q}) = -(\partial_x\phi)\,\hat{m}_w\hat{D}\,\hat{\mathbf{f}}, \end{cases} \tag{5.26}$$

where $\hat{p}$ is such that $\hat{w} = \hat{\rho}\hat{u}^2/2 + \hat{p}/2$ and

$$\hat{q} = \frac{1}{2}\Delta v\ \mathbf{1}^T\Phi\big(\hat{A} - \hat{u}\,\mathrm{Id}\big)^3\hat{\mathbf{f}}. \tag{5.27}$$

Proof. Indeed, using relations (5.17), the fluxes of the reduced density and momentum write:

$$\begin{aligned} &\hat{m}_\rho\hat{A}\hat{\mathbf{f}} = \hat{m}_{\rho u}\hat{A}\hat{\mathbf{f}} = \hat{\rho}\hat{u}, \\ &\hat{m}_{\rho u}\hat{A}\hat{\mathbf{f}} = 2\,\hat{m}_w\hat{\mathbf{f}} = 2\hat{w} = \hat{\rho}\hat{u}^2 + \hat{p}. \end{aligned}$$

Then the flux of the reduced energy is given by:

$$\begin{aligned} \hat{m}_w\hat{A}\widehat{M} &= \frac{1}{2}\Delta v\ \mathbf{1}^T\Phi\hat{A}^3\widehat{M} \\ &= \frac{1}{2}\Delta v\ \mathbf{1}^T\Phi(\hat{A} - \hat{u}\,\mathrm{Id} + \hat{u}\,\mathrm{Id})^3\widehat{M} \\ &= \frac{1}{2}\Delta v\ \mathbf{1}^T\Phi\left[(\hat{A} - \hat{u}\,\mathrm{Id})^3 + 3\hat{u}(\hat{A} - \hat{u}\,\mathrm{Id})^2 + 3\hat{u}^2(\hat{A} - \hat{u}\,\mathrm{Id}) + \hat{u}^3\,\mathrm{Id}\right]\widehat{M} \\ &= \hat{q} + \frac{3}{2}\hat{u}\hat{p} + 0 + \frac{1}{2}\hat{\rho}\hat{u}^3 \\ &= \hat{q} + \hat{u}\hat{p} + \hat{w}\hat{u}, \end{aligned}$$

where, in the second last equality, we used the definition of $\hat{q}$ given in (5.27) and the identities:

$$\begin{aligned}\Delta v\ \mathbf{1}^T\Phi(\hat{A}-\hat{u}\,\mathrm{Id})\hat{\mathbf{f}} &= \hat{\rho}\hat{u}-\hat{\rho}\hat{u}=0,\\ \Delta v\ \mathbf{1}^T\Phi(\hat{A}-\hat{u}\mathrm{Id})^2\hat{\mathbf{f}} &= \Delta v\ \mathbf{1}^T\Phi(\hat{A}^2-2\hat{u}\hat{A}+\hat{u}^2\mathrm{Id})\widehat{M}\\ &= 2\hat{w}-2\hat{\rho}\hat{u}^2+\hat{\rho}\hat{u}^2\\ &= \hat{p}.\end{aligned}$$

Inserting these expressions of fluxes into (5.25), we obtain (5.26). □

Consequently, the particular choice of the reduced moments (5.16) ensures the recovery of an Euler-Poisson type system. Note, however, that the heat flux is a priori non-zero, and there is a priori no conservation of mass.

5.4 Numerical results

We test the reduced order model developed in this paper on the Landau damping test case as considered in [14]. This test consists in considering the following initial distribution:

$$f(0,x,v)=\frac{1}{\sqrt{2\pi}}\exp(-v^2/2)\,(1+\alpha\cos(kx)),\quad x\in\left[0,2\pi/k\right],\ v\in[-v_{\max},v_{\max}],$$

with $k=0.5$, $v_{\max}=6$. We set $\alpha\in[0.01,0.2]$ and $\varepsilon\in[0.01,10]$ in order to assess the method on both nonlinear Landau damping (large α) and collisional regime (small ε). The Vlasov-Poisson-Fokker-Planck model is discretized with $N_x=N_v=128$.

5.4.1 Reduction for given parameters

In this section, we consider the following parameters: $(\varepsilon,\alpha)\in\{0.01,0.1,1,10\}\times\{0.01,0.1,0.2\}$. For each pair (ε,α), the reduced model is built on $N_s=200$ uniform samples. Then we will discuss the reduced dimensions K, K_M, and K_Q needed to recover the right dynamics. The final time is taken equal to $T=40$ in this test and $\delta=1\times10^{-10}$.

Fig. 5.1 shows the singular value distributions corresponding to the samples matrices $N_s\times N_s$ of the distribution function $\mathbf{f}$, the Maxwellian $M(mf)$ and the collision operator $Q(f)$ for the different choices of (ε,α). We remark that the singular values associated with the Maxwellian (in green dots) decrease fast down to 10^{-11} regardless (ε,α). In red dots, the singular values of the distribution function decrease slowly for large α and are similar to those of the Maxwellians for small ε. Similarly, the singular values for the collision operator (in blue dots) increase with both α and ε. For instance, with $\varepsilon=10$, they reach a plateau of 10^{-5} for $\alpha=0.01$, 10^{-3} for $\alpha=0.1$ and 10^{-2} for $\alpha=0.2$. Consequently, the model becomes stiffer with larger values of α and ε. Moreover, vertical lines in Fig. 5.1 represent the threshold values in order to obtain correct damping results: only eigenvectors with associated singular values below the threshold are considered in the reduced model. These values are reported in Tab. 5.1.

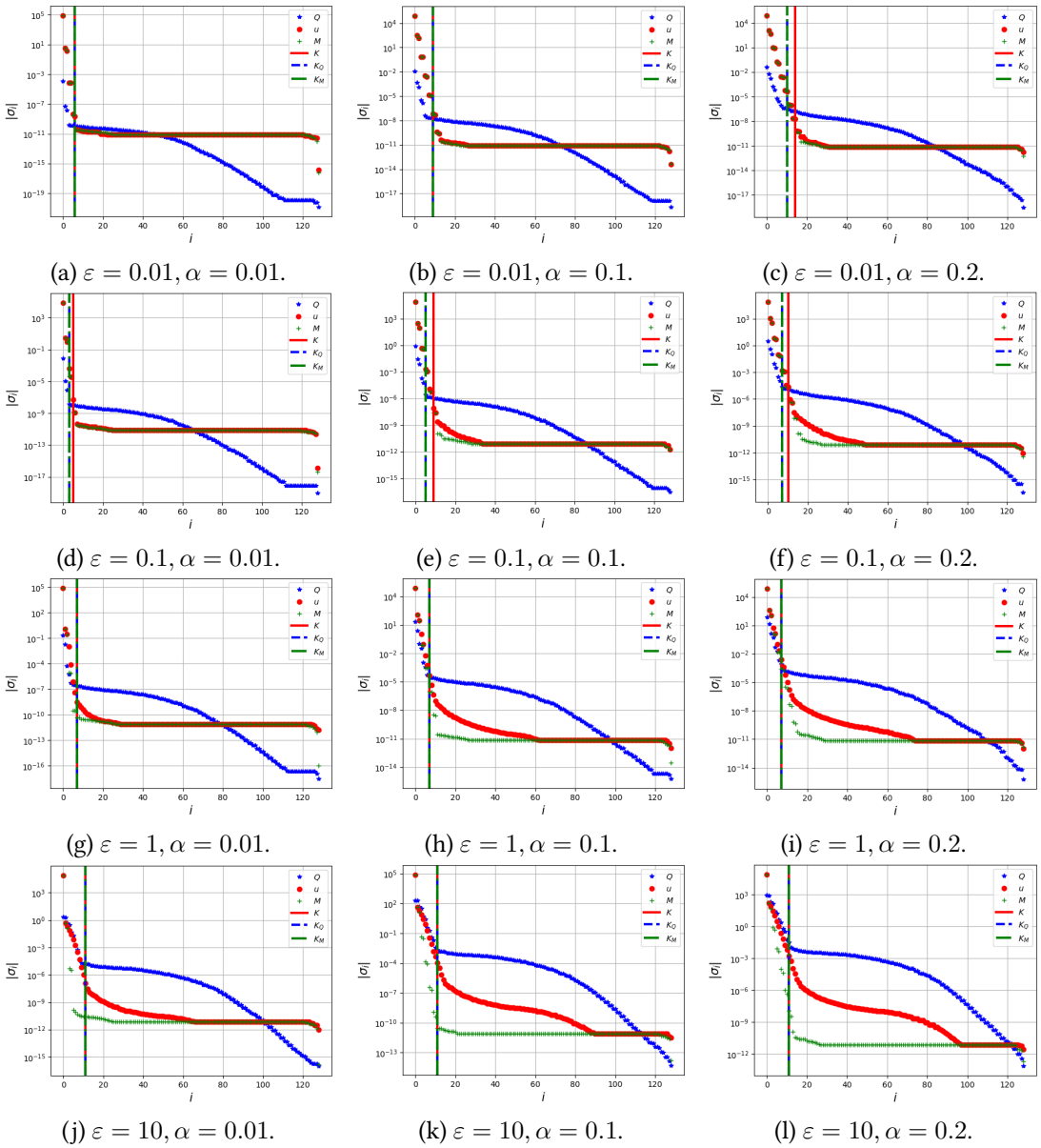
Fig. 5.2 shows the evolution of the L^2 norm of the electric field:

$$|E(t,.)|_{L^2}=\sqrt{\int_0^{\frac{2\pi}{k}}(-\partial_x\phi(t,x))^2\,dx},$$

for some pairs (ε,α) obtained with the reduced model, in log scale, and obtained with both the reduced and the kinetic models. Comparing the results obtained with the reduced model to those obtained with the kinetic model, we observe that a good damping rate is recovered using

the reduced model. Also, we check that the DEIM and corrected moments do not deteriorate the results.

In conclusion, the reduced model provides good results for various ranges of parameters ε and α and enables a drastic reduction of unknowns.



(a) $\varepsilon = 0.01, \alpha = 0.01$. (b) $\varepsilon = 0.01, \alpha = 0.1$. (c) $\varepsilon = 0.01, \alpha = 0.2$.

(d) $\varepsilon = 0.1, \alpha = 0.01$. (e) $\varepsilon = 0.1, \alpha = 0.1$. (f) $\varepsilon = 0.1, \alpha = 0.2$.

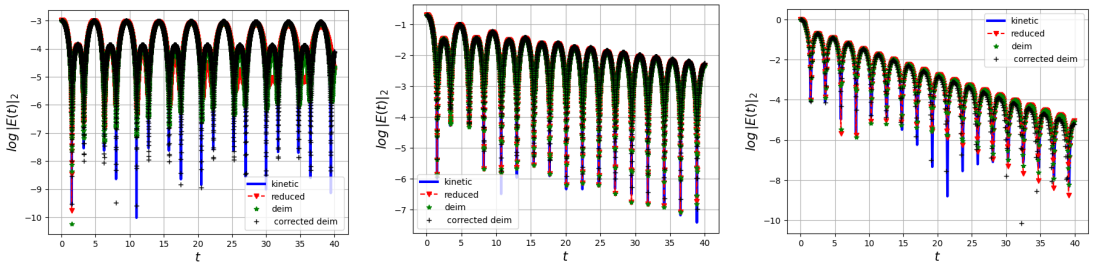
(g) $\varepsilon = 1, \alpha = 0.01$. (h) $\varepsilon = 1, \alpha = 0.1$. (i) $\varepsilon = 1, \alpha = 0.2$.

(j) $\varepsilon = 10, \alpha = 0.01$. (k) $\varepsilon = 10, \alpha = 0.1$. (l) $\varepsilon = 10, \alpha = 0.2$.

Figure 5.1: Singular values $(\sigma_i)_i$ of the samples matrix for the distribution function f, the Maxwellian M and the collision operator Q.

ε \ α	0.01	0.1	0.2
0.01	$(K, K_Q, K_M) = (6, 6, 6)$	$(K, K_Q, K_M) = (9, 9, 9)$	$(K, K_Q, K_M) = (15, 10, 10)$
0.1	$(K, K_Q, K_M) = (5, 3, 3)$	$(K, K_Q, K_M) = (9, 5, 5)$	$(K, K_Q, K_M) = (10, 7, 7)$
1	$(K, K_Q, K_M) = (7, 7, 7)$	$(K, K_Q, K_M) = (7, 7, 7)$	$(K, K_Q, K_M) = (7, 7, 7)$
10	$(K, K_Q, K_M) = (11, 11, 11)$	$(K, K_Q, K_M) = (11, 11, 11)$	$(K, K_Q, K_M) = (11, 11, 11)$

Table 5.1: Number of eigenvectors kept for the solution space (K), the Maxwellian (K_M) and the collision operator (K_Q).



(a) $\varepsilon = \alpha = 0.01, \text{CFL} = 0.0025$. (b) $\varepsilon = 1, \alpha = 0.1, \text{CFL} = 0.1$. (c) $\varepsilon = 10, \alpha = 0.2, \text{CFL} = 0.5$.

Figure 5.2: Logarithm of the L^2 norm of the electric field $|E(t, .)|_{L^2}$ as a function of time for the reference solution, reduced model, with DEIM and DEIM with corrected moments.

5.4.2 Preservation of Maxwellian distributions

In Sec. 5.3.3, we proposed a second correction of the reduced collisional operator (see Eq. (5.23)), which forces it to vanish for reduced Maxwellians thanks to a prefactor. The correction is parameterized by $\delta \geqslant 0$. This correction is aimed at preserving Maxwellian distributions in time, up to the compression-decompression error. To assess the effectiveness of this correction, we consider an homogeneous test case with a Maxwellian initial distribution:

$$\partial_t f(t, x, v) = \frac{1}{\varepsilon} Q(f)(t, x, v), \quad x \in \left[0, 2\pi/k\right], \ v \in [-v_{\max}, v_{\max}],$$
$$f(0, x, v) = \frac{1}{\sqrt{2\pi}} e^{-\frac{v^2}{2}},$$

with the parameter $\varepsilon \in \{0.01, 0.1, 1, 10\}$ on the time interval $[0, T]$ with $T = 20$. A specific reduced model is built for each ε using $N_s = 200$ uniform samples and reduced dimensions $(K, K_Q, K_M) = (11, 11, 11)$. We then compare the solution obtained using the reduced models at time T with the exact solution $f(T, x, v) = f(0, x, v)$, when using different values of the parameter δ involved in the correction: from $\delta = 0$, which corresponds to no correction, to $\delta = 1 \times 10^{-2}$. In Fig. 5.3, we observe the L^2 error for the four different reduced models built for each ε. While for $\varepsilon = 10$, the correction improves the result by a factor 5 only, the error can be reduced by a factor 1000 when $\varepsilon = 1$.

For a smaller $\varepsilon = 0.1$, the second correction is even more important as the reduced model without correction ($\delta = 0$) leads to a large error of order 5×10^{-1}, while the correction reduces the error to about 2×10^{-8} for δ larger than 1×10^{-7}. Additionally, for $\varepsilon = 0.01$, when δ values range between 0 and 10^{-5}, the error remains constant and relatively high. Conversely, the error improves as delta exceeds 10^{-4}.

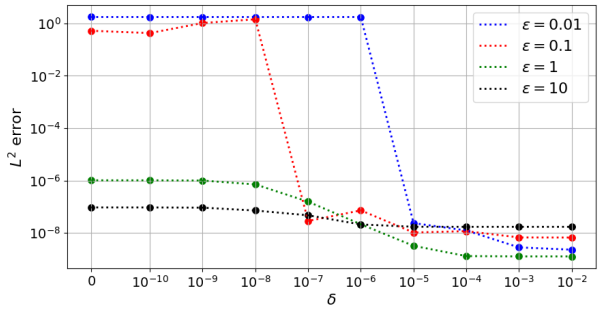
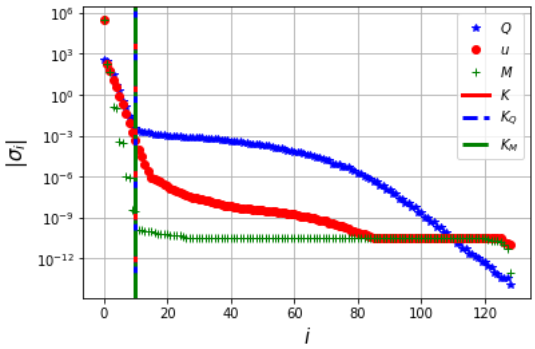


Figure 5.3: L^2 error of the DEIM with second correction reduced model as a function of δ

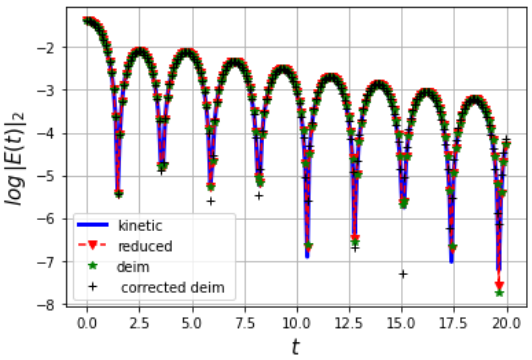
5.4.3 Generalization to other parameters

In Sec. 5.4.1, a different reduced model has been constructed for each set of parameters. Here, we explore the ability to construct a reduced model valid in a full range of the parameters. This generalization property is a key feature: the reduced model could then be used for parameters that are not involved in the construction of the reduced model, without the need to carry out the full kinetic simulations. Here, we would like to build a reduced model valid for all the values (ε, α) in the domain $\mathcal{D} = [1, 10] \times [0.01, 0.1]$. Therefore, we consider 25 pairs (ε, α), randomly chosen in $\mathcal{D}$ and the reduced models is built using 30 samples per pairs (ε, α), resulting into $N_s = 750$ samples. The final time equals $T = 20$. We set $\delta = 1 \times 10^{-10}$.

According to singular value distributions of the sample matrices (see Fig. 5.4), we set $K = K_M = K_Q = 10$. In Fig. 5.4, on the right, is depicted the electric energy field for $(\varepsilon, \alpha) = (5, 5 \times 10^{-2}) \in \mathcal{D}$. We observe that the dynamics are well recovered for each algorithm (with/without DEIM or corrected moments). We also compute the relative L^2 error between the solution of the reduced model after decompression $\Phi \hat{f}$ and the solution of the underlying kinetic model $\mathbf{f}$ at the final time T. Furthermore we compute the L^2 norm of the electric field over the time interval $[0, T]$ in Tab. 5.2. We observe that each algorithm performs well. Moreover, the error is larger for the corrected DEIM ROM method.



(a) Singular values $(\sigma_i)_i$ of the snapshots matrix from the solution u, the Maxwellian M and the collision operator Q.



(b) Electric energy as a function of time for the reference solution and each reduced model.

Figure 5.4: Singular values distributions (left) and logarithm of the norm of the electric field $|E(t, .)|_{L^2}$ as a function of time for $(\varepsilon, \alpha) = (5, 5 \times 10^{-2}) \in \mathcal{D}$.

	ROM	DEIM	corrected DEIM
$\Phi\hat{\mathbf{f}}(T)$	5.36×10^{-4}	5.62×10^{-4}	5.66×10^{-4}
$\lvert E(t,.)\rvert_{L^2}$	1.73×10^{-2}	1.85×10^{-2}	2.63×10^{-2}

Table 5.2: relative L^2 errors, $(\varepsilon, \alpha) = (5, 5 \times 10^{-2}) \in \mathcal{D}$.

In addition, we aim to test the reduced model outside its learned domain. We set a final time of simulation $T_{test} = 25 > T$ and consider $(\varepsilon, \alpha) \in \mathcal{D}_{test} \not\subset \mathcal{D}$. We choose $\mathcal{D}_{test} = \{0.1, 0.5, 1, 10, 15, 20\} \times \{0.2, 0.3\}$ to have more nonlinear damping with both collisional and non-collisional regime. The obtained relative L^2 errors on $\Phi\hat{\mathbf{f}}$ at time T_{test} and on the norm of the electric field on the time interval $[0, T_{test}]$ are given in Tab. 5.3. Overall, each ROM method provides good results with $(t, (\alpha, \varepsilon)) \in [0, T] \times \mathcal{D}$ and shows good generalization performance with $(t, (\alpha, \varepsilon)) \in [T, T_{test}] \times \mathcal{D}_{test}$. For instance, even with $\varepsilon = 0.1$ and 20, solution errors remain bounded by $2 \times 10^{-1}, 2 \times 10^{-2}$ respectively. Going outside the training set, the precision of the models decrease slowly and we still have correctness on both solutions and electric energies.

Going further, we notice that our corrected DEIM ROM performs better when $\varepsilon < 1$. That is due to the fact that its moments are corrected, such as to obtain an Euler-type system. For example, let us observe the energy damping with $(\varepsilon, \alpha) = (0.1, 0.3)$ in Fig. 5.5a. While the ROM and DEIM ROM drift from the reference solution, the corrected DEIM ROM remains close to it. In Fig. 5.5b, we observe that it is no more the case with larger ε.

In conclusion, we have shown that our corrected moments DEIM performs better for small Knudsen numbers $\varepsilon \leqslant 1$ and is better to be used instead of the usual DEIM. Conversely, this correction is not effective with large $\varepsilon > 1$ and the latter is better to be used.

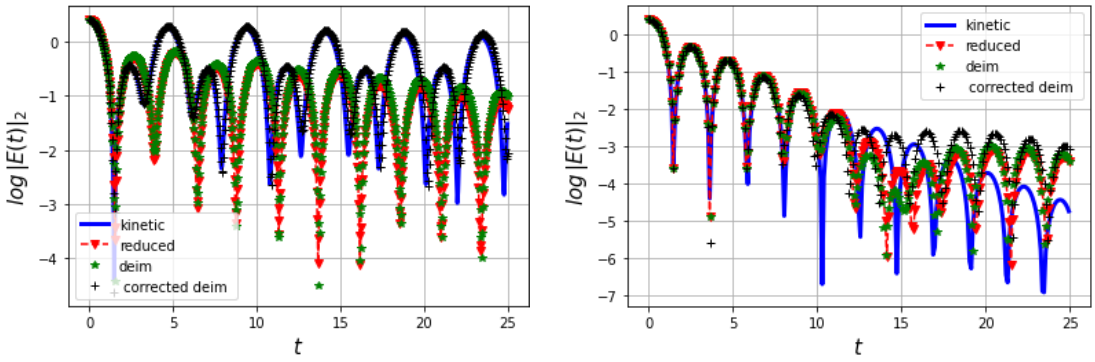
α	Method	ε					
		0.1	0.5	1	10	15	20
0.2	ROM	1.41×10^{-1}	5.58×10^{-2}	3.55×10^{-2}	6.02×10^{-3}	4.44×10^{-1}	6.89×10^{-3}
	DEIM	1.36×10^{-1}	5.56×10^{-2}	3.59×10^{-2}	6.26×10^{-3}	5.48×10^{-3}	8.20×10^{-3}
	cDEIM	1.53×10^{-2}	1.42×10^{-2}	1.34×10^{-2}	6.09×10^{-3}	5.93×10^{-3}	7.30×10^{-3}
0.3	ROM	1.94×10^{-1}	1.18×10^{-1}	9.62×10^{-2}	2.15×10^{-2}	1.75×10^{-2}	1.62×10^{-2}
	DEIM	1.98×10^{-1}	1.21×10^{-1}	9.84×10^{-2}	2.23×10^{-2}	1.97×10^{-2}	1.95×10^{-2}
	cDEIM	3.99×10^{-2}	4.78×10^{-2}	3.93×10^{-2}	1.55×10^{-2}	1.57×10^{-2}	1.64×10^{-2}

(a) relative errors on $\Phi\hat{\mathbf{f}}(T_{test})$.

α	Method	ε					
		0.1	0.5	1	10	15	20
0.2	ROM	5.98×10^{-1}	3.30×10^{-1}	2.15×10^{-1}	2.68×10^{-2}	5.75×10^{-2}	6.96×10^{-2}
	DEIM	5.79×10^{-1}	3.27×10^{-1}	2.12×10^{-1}	2.69×10^{-2}	6.52×10^{-2}	7.92×10^{-2}
	cDEIM	5.42×10^{-2}	7.35×10^{-2}	7.83×10^{-2}	6.75×10^{-2}	8.92×10^{-2}	9.30×10^{-2}
0.3	ROM	6.37×10^{-1}	4.77×10^{-1}	3.65×10^{-1}	6.32×10^{-2}	5.88×10^{-2}	6.18×10^{-2}
	DEIM	6.29×10^{-1}	4.80×10^{-1}	3.65×10^{-1}	6.14×10^{-2}	5.79×10^{-2}	6.36×10^{-2}
	cDEIM	1.03×10^{-1}	1.49×10^{-1}	1.54×10^{-1}	1.05×10^{-1}	1.05×10^{-1}	1.03×10^{-1}

(b) relative errors on $\lvert E(t,.)\rvert_{L^2}$ over $[0, T_{test}]$.

Table 5.3: relative L^2 errors, $(\alpha, \varepsilon) \in \mathcal{D}_{test}$.



(a) $(\varepsilon, \alpha) = (0.1, 0.3)$ (b) $(\varepsilon, \alpha) = (20, 0.3)$

Figure 5.5: Logarithm of the norm of the electric field as a function of time for several $(t, \mu) \in [0, T_{test}] \times \mathcal{D}^{\mu}_{test}$.

5.5 Conclusion

In this paper, we have proposed a reduced order modeling approach in velocity applied to the Vlasov-Poisson-Fokker-Planck equation. This results in a hyperbolic system with source terms approximating the dynamics for a given initial data and a given Knudsen number or a range of initial data and Knudsen numbers. The numerical results show that both collisional and non-collisional regimes can be approximated with a reduced system of size ≈ 10. We also note that we can consider a smaller reduced system in a collisional regime. Inversely, the system size should be larger to capture nonlinear dynamics. We have introduced a corrected reduced collision operator that preserves moments. In addition to ensuring some mathematical structure of the reduced moment equations, it numerically provides better results in collisional regimes. Finally, to extend the generalization range of the method, we would have to turn to nonlinear reduction methods like quadratic ones [15] or autoencoder reduction [16].

References

[1] E. Franck et al. “Hyperbolic reduced model for Vlasov-Poisson equation with Fokker-Planck collision”. In: *ESAIM: ProcS* 77 (2024), pp. 213–228. DOI: 10.1051/proc/202477213.

[2] C. Buet, S. Dellacherie, and R. Sentis. “Numerical solution of an ionic Fokker-Planck equation with electronic temperature”. In: *SIAM J. Numer. Anal.* 39.4 (2001), pp. 1219–1253. ISSN: 0036-1429,1095-7170. DOI: 10.1137/S0036142999359669.

[3] I. Almuslimani and N. Crouseilles. “Conservative stabilized Runge-Kutta methods for the Vlasov-Fokker-Planck equation”. In: *J. Comput. Phys.* 488 (2023), Paper No. 112241, 21. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2023.112241.

[4] J. S. Hesthaven, C. Pagliantini, and G. Rozza. “Reduced basis methods for time-dependent problems”. In: *Acta Numer.* 31 (2022), pp. 265–345. ISSN: 0962-4929,1474-0508. DOI: 10.1017/S0962492922000058.

[5] J. S. Hesthaven, C. Pagliantini, and N. Ripamonti. “Adaptive symplectic model order reduction of parametric particle-based Vlasov-Poisson equation”. In: *Math. Comp.* 93.347 (2024), pp. 1153–1202. ISSN: 0025-5718,1088-6842. DOI: 10.1090/mcom/3885.

[6] F. Cassini and L. Einkemmer. “Efficient 6D Vlasov simulation using the dynamical low-rank framework Ensign”. In: *Comput. Phys. Commun.* 280 (2022), Paper No. 108489, 12. ISSN: 0010-4655,1879-2944. DOI: 10.1016/j.cpc.2022.108489.

[7] V. Ehrlacher and D. Lombardi. “A dynamical adaptive tensor method for the Vlasov-Poisson system”. In: *J. Comput. Phys.* 339 (2017), pp. 285–306. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2017.03.015.

[8] W. Guo, J. F. Ema, and J.-M. Qiu. “A local macroscopic conservative (LoMaC) low rank tensor method with the discontinuous Galerkin method for the Vlasov dynamics”. In: *Commun. Appl. Math. Comput.* 6.1 (2024), pp. 550–575. ISSN: 2096-6385,2661-8893. DOI: 10.1007/s42967-023-00277-7.

[9] S. Chaturantabut and D. C. Sorensen. “Nonlinear model reduction via discrete empirical interpolation”. In: *SIAM J. Sci. Comput.* 32.5 (2010), pp. 2737–2764. ISSN: 1064-8275,1095-7197. DOI: 10.1137/090766498.

[10] S. Riffaud. “Reduced-order models: convergence between scientific computing and data for fluid mechanics”. PhD thesis. Université de Bordeaux, 2020.

[11] I. M. Gamba and S. H. Tharkabhushanam. “Spectral-Lagrangian methods for collisional models of non-equilibrium statistical states”. In: *J. Comput. Phys.* 228.6 (2009), pp. 2012–2036. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2008.09.033.

[12] C. Zhang and I. M. Gamba. “A conservative discontinuous Galerkin solver for the space homogeneous Boltzmann equation for binary interactions”. In: *SIAM J. Numer. Anal.* 56.5 (2018), pp. 3040–3070. ISSN: 0036-1429,1095-7170. DOI: 10.1137/16M1104792.

[13] G. Dimarco and R. Loubere. “Towards an ultra efficient kinetic scheme. Part II: The high order case”. In: *J. Comput. Phys.* 255 (2013), pp. 699–719. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2013.07.017.

[14] A. Crestetto, N. Crouseilles, and M. Lemou. “Kinetic/fluid micro-macro numerical schemes for Vlasov-Poisson-BGK equation using particles”. In: *Kinet. Relat. Models* 5.4 (2012), pp. 787–816. ISSN: 1937-5093,1937-5077. DOI: 10.3934/krm.2012.5.787.

[15] R. Geelen, S. Wright, and K. Willcox. “Operator inference for non-intrusive model reduction with quadratic manifolds”. In: *Comput. Methods Appl. Mech. Engrg.* 403 (2023), Paper No. 115717, 24. ISSN: 0045-7825,1879-2138. DOI: 10.1016/j.cma.2022.115717.

[16] K. Lee and K. T. Carlberg. “Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders”. In: *J. Comput. Phys.* 404 (2020), pp. 108973, 32. ISSN: 0021-9991,1090-2716. DOI: 10.1016/j.jcp.2019.108973.

A Numerical discretization details

A.1 Discretization of the full model

Eq. (5.7) is a hyperbolic system with a source term whose transport parts can be solved using a finite volume method, while the Poisson Eq. (5.8) is discretized with a finite difference method. The n-th iteration of the scheme reads:

$$-\frac{\phi_{i+1}^n - 2\phi_i^n - \phi_{i-1}^n}{\Delta x^2} = \frac{1}{L}\sum_{i=0}^{N_x-1} \Delta x\, \Delta v (1,\dots,1)^T \mathbf{f}_i^n - \Delta v (1,\dots,1)^T \mathbf{f}_i^n,$$

$$\frac{\mathbf{f}_i^{n+1} - \mathbf{f}_i^n}{\Delta t} + \frac{\mathbf{f}_{i+\frac{1}{2}}^n - \mathbf{f}_{i-\frac{1}{2}}^n}{\Delta x} + \left(\frac{\phi_{i+1}^n - \phi_{i-1}^n}{2\Delta x}\right) D\, \mathbf{f}_i^n = \frac{1}{\varepsilon} Q_{\Delta v}(\mathbf{f}^n)_i$$

where $\mathbf{f}^n_{i+\frac{1}{2}}$ is an upwind approximation with respect to the velocity given by:

$$\mathbf{f}^n_{i+\frac{1}{2}} = \mathrm{Diag}(\mathbf{v})\frac{\mathbf{f}^n_{i+1}+\mathbf{f}^n_i}{2} - \mathrm{Diag}(|\mathbf{v}|)\frac{\mathbf{f}^n_{i+1}-\mathbf{f}^n_i}{2}.$$

To obtain better accuracy in space, we use a MUSCL method in the upwind flux. The transport scheme in space and velocity induces a CFL stability condition:

$$\Delta t \leq \min\Big(\frac{\Delta x}{v_{\max}}, \frac{\Delta v}{\max_i \left|\frac{\phi^n_{i+1}-\phi^n_{i-1}}{2\Delta x}\right|}\Big).$$

The explicit treatment of the collision operator results in an additional constraint: $\Delta t \leq \varepsilon C \Delta v^2$ with $C > 0$ related to the discrete Maxwellian distributions [2]. In practice, we take Δt sufficiently small in order to ensure the stability of the numerical simulations. Recently, high-order Runge Kutta methods have been proposed to relax the constraint [3].

A.2 Discretization of the reduced model

Since the reduced model is a hyperbolic system, we use a finite volume scheme. Since the transport is linear, we consider the upwind scheme, whose fluxes write:

$$\hat{\mathbf{f}}^n_{i+\frac{1}{2}} = \hat{A}\,\frac{\hat{\mathbf{f}}^n_{i+1}+\hat{\mathbf{f}}^n_i}{2} - |\hat{A}|\,\frac{\hat{\mathbf{f}}^n_{i+1}-\hat{\mathbf{f}}^n_i}{2}. \tag{5.28}$$

As for the kinetic solver, we use a MUSCL strategy to obtain second order accuracy.

B Solution to the minimization problem

Property B.1. *Let $\Phi \in M_{K,N_v}$, with $\Phi^T\Phi = \mathrm{Id}$, $\hat{m} \in M_{3,K}$ of full rank. Then the unique solution to the least-square problem:*

$$\hat{N}^c = \underset{\widehat{N}\in\mathbb{R}^K \text{ s.t. } \hat{m}\widehat{N}=b}{\arg\min} \|\Phi\hat{N} - N\|^2,$$

is given by:

$$\hat{N}^c = \Phi^T N + \hat{m}^T(\hat{m}\hat{m}^T)^{-1}\left(b - \hat{m}\Phi^T N\right).$$

Proof. Φ being of full rank, the least-square problem under constraints has a unique solution denoted $\hat{N}^c$. Let $f(\hat{N}) = \|\Phi\hat{N} - N\|^2$. Its gradient is given by:

$$\nabla f(\hat{N}) = 2(\Phi^T\Phi\hat{N} - \Phi^T N) = 2(\hat{N} - \Phi^T N).$$

Applying the Lagrange multiplier method, there exists $\lambda \in \mathbb{R}^K$ such that

$$2(\hat{N}^c - \Phi^T N) + \hat{m}^T\lambda = 0, \tag{5.29}$$

$$\hat{m}\hat{N}^c = b. \tag{5.30}$$

Then multiplying (5.29) by $\hat{m}$ and using (5.30), we get the value of λ:

$$\lambda = -2(\hat{m}\hat{m}^T)^{-1}(b - \hat{m}\Phi^T N).$$

Then, reporting this value into (5.29), we obtain the expected solution. □

Chapter 6

Conclusion & perspectives

In this thesis, we have developed new Hamiltonian model order reduction methods, based on deep learning. In Chap. 2, we presented the basics of Hamiltonian dynamics, starting from the framework of Hamiltonian Partial Differential Equations (PDEs). These PDEs govern numerous physical systems, such as wave dynamics, fluid dynamics and plasma physics, where energy conservation and symplectic structure play a central role. Indeed, knowing the Hamiltonian function together with the symplectic form, expressed via a Poisson bracket or equivalently a skew-adjoint operator, ensures the conservation of several quantities, including the Hamiltonian itself and the underlying symplectic structure. When enough quantities are conserved, the system becomes integrable and its dynamics can be described using a limited number of variables, known as degrees of freedom.

To perform numerical simulations, we first semi-discretize them into Hamiltonian Ordinary Differential Equations (ODEs) by using finite element, finite difference, or particle-based methods. To preserve the underlying structure, symplectic time discretization schemes must then be employed. In the end, the discretized system is high-dimensional, potentially nonlinear, and thus computationally intensive, particularly in control, real-time, or embedded applications. To address this the Proper Symplectic Decomposition (PSD) has been proposed: this is a symplectic variant of the Proper Orthogonal Decomposition (POD). Assuming the solution manifold is well approximated by a linear subspace, the PSD yields an appropriate low-dimensional reduced model. With additional effort, hyper-reduction techniques, such as the Symplectic Discrete Empirical Interpolation Method (SDEIM), enable an practical and effective reduction in computation time while maintaining a controlled error, particularly in nearly linear regimes.

As presented in Chaps. 3 and 4, the PSD approximation is not well suited to nonlinear dynamics such as the Vlasov-Poisson system, the shallow-water equations or nonlinear wave models. To address this, we proposed deep learning based approaches tailored to nonlinear Hamiltonian dynamics. Our first idea was to replace the linear projection with an AutoEncoder (AE), trained at least to minimize the reconstruction error.

This yields a nonlinear, dynamics-specific projection, but at the expense of losing symplecticity and lacking of an explicit expression for the reduced model. To resolve this issue, we employed a Hamiltonian Neural Network (HNN) to learn the reduced dynamics while preserving the symplectic structure. Instead of enforcing a weakly symplectic AE via a loss term, we opted for a coupled training strategy to ensure the reduced dynamics closely approximates the full system, thus maintaining a limited approximation error. In addition, we employ convolutional AEs to lighten our model in terms of the number of parameters and, when applicable, to leverage the spatial structure of the data in the encoding process.

In Chap. 3, we tested the AE-HNN method on various nonlinear test cases and obtained satisfactory results. Our next objective was to extend our reduction method to plasma physics, par-

ticularly Particle-In-Cell (PIC) simulations of Vlasov-Poisson dynamics, which are widely used in this field. Unlike previous systems, this numerical model introduces two additional challenges: solutions are represented as large, unstructured distributions of discrete particles. To tackle this, we developed a two-stage reduction strategy combining the PSD with our AE-HNN method, as detailed in Chap. 4, and again observed promising results across several benchmark problems.

We summarize the main strengths and limitations of our methods. To start with, they are both generic and non-intrusive: they adapt to the structure of the model at hand and learns tailored mappings—whether the solution involves waves, free surfaces, or reduced basis coefficients—without directly manipulating the governing equations. This highlights the strength of data-driven approaches compared with analytical reduction strategies. In addition, constructing nonlinear symplectic projections remains an active area of research with few available results; the joint training strategy is a practical workaround. While neural network training is generally computationally intensive (offline stage), the prediction of reduced dynamics (online stage) is often sufficiently efficient to be performed on a personal computer. Furthermore, neural network evaluations are highly parallelizable on GPUs, enabling multiple reduced model evaluations in parallel.

Next, the weaknesses of our methods are comparable to those in many scientific machine learning applications. The reduced model's performance depends on a high-dimensional minimization process in the neural network parameter space, which is non-convex with local minima found iteratively without a clear guarantee of global convergence. While training often converges to a satisfactory reduced model with sufficient effort, this process remains quite empirical. There are ongoing efforts made to derive error bounds for deep learning-based model order reduction, as illustrated in [1]. However, such results remain scarce and recent, reflecting the early stage of this research area. Thus, there is no systematic way to improve the accuracy of our reduced model yet. Unlike classical reduction methods, where the main factor influencing accuracy is the dimension/rank of the reduced model, reduction with neural networks offers many more tunable levers via hyperparameters (number of layers, layer sizes, activation functions, specific loss or optimizers, etc.).

Perspectives The field of structure-preserving model order reduction combined with deep learning is still in its early stages and remains highly active and rapidly evolving. We discuss here some directions for future research and possible extensions of this work.

1. A notable improvement would be the ability to systematically enhance the accuracy of the reduced model, as mentioned earlier. Currently, hyperparameter tuning is performed manually, relying partly on experience. It would be beneficial to automate this step, perhaps using a genetic algorithm, or to identify key hyperparameters through sensitivity analysis.

2. In practice, the HNN often succeeds in learning a Hamiltonian that effectively captures the reduced dynamics. Nonetheless, the primary limitation lies in the AE. Minimizing the reconstruction error alone tends to converge fairly quickly and yields good results. However, coupling the AE with the HNN requires reconciling potentially competing objectives. In other words, is it better to minimize reconstruction error at the cost of a reduced space that hinders the HNN's learning, or vice versa? We tried adding a weak symplecticity loss term, as in [2], but it did not improve the results and incurred high computational costs in our experiments. A major objective would be to design a symplectic AE by construction that outperforms our current unconstrained AE with an HNN. Some work in this direction is presented in [3], where SympNets and PSD are combined into a network with a tailored gradient descent.

3. The HNN is limited to learning canonical systems, which may be a constraining factor. Several studies have already focused on learning non-canonical Hamiltonian systems [4, 5, 6]. A promising direction for future work is to extend our approach to these problems.

4. Physics-informed machine learning is not limited to Hamiltonian systems: it generalizes to dissipative systems, e.g. with port-Hamiltonian systems [7] or GFINNs [8]. It would be interesting to develop model order reduction techniques for models combining both Hamiltonian and dissipative components. However, this may prove significantly more challenging due to the increased number of operators involved.

References

[1] S. Brivio et al. "Error estimates for POD-DL-ROMs: a deep learning framework for reduced order modeling of nonlinear parametrized PDEs enhanced by proper orthogonal decomposition". In: *Adv. Comput. Math.* 50.3 (2024), p. 33. ISSN: 1572-9044. DOI: 10.1007/s10444-024-10110-1.

[2] P. Buchfink, S. Glas, and B. Haasdonk. "Symplectic model reduction of Hamiltonian systems on nonlinear manifolds and approximation with weakly symplectic autoencoder". In: *SIAM J. Sci. Comput.* 45.2 (2023), A289–A311. ISSN: 1064-8275,1095-7197. DOI: 10.1137/21M1466657.

[3] B. Brantner and M. Kraus. *Symplectic Autoencoders for Model Reduction of Hamiltonian Systems*. 2023. DOI: 10.48550/arXiv.2312.10004. arXiv: 2312.10004 [cs.LG].

[4] A. Choudhary et al. "Forecasting Hamiltonian dynamics without canonical coordinates". In: *Nonlinear Dyn.* 103.2 (2021), pp. 1553–1562. ISSN: 1573-269X. DOI: 10.1007/s11071-020-06185-2.

[5] A. Gruber and I. Tezaur. "Canonical and noncanonical Hamiltonian operator inference". In: *Comput. Methods Appl. Mech. Engrg.* 416 (2023), Paper No. 116334, 45. ISSN: 0045-7825,1879-2138. DOI: 10.1016/j.cma.2023.116334.

[6] P. Jin et al. "Learning Poisson systems and trajectories of autonomous systems via Poisson neural networks". In: *IEEE Trans. Neural Netw. Learn. Syst.* 34.11 (2023), pp. 8271–8283. ISSN: 2162-237X,2162-2388. DOI: 10.1109/tnnls.2022.3148734.

[7] S. A. Desai et al. "Port-Hamiltonian neural networks for learning explicit time-dependent dynamical systems". In: *Phys. Rev. E* 104 (3 2021), p. 034312. DOI: 10.1103/PhysRevE.104.034312.

[8] Z. Zhang, Y. Shin, and G. E. Karniadakis. "GFINNs: GENERIC formalism informed neural networks for deterministic and stochastic dynamical systems". In: *Philos. Trans. Roy. Soc. A* 380.2229 (2022), Paper No. 20210207, 21. ISSN: 1364-503X,1471-2962. DOI: 10.1098/rsta.2021.0207.

Dans cette thèse, nous développons de nouvelles méthodes de réduction d'ordre pour les systèmes hamiltoniens, en nous appuyant sur des techniques d'apprentissage profond.

Dans les deux premiers chapitres, nous présentons les notions et méthodes essentielles à ce travail. Nous introduisons d'abord les systèmes hamiltoniens ainsi que leur structure symplectique. Nous décrivons ensuite un ensemble de méthodes de discrétisation qui préservent cette structure, afin d'aboutir à un modèle numérique hamiltonien. La préservation de cette structure est cruciale, car elle garantit de nombreuses propriétés numériques remarquables, telles que la conservation de l'énergie totale, la stabilité à long terme, la fidélité aux lois physiques, ainsi que la préservation du volume dans l'espace des phases, entre autres.

Néanmoins, les systèmes d'équations, dits complets, qui nous intéressent sont de grande dimension et présentent des non-linéarités qui rendent les simulations numériques particulièrement coûteuses. Cela nous conduit à développer des modèles réduits, c'est-à-dire des modèles de taille beaucoup plus faible, capables de reproduire la dynamique du système complet. Par ailleurs, nous souhaitons que ces modèles réduits conservent leur structure hamiltonienne afin de bénéficier des propriétés numériques associées. Dans ce but, nous élaborons des méthodes de réduction combinant des approches classiques avec des réseaux de neurones, en particulier des auto-encodeurs convolutifs et des réseaux de neurones hamiltoniens. Ces composantes sont fortement couplées au moyen de stratégies d'apprentissage, afin de garantir que le modèle réduit conserve la structure hamiltonienne et reste fidèle à la dynamique du modèle complet.

Ces méthodes sont développées dans les chapitres trois et quatre, et testées sur différents ensembles d'équations. Dans un premier temps, nous nous concentrons sur des équations d'ondes 1D, linéaires et non-linéaires. Nous appliquons ensuite notre méthode à la dynamique des fluides, à travers les équations de Saint-Venant en 1D et 2D. Enfin, nous réduisons également un modèle Particle-In-Cell hamiltonien de l'équation de Vlasov-Poisson en 1D-1V, qui modélise la dynamique des plasmas dans un tore.

Le dernier chapitre, quant à lui, est consacré à l'équation de Vlasov-Poisson 1D-1V avec un terme de collision non-linéaire de Fokker-Planck. Un modèle réduit non hamiltonien y est développé. Ce travail se concentre sur d'autres propriétés de l'opérateur de collision : on cherche à ce qu'il vérifie un système de type Euler, qui est proche des régimes fortement collisionnels.

INSTITUT DE RECHERCHE MATHÉMATIQUE AVANCÉE
UMR 7501
Université de Strasbourg et CNRS
7 Rue René Descartes
67 084 STRASBOURG CEDEX

Tél. 03 68 85 01 29
Fax 03 68 85 03 28

https://irma.math.unistra.fr
irma@math.unistra.fr

Université de Strasbourg

IRMA
Institut de Recherche Mathématique Avancée

IRMA 2025/004
ISSN 0755-3390
https://tel.archives-ouvertes.fr/tel-05219770