2019 IEEE 58th Conference on Decision and Control (CDC)
Palais des Congrès et des Expositions Nice Acropolis
Nice, France, December 11-13, 2019

Inferring Particle Interaction Physical Models and Their Dynamical Properties

Ion Matei, Christos Mavridis, John S. Baras and Maksym Zhenirovskyy

***Abstract*— We propose a framework based on port-Hamiltonian modeling formalism aimed at learning interaction models between particles (or networked systems) and dynamical properties such as trajectory symmetries and conservation laws of the ensemble (or swarm). The learning process is based on approaches and platforms used for large scale optimization and uses features such as automatic differentiation to compute gradients of optimization loss functions. We showcase our approach on the Cucker-Smale particle interaction model, which is first represented in a port-Hamiltonian form, and for which we re-discover the interaction model, and learn dynamical properties that are previously proved analytically. Our approach has the potential for discovering novel particle cooperation rules that can be extracted and used in cooperative control system applications.**

I. Introduction

Extracting physical laws that govern a given system from data is a central challenge in many diverse areas of science and engineering. Most complex systems can be described as discrete structures (graphs) with dynamical relations [1]. Such networked systems are ubiquitous and include multi-body systems, chemical reaction networks, animal and UAV swarms, and power systems. A fundamental challenge in complex networked systems is to infer the laws of interaction between particles and their dynamical properties [1]. The problem has been approached either by using statistical learning [2] [3], or by learning the parameters of equations modeling the system. In [4] symbolic equations are generated from the numerically calculated derivatives of the system variables. In [5], [6] the constitutive equations of physical components the system are learned using acausal representations, while in [7] the order of fractional differential equations modeling the system is estimated.

A general and powerful geometric framework to model complex dynamical networked systems is the port-Hamiltonian modeling formalism [8], [9], [10]. Port-Hamiltonian systems are based on a known energy function (Hamiltonian) and the interconnection of atomic structure elements (e.g. inertias, springs and dampers for mechanical systems) that interact by exchanging energy. They provide an energy-consistent description of a physical system, having the property that a power conservative interconnection of port-Hamiltonian systems is again a port-Hamiltonian system [11].

In addition, the port-Hamiltonian system framework is particularly suited for finding symmetries and conserved quantities [12]. In particular, it allows to find conserved quantities, in addition to the Hamiltonian, called Casimir functions [9], by examining conditions related to the port-Hamiltonian system at hand, which can lead to model simplification (reduction). Moreover, finding parameterized symmetries, e.g., Lie groups of transformations, can lead to data generation without experimentation, as well as provide insight on the modeling equations of the system itself [13], [14].

In this work, we are interested in models describing the dynamics of swarms or *particle* ensembles (e.g. bird flocks), which have been studied intensively through the years [15], [16], [17]. We model the system of interacting particles as a graph topology based on port-Hamiltonian components, and investigate its dynamical properties, such as discrete symmetries of trajectories, Lie groups of invariance transformations and conservation laws. To showcase our approach we use the Cucker-Smale (CS) model [16] to generate training and test data for the learning tasks. We apply large scale optimization methods implemented on deep learning platforms to learn the particle interaction model from data, and recover its dynamical properties. Finally, we compare the results of our method to the ones derived by the theoretical analysis.

The rest of the manuscript is organized as follows: Section II introduces the CS dynamical model for particle interactions, the port-Hamiltonian formalism, discrete symmetries and the Lie groups of invariance transformations. Section III describes the port-Hamiltonian representation of the CS interaction model. In Section IV we prove theoretical results on discrete symmetries, Lie groups of invariance transformations and conservation laws based on Casimir functions. Section V describes optimization based learning algorithms that are used to recover the particle interaction model, the discrete and Lie symmetry maps and the conserved quantities. Finally, Section VI concludes the paper.

II. Preliminaries

In this section we first describe the CS dynamical model used to showcase our approach, we give a brief description of the port-Hamiltonian formalism and introduce the notions of symmetries and Lie groups of invariance transformations.

Ion Matei and Maksym Zhenirovskyy are with the Palo Alto Research Center (PARC), Palo Alto, CA (emails: ion.matei@parc.com, maksym.zhenirovskyy@parc.com). Christos Mavridis and John S. Baras are with the Department of Electrical and Computer Engineering and the Institute for Systems Research, University of Maryland, College Park, MD (emails: mavridis@umd.edu, baras@isr.umd.edu).

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00111990027.

A. Cucker-Smale Particle Interaction Model

Let i denote a particle in an ensemble of N particles. The CS particle interaction model [16] is given by $\dot{x}_i = v_i$ and $\dot{v}_i = \frac{1}{N}\sum_{j=1}^{N} G(\|x_i - x_j\|)(v_j - v_i)$, where a typical choice for the interaction function G is $G(r) = \frac{1}{(1+r^2)^\gamma}$. The above dynamics ensures velocity alignment of all particles [16], [17]. An extension of the original model comes from adding a potential function [16], [17], resulting in the dynamics $\dot{x}_i = v_i$ and $\dot{v}_i = \frac{1}{N}\sum_{j=1}^{N} G(\|x_i - x_j\|)(v_j - v_i) - \frac{1}{N}\sum_{i\neq j}\nabla U(\|x_i - x_j\|)$, where the potential function takes the form $U(r) = -C_A e^{-r/l_A} + C_R e^{-r/l_R}$, with C_A, C_R, l_A, l_R positive scalars. The above model can be compactly written as

$$\dot{\mathbf{x}} = \mathbf{v} \quad (1)$$

$$\dot{\mathbf{v}} = \mathbf{G}(\mathbf{x})\mathbf{v} - \nabla\mathbf{U}(\mathbf{x}), \quad (2)$$

where $[\mathbf{x}]_i = x_i$, $[\mathbf{v}]_i = v_i$, $[\ \mathbf{G}(\mathbf{x})]_{i,i} = -\frac{1}{N}\sum_{i=1}^{N} G(\|x_i - x_j\|)$, $[\ \mathbf{G}(\mathbf{x})]_{i,j} = \frac{1}{N}G(\|x_i - x_j\|)$, for $i \neq j$, $[\nabla\mathbf{U}(\mathbf{x})]_{i,i} = 0$, and $[\nabla\mathbf{U}(\mathbf{x})]_{i,j} = \frac{1}{N}\nabla U(\|x_i - x_j\|)$, for $i \neq j$.

B. Port-Hamiltonian Systems

Consider a finite-dimensional linear state space $\mathcal{X}$ along with a Hamiltonian $H : \mathcal{X} \to \mathbb{R}_+$ defining energy-storage, and a set of pairs of *effort* and *flow* variables $\{(e_i, f_i) \in \mathcal{E}_i \times \mathcal{F}_i,\ i \in \{S, R, P\}\}$, describing ports (ensembles of elements) that interact by exchanging energy. Then, the dynamics of a port-Hamiltonian system $\Sigma = (\mathcal{X}, H, \mathcal{S}, \mathcal{R}, \mathcal{P}, \mathcal{D})$ are defined by a Dirac Structure $\mathcal{D}$ [9], [10] as

$$(f_\mathcal{S}, e_\mathcal{S}, f_\mathcal{R}, e_\mathcal{R}, f_\mathcal{P}, e_\mathcal{P}) \in \mathcal{D} \Leftrightarrow e_\mathcal{S}^T f_\mathcal{S} + e_\mathcal{R}^T f_\mathcal{R} + e_\mathcal{P}^T f_\mathcal{P} = 0,$$

where (i) $\mathcal{S} = (f_\mathcal{S}, e_\mathcal{S}) \in \mathcal{F}_R \times \mathcal{E}_R = \mathcal{X} \times \mathcal{X}$ is an energy-storing port, consisting of the union of all the energy-storing elements of the system (e.g. inertias and springs in mechanical systems), satisfying $f_\mathcal{S} = -\dot{x}, e_\mathcal{S} = \frac{\partial H}{\partial x}(x)$, $x \in \mathcal{X}$ such that $\frac{d}{dt}H = -e_\mathcal{S}^T f_\mathcal{S} = e_\mathcal{R}^T f_\mathcal{R} + e_\mathcal{P}^T f_\mathcal{P}$, (ii) $\mathcal{R} = (f_\mathcal{R}, e_\mathcal{R}) \in \mathcal{F}_R \times \mathcal{E}_R$ is an energy-dissipation (resistive) port, consisting of the union of all the resistive elements of the system (e.g. dampers in mechanical systems), satisfying $\langle e_\mathcal{R}, f_\mathcal{R}\rangle \leq 0$ and, usually, an input-output relation $f_\mathcal{R} = -R(e_\mathcal{R})$, (iii) $\mathcal{P} = (f_\mathcal{P}, e_\mathcal{P}) \in \mathcal{F}_P \times \mathcal{E}_P$ is an external port modeling the interaction of the system with the environment, consisting of a control port $\mathcal{C}$ and an interconnection port $\mathcal{I}$, and (iv) $\mathcal{D} \subset \mathcal{F} \times \mathcal{E} = \mathcal{F}_R \times \mathcal{E}_R \times \mathcal{F}_R \times \mathcal{E}_R \times \mathcal{F}_P \times \mathcal{E}_P$ is a central power-conserving interconnection (energy-routing) structure (e.g. transformers in electrical systems), satisfying $\langle e, f\rangle = 0$, $\forall (f, e) \in \mathcal{D}$, and $dim\mathcal{D} = dim\mathcal{F}$, where $\mathcal{E} = \mathcal{F}^*$, and the duality product $\langle e, f\rangle$ represents power.

The basic property of port-Hamiltonian systems is that the power-conserving interconnection of any number of port-Hamiltonian systems is again a port-Hamiltonian system. An important and useful special case is the class of input-state-output port-Hamiltonian systems $\dot{x} = [J(x) - R(x)]\frac{\partial H}{\partial x}(x) + g(x)u$, $y = g^T(x)\frac{\partial H}{\partial x}(x)$, where u, y are the input–output pairs corresponding to the control port $\mathcal{C}$, $J(x) = -J^T(x)$ is skew-symmetric, while the matrix $R(x) = R^T(x) \geq 0$ specifies the resistive structure.

C. Symmetries and Lie Group of Transformations

Give a a differential algebraic equation (DAE) $F(\dot{x}, x) = 0$ with $x \in \mathcal{X} \subseteq \mathbb{R}^n$, the map $\Psi : \mathcal{X} \times S \to \mathcal{X}$ is a symmetry map, if it is a diffeomorphism, and $\hat{x} = \Psi(x)$ is a solution for the DAE, that is $F(\hat{\dot{x}}, \hat{x}) = 0$. Therefore the symmetry map must obey the symmetry condition $F\left(\frac{\partial\Psi(x)}{\partial x}\dot{x}, \Psi(x)\right) = 0$.

A particular type of symmetry maps are Lie groups of invariance transformations. The map $\Psi : \mathcal{X} \times S \to \mathcal{X}$, where $S \subset \mathbb{R}$ is an interval with $0 \in S$, along with a composition law $\phi : S \times S \to S$, defines a parametrized (Lie) symmetry [12], and, in particular, a Lie group of invariance transformations, if for any solution x of the DAE, and for all $\epsilon \in S$, $\hat{x}(t) = \Psi(x(t), \epsilon)$ is also a solution of the system, and (i) $\Psi(x, \epsilon)$ is smooth in x and analytic in ϵ, (ii) $\Psi(\cdot, \epsilon)$ is an injection for all $\epsilon \in S$, (iii) (S, ϕ) forms a group with identity element zero, and ϕ is analytic, (iv) $\Psi(x, 0) = x$, $\forall x \in D$, and (v) if $x_\epsilon = \Psi(x, \epsilon)$ and $x_\delta = \Psi(x_\epsilon, \delta)$, then $x_\delta = \Psi(x, \phi(\epsilon, \delta))$. Using the infinitesimal operator $\boldsymbol{X} = \frac{\partial\Psi(x,\epsilon)}{\partial\epsilon}|_{\epsilon=0}\frac{\partial}{\partial x} = \eta(x)\frac{\partial}{\partial x}$, we have that $\hat{x} = \Psi(x, \epsilon) = e^{\epsilon\boldsymbol{X}}x$ and the symmetry condition becomes $\boldsymbol{X}'F(\dot{x}, x) = 0$, or $\eta(x)^T\frac{\partial F}{\partial x}(\dot{x}, x) + \dot{x}^T\frac{\partial \eta}{\partial x}(x)\frac{\partial F}{\partial \dot{x}}(\dot{x}, x) = 0$.

III. Port-Hamiltonian representation of the Cucker-Smale interaction model

We introduce the notion of generalized mass-spring damper (gMSD) components. We typically consider masses as having one port, and springs and dampers as having two ports. Ports are component interfaces through which energy is exchanged. Their dynamical representations of the gMSD are as follows: mass $\dot{p} = f$, $v = \frac{\partial H}{\partial q}$, spring $\dot{q} = v$, $f = \frac{\partial H}{\partial q}$, damper $f = R(q)v$. In the case of the mass, p is the momentum, f is the force acting on the mass, v is the mass velocity and H is the mass Hamiltonian function. In the case of the spring, q is the spring elongation (the difference between the positions at the two ports), v is the relative velocity, f is the force through the spring and H denotes the spring's Hamiltonian. In the case of the damper, f is the force through the damper, q is the relative position of the damper, R is a resitive term as a function of q and v is the relative velocity.

Proposition 3.1: The CS model with potential is equivalent to a fully connected N-dimensional network of generalized mass-spring-dampers, where each node i in the network is a mass, and each link (i, j) a parallel composition of a spring and damper. The Hamiltonian functions for the mass-springs are given by $H(p) = \frac{1}{2}p^T p$ and $H(q) = \frac{1}{N}\left[-C_A e^{-\frac{\|q\|}{l_A}} + C_R e^{-\frac{\|q\|}{l_R}}\right]$, respectively, and and the resistive function of the damper is given by $R(q) = \frac{1}{N[1+\|q\|^2]^\gamma}$. □

We show an example of this result for the one dimensional case ($p, q \in \mathbb{R}$) and for the 3 particles case. The result holds for the general case, but the notations become more cumbersome. The fully connected topology of the gMSD network is shown in Figure 1. We denote by H_i and H_{ij} the Hamiltonian functions of the masses and springs, respectively. We note that since we assume, unitary masses, the momenta are equal to the mass velocities, that is, $p_i = v_i$,

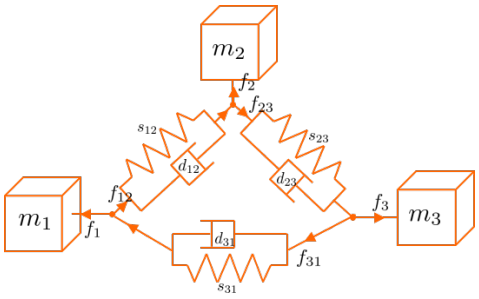

Fig. 1: Fully connected, 3-dimensional gMSD network

$i = \{1,2,3\}$. The forces through the links are the sum of the forces through the dampers and springs, and are given by $f_{ij} = \frac{\partial H_{ij}}{\partial q_{ij}} + R(q_{ij})(v_i - v_j)$, for $(i,j) \in \{(1,2),(2,3),(3,1)\}$. The forces through the masses can be expressed as: $f_1 = f_{31} - f_{12}$, $f_2 = f_{12} - f_{23}$ and $f_3 = f_{23} - f_{31}$. We get the expressions for the mass momenta dynamics as:

$$\dot{p}_1 = \frac{\partial H_{31}}{\partial q_{31}} - \frac{\partial H_{12}}{\partial q_{12}} + R(q_{31})(v_3 - v_1) + R(q_{12})(v_2 - v_1), \quad (3)$$

$$\dot{p}_2 = \frac{\partial H_{12}}{\partial q_{12}} - \frac{\partial H_{23}}{\partial q_{23}} + R(q_{12})(v_1 - v_2) + R(q_{23})(v_3 - v_2), \quad (4)$$

$$\dot{p}_3 = \frac{\partial H_{23}}{\partial q_{23}} - \frac{\partial H_{31}}{\partial q_{31}} + R(q_{23})(v_2 - v_3) + R(q_{31})(v_1 - v_3). \quad (5)$$

The dynamics for the spring elongations are

$$\dot{q}_{ij} = v_i - v_j = \frac{\partial H_i}{\partial p_i} - \frac{\partial H_j}{\partial p_j} \quad (6)$$

for $(i,j) \in \{(1,2),(2,3),(3,1)\}$. To recover the CS model with potential, we replace the relative positions q_{ij} with the absolute positions, namely $q_{ij} = q_i - q_j$ Recalling that spring potentials are symmetric functions, we get that

$$\frac{\partial H_{31}}{\partial q_{31}} - \frac{\partial H_{12}}{\partial q_{12}} = -\frac{1}{3}(\nabla U(q_1 - q_3) - \nabla U(q_1 - q_2)) \quad (7)$$

$$\frac{\partial H_{12}}{\partial q_{12}} - \frac{\partial H_{23}}{\partial q_{23}} = -\frac{1}{3}(\nabla U(q_2 - q_1) - \nabla U(q_2 - q_3)) \quad (8)$$

$$\frac{\partial H_{23}}{\partial q_{23}} - \frac{\partial H_{31}}{\partial q_{31}} = -\frac{1}{3}(\nabla U(q_3 - q_2) - \nabla U(q_3 - q_2)) \quad (9)$$

Substituting (7)-(9) in (3)-(5), and recalling that under our assumptions $p_i = v_i$, we recover exactly the CS model with potential. □

By introducing the notation $\mathbf{z}^T = [\mathbf{p}^T, \mathbf{q}^T]$, with $\mathbf{p}^T = [p_1, p_2, p_3]$ and $\mathbf{q}^T = [q_{12}, q_{23}, q_{31}]$, the equations (3)-(5) and (6) can be expressed compactly as

$$\dot{\mathbf{z}} = [\mathbf{J}(\mathbf{z}) - \mathbf{R}(\mathbf{z})] \frac{\partial \mathbf{H}(\mathbf{z})}{\partial \mathbf{z}}, \quad (10)$$

where $\mathbf{H}(\mathbf{z}) = H_1(p_1) + H_2(p_2) + H_3(p_3) + H_{12}(q_{12}) + H_{23}(q_{23}) + H_{31}(q_{31})$, and

$$\mathbf{R}(\mathbf{z}) = \begin{bmatrix} R(\mathbf{z}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

with

$$R(\mathbf{z}) = \begin{bmatrix} R(q_{12}) + R(q_{31}) & -R(q_{12}) & -R(q_{31}) \\ -R(q_{12}) & R(q_{12}) + R(q_{23}) & -R(q_{23}) \\ -R(q_{31}) & -R(q_{23}) & R(q_{31}) + R(q_{23}) \end{bmatrix}$$

and where

$$\mathbf{J}(\mathbf{z}) = \begin{bmatrix} \mathbf{0} & J \\ -J^T & \mathbf{0} \end{bmatrix}, \text{ with } J = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}.$$

We recognize equation (10) as the typical input-state-output port-Hamiltonian system [9], [10].

IV. Dynamical properties of the Cucker-Smale model

In this section we introduce a set of maps for which we demonstrate that they satisfy the required properties for being symmetry or Lie symmetry maps. In addition, we introduce a conserved quantity that differs from the Hamiltonian function. The maps and the conserved quantity will be rediscovered in the learning section. The symmetry maps will be introduced for both the original CS model and its port-Hamiltonian representation. We consider the 1-d case ($\mathbf{p} \in \mathbb{R}^N$), since the results can be easily generalized to higher dimensions.

A. Symmetry maps

The following result introduces a symmetry map for the CS dynamics in port Hamiltonian form.

Proposition 4.1: The map $\Gamma(\mathbf{p},\mathbf{q}) = (\mathbf{p} + \alpha\mathbb{1}, \mathbf{q})$ for $\alpha \in \mathbb{R}$ is a symmetry map for the port-Hamiltonian dynamics (10).□

For the CS dynamics with potential in its original form (1)-(2), the symmetry map is slightly different, as shown next.

Proposition 4.2: The map $\Gamma(\mathbf{x},\mathbf{v},t) = (\mathbf{x} + \alpha\mathbb{1}t + \beta\mathbb{1}, \mathbf{v} + \alpha\mathbb{1}, t)$ for $\alpha \in \mathbb{R}$ is a symmetry map for the CS dynamics with potential (1)-(2).□

B. Lie group of invariance transformations

As introduced in Section II-C, the Lie group of invariance transformations [18], [19] are a particular type of symmetry maps with the form $\hat{\mathbf{z}} = \mathbf{z} + \varepsilon\eta(\mathbf{z},t) + O(\varepsilon^2)$. The following result introduces the infinitesimal of the CS model in the original form.

Proposition 4.3: The map $\eta(\mathbf{z},t) = \eta(\mathbf{x},\mathbf{v},t) = \alpha\left[\mathbb{1}^T, 0^T\right]^T + \left[\beta\mathbb{1}^T, \alpha\mathbb{1}^T\right]^T$, for all $\alpha,\beta \in \mathbb{R}$ is an infinitesimal for the Lie group of invariance transformations corresponding to the CS dynamics in its original form.□

A similar result holds for CS model in port-Hamiltonian form, where the time dependence of the infinitesimal map is no longer present.

Proposition 4.4: The map $\eta(\mathbf{z}) = \eta(\mathbf{p},\mathbf{q}) = \alpha\left[\mathbb{1}^T, 0^T\right]^T$, for all $\alpha \in \mathbb{R}$ is an infinitesimal for the Lie group of invariance transformations corresponding to the CS dynamics in port Hamiltonian form. □

C. Conserved quantities

The port-Hamiltonian representation has the advantage of providing at least one quantity that is conserved, namely the Hamiltonian. In addition to the Hamiltonian function, there are other quantities that are conserved. The following results introduce such quantities for both the original and port-Hamiltonian representation of the CS particle dynamics.

Proposition 4.5: The quantity $\mathbb{1}^T\mathbf{v}$ is conserved by the CS dynamics (1), that is, $\mathbb{1}^T\dot{\mathbf{v}} = 0$, for all $t \geq 0$.□

We can show similar results in the case of the port-Hamiltonian representation. We will make use of the Casimir functions which represent the conserved quantities for port-Hamiltonain systems.

Proposition 4.6: Any function of the form $\mathbf{C}(\mathbf{p},\mathbf{q}) = \alpha \mathbb{1}^T\mathbf{p} + \mathbf{u}^T\mathbf{q} + \beta$, where $\mathbf{u} \in Null(J)$, and $\alpha, \gamma \in \mathbb{R}$ is a conserved quantity for the CS dynamics in port-Hamiltonian form (10), where $\mathbf{z}^T = [\mathbf{p}^T, \mathbf{q}^T]$.

Remark 4.1: Note that in the 3 particle example, the matrix J is square and the null space of J and J^T is the same. In general this it is not true since $J \in \mathbb{R}^{N\times M}$, where $M = N(N-1)/2$. Hence, only the null space of J^T is given by $\{\alpha\mathbb{1}, \alpha \in \mathbb{R}\}$.

V. Learning interaction models and their dynamical properties

To demonstrate that we can indeed recover the theoretical results proved in the previous sections, we consider an example where twenty particles ($N = 20$) evolve according to the CS dynamics. We consider both the original and port-Hamiltonian representation of the CS dynamics. The particles operate in a two dimensional space, that is, the (relative) position and velocity vectors of each particle have dimension two. The training data were generated by simulating the CS model with parameter $\gamma = 0.15$, over the time interval $[0,40]$ sec, starting with random initial conditions in the interval $[0,10]$. A realization of the CS, simulation results is shown in Figure 2, where we plot the particle speed (norm of the velocity vector). The structure of the

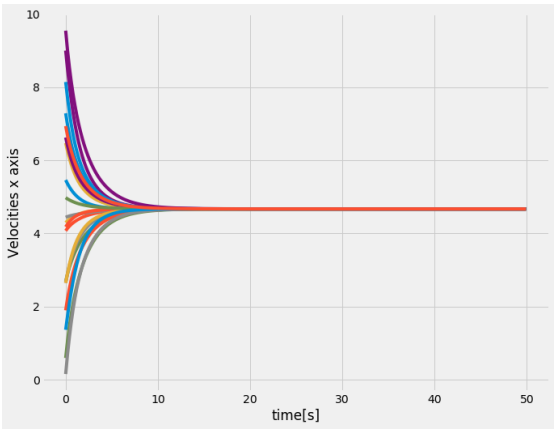

Fig. 2: Particle speed over time $\|v_i(t)\|, i \in \{1,\dots,N\}$

time series used for training is $\mathbf{z}^T = [x^T, y^T, v_x^T, v_y^T]$, where $x, y, v_x, v_y \in \mathbb{R}^N$. In the port-Hamiltonian representation, the structure is slightly different, namely $\mathbf{z}^T = [p_x^T, p_y^T, q_x^T, q_y^T]$, where $q_x, q_y \in \mathbb{R}^{\frac{N(N-1)}{2}}$, and $p_x^T, p_y^T \in \mathbb{R}^N$. The computation of the gradients and Jacobians was done using automatic differentiation. The learning problems were implemented using the Python package Autograd [20] and the deep learning platform Pytorch [21] featuring automatic differentiation.

A. Particle interaction model

Our first task is to recover the interaction model between particles. We consider the port-Hamiltonian representation case, without potential, which can be obtained by approximating the spring potential function with zero, by appropriately choosing the parameters of the potential function. Using the port-Hamiltonian formalism, this task translates to learning the constitutive equation for a generalized damper. In particular we learn $F_{ij} = g(q_{ij}^2; w)\dot{q}_{ij}$ that describes the force acting between two particles i, j, where q_{ij} is the relative position between two the particles. We choose the map g to be a neural network (NN) with one hidden layer of size 12, whose output is given by $y = W^{[1]}\left(\tanh\left(W^{[0]}u + b^{[0]}\right)\right) + b^{[1]}$, where the weight exponents denote the layer number. Hence we have a total of 37 parameters. Note that we can add a ReLu type of activation on the last layer to impose a non-negative output of the NN. To learn the parameters of the map g, we solve the optimization problem $\min_w \frac{1}{n}\sum_{i=1}^n \|\mathbf{z}(t_i) - \hat{\mathbf{z}}(t_i; w)\|^2$, where n is the number of time samples, $w = \{W^{[0]}, b^{[0]}, W^{[1]}, b^{[1]}\}$ is the set of optimization variables, $\hat{\mathbf{z}}(t_i)$ are time samples of the solution of (10) with the resistive term defined by $R(q) = g(\|q\|^2; w)$, and no potential between particles. The initial positions and velocity were uniformly drawn from the interval $[0,10]$. We used the Autograd package and its Adam algorithm implementation to solve the least square problem introduced above. The optimization error was set to terminate when a value smaller that 10^{-5} is reached. We compared the trained interaction model with the "real" interaction model, as shown in Figure 3. We limited ourselves to a relative

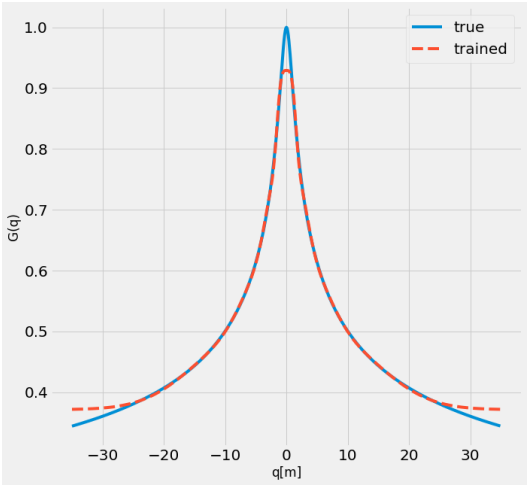

Fig. 3: Comparison between the "real" (blue) and the trained (dotted red) particle interaction models

distance between $[-35,35]$ since this was the maximum distance the particles reached between them over time. The MSE between the trained and the "real" interaction curves over the interval [-35,35] is 1.3×10^{-4}. We note that there is some miss-match near zero due to the fact that the particles never got close enough. Next, we tested the interaction model on data not used in the training but whose initial conditions

have similar statistics as the initial conditions of the training data. The $MSE_{test}(t_i) = \frac{1}{N}\|\mathbf{z}(t_i) - \hat{\mathbf{z}}(t_i)\|^2$, where $\mathbf{z}(t_i)$, $\hat{\mathbf{z}}(t_i)$ designate samples of the time series obtained with the "true" and learned interaction models, respectively, is shown in Figure 4. We note that the prediction error stabilizes to a reasonable small value.

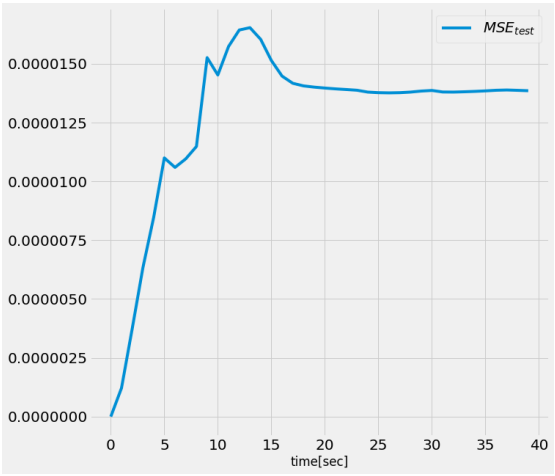

Fig. 4: The MSEs of the velocity vectors for test data

B. Lie group of invariance transformations

The Lie group of transformations ψ has a structure of the form $\psi(\mathbf{z}) = \mathbf{z} + \varepsilon\eta(\mathbf{z};w) + o(\varepsilon^2)$, where $\eta(\mathbf{z};w)$ is the infinitesimal of the transformation [18], [19]. We consider a linear parameterization of the form $\eta(\mathbf{z};w) = A\mathbf{z} + b$ and the goal is to find the parameters of the infinitesimal by solving the optimization problem $\min_{A,b} \frac{1}{n\times N}\sum_{i=1}^{n}\left\|\frac{\partial\eta}{\partial\mathbf{z}}(\mathbf{z}^{(i)})f(\mathbf{z}^{(i)}) - \frac{\partial f}{\partial\mathbf{z}}(\mathbf{z}^{(i)})\eta(\mathbf{z}^{(i)})\right\|^2$, where n denotes the total number of vector samples. The optimization problem was solved for the port-Hamiltonain representation, using the Adam algorithm and Autograd to compute the gradient of the cost function using automatic differentiation. To improve the speed of the optimization algorithm we computed offline the values for the maps $f(\mathbf{z})$ and $\frac{\partial f}{\partial\mathbf{z}}$ at each sample of the training data $\mathbf{z}^{(i)}$. We generated 50 time series describing the CS dynamics over the time interval $[0,40]$ sec, using $M = 50$ initial condition vectors uniformly drawn from $[0,10]$, generating roughly 5000 data samples. We stopped the optimization process when the MSE loss function reached $MSE_{train} = 1.1\times10^{-4}$. As sanity check, we looked at the structure of the learned A and b. The structure of b is according to what we would expect: same values for the first half of the vector (of roughly 1.8049) and small values for the second half ($< 10^{-4}$). The entries of A although small, they were not zero, which may be a result of the fact that we limited the number of optimization iterations.

The test data were generated randomly, in a similar way as the training data, using a time interval $[0,80]$ sec, generating roughly 10000 samples. The longer time interval checks the time extrapolation as well. As metric we used the MSE applied on trajectories this time. We have two types of trajectories. The first type denoted by $\mathbf{z}(t)$ is a trajectory generated by solving the CS differential equations, with initial conditions obtained by applying the *learned* symmetry transformation to the initial conditions of the test data. The second type, denoted by $\hat{\mathbf{z}}(t)$ is obtained by applying the learned symmetry map on the test data itself. Formally, we define the metric $MSE_{test} = \frac{1}{n\times M\times N}\sum_{i=1}^{M}\sum_{j=1}^{n}\|\mathbf{z}^{(i)}(t_j) - \hat{\mathbf{z}}^{(i)}(t_j)\|^2$, where n is the number of time samples per time series, M is the number of time series, and $\mathbf{z}^{(i)}(t_j)$ is the vector of position and velocity coordinates at time t_j of the time series i. We obtained the following MSE for the test data: $MSE_{test} = 6.2\times10^{-4}$. We computed also the MSE evolution over time for the trajectories, where the averaging was taken over the time series indices (M of them) and entries of the state vector, but not over time as well. The result is shown in Figure 5. We note that that prediction

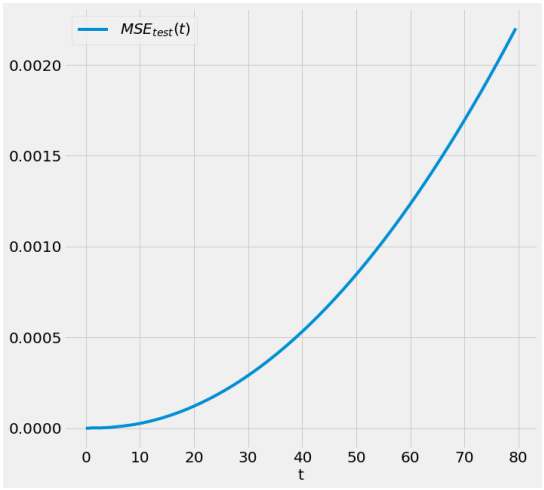

Fig. 5: MSE particle velocities over time

error accumulates over time, which most likely comes from the fact that the learned symmetry map was not exact, due in part to the limited number of optimization iterations.

C. Symmetry maps

We repeat the learning process for the discrete symmetry case, using this time the CS model in its original form. We search for a map Γ so that $\hat{\mathbf{z}} = \Gamma(\mathbf{z},t)$ is a solution of the CS ODE $\dot{\mathbf{z}} = f(\mathbf{z})$, as well. We assume that the time remains unchanged by the symmetry, hence no map for the time is included. We consider a linear parameterization of the symmetry map, $\Gamma(\mathbf{z},t) = A\mathbf{z} + bt + c$, which includes time dependence as well. To learn the map parameters, we solve the following optimization problem $\min_{A,b,c} \frac{1}{n\times N}\sum_{i=1}^{n}\left\|\frac{\partial\Gamma}{\partial\mathbf{z}}(\mathbf{z}^{(i)},t_i)f(\mathbf{z}^{(i)}) + \frac{\partial\Gamma}{\partial t}(\mathbf{z}^{(i)},t_i) - f\left(\Gamma(\mathbf{z}^{(i)},t_i)\right)\right\|^2$. We used the Pytorch deep learning platform to implement the optimization process, using the same Adam algorithm as in the case of the Lie symmetries group. Pytorch features automatic differentiation as well, but has the advantage that can be used with graphics processing units (GPUs), when the optimization problem can be parallelized. To give an

idea of why Pytorch can be more effective when scaling up the problem in number of particles, Figure 6 shows a comparison between the average time for an optimization iteration of the Pytorch Adam's algorithm when using CPU and GPUs, as a function of the number of particles. We note that unlike the CPU case, when using GPUs the average iteration time grows linearly with the number of particles. In addition, in terms of average iteration time when using the CPU, Pytorch is superior to Autograd: 4.9 sec for Autograd versus 2.2 sec for Pytorch for the 20 particle case, for the same number of training samples. We use a similar

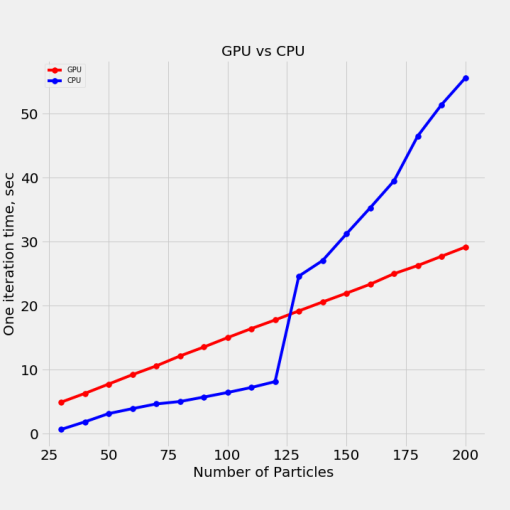

Fig. 6: Average time for an Adam iteration when using CPU (blue curve) and GPUs (red curve), as a function of the number of particles

strategy to generate training and test data, as in the case of the Lie symmetries group. We stopped the optimization algorithms when the MSE reached 5.5×10^{-5}. The MSE for the test data was 0.007. The test data MSE as a function of time is shown in Figure 7. The same phenomenon of error accumulation over time is noticed as in the case of the Lie symmetries group.

D. Conservation laws

In this section we demonstrate that we can recover conserved quantities as introduced in Proposition 4.6, whose statement can be easily generalized to the two dimensional case. Namely, the Casimir functions have the form $C(p_x, p_y, q_x, q_y) = \alpha_x \mathbb{1}^T p_x + \alpha_y \mathbb{1}^T p_y + u_x^T q_x + u_y^T q_y + \beta$, $\forall \alpha \in \mathbb{R}$ and $\forall u_x, u_y \in Null(J)$. To learn the Casimir function $C(\mathbf{z}; w)$, we solve the optimization problem $\min_w \frac{1}{n} \sum_{i=1}^{n} \left\| J^T \frac{\partial C}{\partial p_x}(\mathbf{z}^{(i)}) \right\|^2 + \left\| J^T \frac{\partial C}{\partial p_y}(\mathbf{z}^{(i)}) \right\|^2 + \left\| J \frac{\partial C}{\partial q_x}(\mathbf{z}^{(i)}) \right\|^2 + \left\| J \frac{\partial C}{\partial q_y}(\mathbf{z}^{(i)}) \right\|^2$. We considered two type of parameterizations: a linear parameterization given by $C(p_x, p_y, q_x, q_y) = a_x^T p_x + a_y^T p_y + b_x^T q_x + b_y^T q_y$ and a nonlinear parameterization given by a neural network with a hidden layer of size $2N + N(N-1)$ defined by $C(\mathbf{z}) = W^{[1]}\left(\tanh\left(W^{[0]}\mathbf{z} + b^{[0]}\right)\right) + b^{[1]}$. For the linear case, the partial derivatives of C were hard-coded since

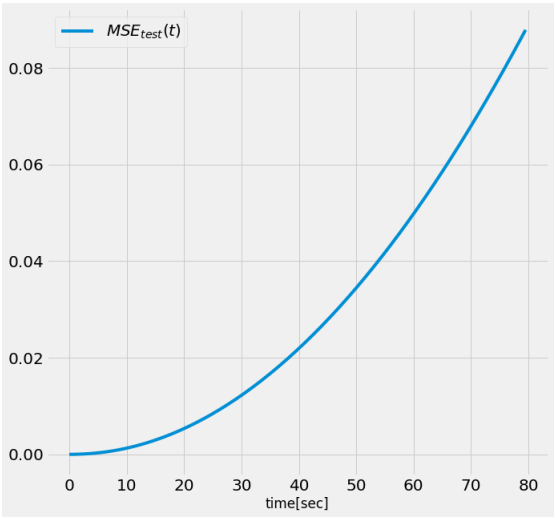

Fig. 7: MSE particle velocities over time

they are simple and do not depend on the training data. We did use though Autograd to compute analytically the gradient of the loss function. We initialized the optimization variables randomly, and we run the Adam algorithm for 2500 iterations with a fixed step of 0.001. Each iteration in the Adam algorithm takes roughly 3msec. For sanity check, we looked at the structure of the learned vectors a_x and a_y, whose entries are shown in Figure 8. We note that we indeed recovered the expected structure, namely $\alpha_x \mathbb{1}$ and $\alpha_y \mathbb{1}$.

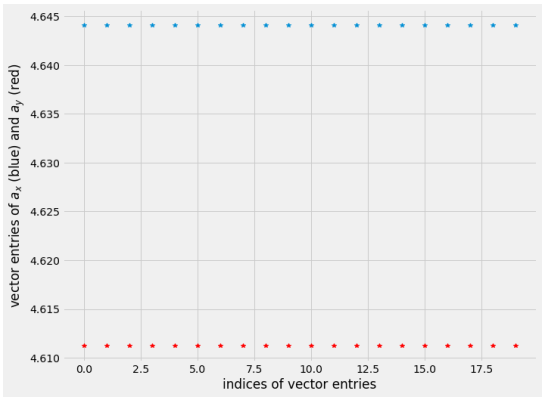

Fig. 8: Entries of vectors a_x and a_y

Another sanity check measure is to plot the evolution of the Casimir function as a function of time depicted in Figure 9, showing that it has a constant value of 1077.53. The value of the Casimir function depends on the initial conditions for both the training data and the optimization variables. We repeated the learning process for a nonlinear (neural network) parameterization. In this case, we used Autograd to construct functions that can be called to compute the partial derivatives of the Casimir function. In addition of these Jacobians, we used Autograd to generate the gradient of the loss function. As a result, each iteration of the Adam algorithm becomes slower, namely 3 sec. We run the algorithm for

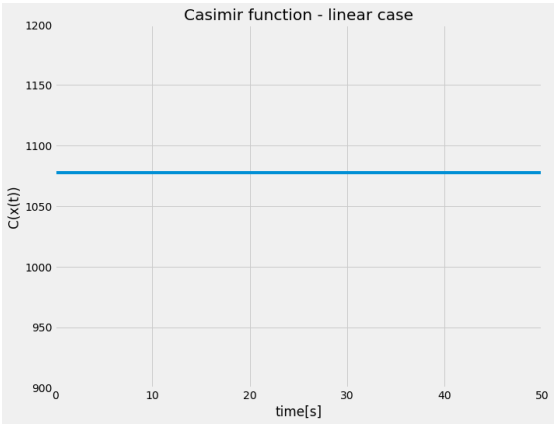

Fig. 9: Casimir function over time for the linear parametrization

500 iterations starting from random initial conditions for the optimization variables, selected around the zero value. The Casimir function for the nonlinear parametrization, computed at each point on the state trajectory is shown in Figure 10, where we notice that the function takes a constant value of approximately -0.4641

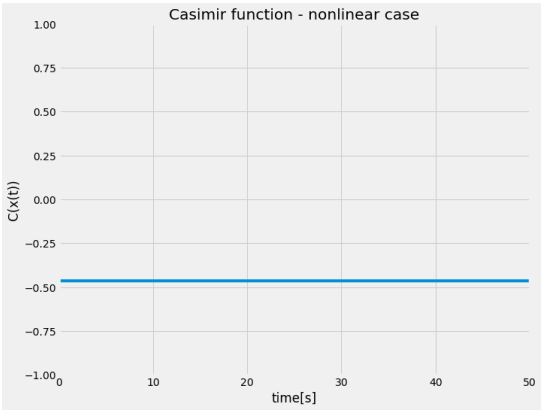

Fig. 10: Casimir function over time for the nonlinear parameterization

VI. Conclusions

In this paper we proposed a framework based on port-Hamiltonian modeling formalism, aimed at learning interaction models between particles and dynamical properties such as trajectory symmetries and conservation laws of ensembles(or swarms) using large-scale optimization approaches. We built upon the Cucker-Smale particle interaction model, which we represented in a port-Hamiltonian form, and for which we re-discovered the interaction model, and learned the dynamical properties that were previously proved analytically. Our approach can potentially be used for discovering novel particle interaction rules which can lead to new cooperative control system laws. The future steps will include scaling up the problem to a very large number of particles, considering non-linear parameterizations for the symmetry maps, and re-casting the learning tasks in a form that is compatible with parallel GPU computations on deep learning platforms. In addition, we will explore if symbolic computation of Jacobians together with automatic differentiation of the loss function will lead to a significant decrease in time per optimization iteration.

References

[1] J.S. Baras. A fresh look at network science: Interdependent multigraphs models inspired from statistical physics. In *Proceedings of the 6th International Symposium on Communication, Control and Signal Processing*, pages 497–500, May 2014.

[2] F. Lu, M. Zhong, S. Tang, and M. Maggioni. Nonparametric inference of interaction laws in systems of agents from trajectory data. *arXiv preprint arXiv:1812.06003*, 2018.

[3] S.L. Brunton, J.L. Proctor, and J.N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[4] J. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007.

[5] I. Matei, J. de Kleer, and R. Minhas. Learning constitutive equations of physical components with constraints discovery. In *2018 Annual American Control Conference (ACC)*, pages 4819–4824, June 2018.

[6] I. Matei, J. De Kleer, M. Zhenirovskyy, and A. Feldman. Learning constitutive equations of physical components with predefined feasibility conditions. In *2019 American Control Conference (ACC)*, pages 922–927, July 2019.

[7] Z. Mao, Z. Li, and G.E. Karniadakis. Nonlocal flocking dynamics: Learning the fractional order of pdes from particle simulations. *arXiv preprint arXiv:1810.11596*, 2018.

[8] A.J. van der Schaft and B.M. Maschke. Port-hamiltonian systems on graphs. *SIAM Journal on Control and Optimization*, 51(2):906–937, 2013.

[9] A.J. van der Schaft. Port-hamiltonian systems: an introductory survey. In M. Sanz-Sole, J. Soria, J.L. Varona, and J. Verdera, editors, *Proceedings of the International Congress of Mathematicians Vol. III*, number suppl 2, pages 1339–1365. European Mathematical Society Publishing House (EMS Ph), 2006.

[10] A.J. van der Schaft and D. Jeltsema. Port-hamiltonian systems theory: An introductory overview. *Foundations and Trends® in Systems and Control*, 1(2-3):173–378, 2014.

[11] J. Cervera, A.J. van der Schaft, and A. Baños. Interconnection of port-hamiltonian systems and composition of dirac structures. *Automatica*, 43(2):212–225, 2007.

[12] A. Mouchet. Applications of Noether conservation theorem to Hamiltonian systems. *Annals of Physics*, 372, 12 2015.

[13] J. Schwichtenberg. Physics from symmetry. Springer, 2015.

[14] J.S. Baras. Group invariance and symmetries in nonlinear control and estimation. *Nonlinear Control in the Year 2000, A. Isidori, F. Lamnabhi-Lagarrigue, W. Respondek (Edts.)*, 1:137–171, December 2000.

[15] C.W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH computer graphics*, volume 21, pages 25–34. ACM, 1987.

[16] J.A. Carrillo, M. Fornasier, G. Toscani, and F. Vecil. *Particle, kinetic, and hydrodynamic models of swarming*. Birkhäuser Boston, Boston, 2010.

[17] J.A. Carrillo, S. Martin, and V. Panferov. A new interaction potential for swarming models. *Physica D: Nonlinear Phenomena*, 260:112–126, 2013.

[18] G.W. Bluman and S.C. Anco. Symmetry and integration methods for differential equations. *Applied Mathematical Sciences*, (154), 2002.

[19] A.F. Cheviakov G. W. Bluman and S.C. Anco. Applications of symmetry methods to partial differential equations. *Applied Mathematical Sciences*, (163), 2010.

[20] D. Maclaurin, D. Duvenaud, M. Johnson, and J. Townsend. Autograd. `https://github.com/HIPS/autograd`, 2018.

[21] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. 2017.