ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 93–99*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 514.8

ПОРТ-ГАМИЛЬТОНОВЫ СИСТЕМЫ: РАСПОЗНАВАНИЕ СТРУКТУРЫ И ПРИЛОЖЕНИЯ

© 2024 г. В. Н. Сальников[a], *

[a]*ЦНРС и Университет города Ла-Рошель, 17042 Ла-Рошель, Проспект Мишеля Крепо, Франция*

**E-mail: vladimir.salnikov@univ-lr.fr*

Поступила в редакцию 11.08.2023
После доработки 10.08.2023
Принята к публикации 01.10.2023

В данной работе мы продолжаем рассматривать задачу восстановления порт-Гамильтоновой структуры для произвольной системы дифференциальных уравнений. Мы дополняем предыдущую работу по этой теме, объясняя выбор и подробности применения алгоритмов машинного обучения. Мы также объясняем, какие возможности открывает такой подход для потенциально нового определения канонических форм и классификации систем дифференциальных уравнений.

Ключевые слова: геометризация механики, порт-Гамильтоновы системы, методы машинного обучения для ОДУ

DOI: 10.31857/S0132347424020121 **EDN:** RNXHNU

1. ВВЕДЕНИЕ / МОТИВАЦИЯ

Данная статья – продолжение серии работ в рамках глобального проекта по “геометризации механики”. Под этим термином, не обязательно общепринятым, мы понимаем различные подходы, основанные изначально на результатах и конструкциях из дифференциальной или обобщённой геометрии, которые применяются к качественному анализу уравнений, описывающих механические системы. В этот проект входит разработка геометрического формализма, релевантного для максимально широкого класса задач механики. Он также посвящен поиску или построению разумных стратегий моделирования механических систем и численного решения полученных уравнений. Основная прикладная цель проекта – предложить наиболее полный “набор инструментов”, оптимальный в “бытовом” смысле, то есть обеспечивающий достаточно точное решение таких задач с разумными затратами вычислительных ресурсов. Мы приводим некоторый обзор таких методов в [1]: в нем мы показываем, что конструкции из обобщенной и градуированной геометрии играют важную роль.

В [2, 3] мы сформулировали ряд открытых вопросов и задач, потенциально решаемых современными методами компьютерной алгебры; построение порт-Гамильтоновой структуры заданных уравнений и изучение порт-Гамильтовых систем (ПГС – набор Гамильтоновых систем, соединённых по определённым правилам) было одной из них. В [4] мы подробно рассмотрели Гамильтонову составляющую ПГС, которая была существенно связана с результатами из симплектической и Пуассоновой геометрии. Про порты мы лишь упомянули, что их можно удобно описывать с помощью графов, к которым и применяются алгоритмы анализа. В данной статье мы остановимся на этом вопросе подробнее, а именно объясним релевантность подхода к анализу с помощью машинного обучения и приведем некоторые аргументы в пользу выбранных алгоритмов. Мы также остановимся чуть подробнее на некоторой новой философии анализа внутренней структуры произвольных систем дифференциальных уравнений, естественным образом вытекающей из формализма ПГС.

В секции 2 мы напомним определения и обозначения из ПГС, сформулируем их важные свойства. В секции 3 мы опишем “опыт” применения различных подходов к анализу ПГС. Мы закончим статью упоминанием прямых приложений данного подхода, связанных с геометрическими интеграторами в моделировании сложных систем. Мы также обсудим потенциальные приложения для нового подхода к определению канонической формы систем дифференциальных уравнений.

2. ПОРТ-ГАМИЛЬТОНОВЫ СИСТЕМЫ

Идея порт-Гамильтоновых систем достаточно проста и изящна: для нескольких консервативных механических систем, описываемых в рамках классической Гамильтоновой динамики, рассмотрим взаимодействие, заданное некоторыми силами – эти

силы назовём портами. Насколько нам известно, впервые такой подход был подробно описан в [5], с инженерной точки зрения, а затем пересмотрен в [6] с некоторым описанием классификации физики портов. Достаточно общепринятая терминология – называть каждую из таких систем вместе с её входящими и исходящими портами *узлом*.

Напомним, как эта конструкция формализуется математически. Каждый узел можно записать следующим образом:

$$\dot{\mathbf{x}} = (J(\mathbf{x}) - R(\mathbf{x}))\frac{\partial H}{\partial \mathbf{x}} + w(\mathbf{x})\mathbf{u}. \qquad (2.1)$$

Здесь $\mathbf{x}$ – векторная переменная, обозначающая состояние узла, $J(\mathbf{x})$ – кососимметричная матрица, описывающая симплектическую или Пуассонову структуру (см. [4]), $H(\mathbf{x})$ – Гамильтониан; $R(\mathbf{x}),w(\mathbf{x})$ и $\mathbf{u}$ соответствуют внутренним или внешним портам.

Порт-Гамильтонова система составляется из таких узлов соединением через порты: внешние переменные одних узлов зависят от внутренних для других. Для дальнейшего важно понимать ключевое свойство ПГС: несколько узлов, соединённых вместе, снова являются узлом.

Понятно, что “сборка” порт-Гамильтоновой системы из модулей – это формально простая техническая конструкция. Здесь мы обсуждаем задачу, обратную к ней: зная, что система дифференциальных уравнений

$$\dot{\mathbf{X}} = \mathbf{F}(\mathbf{X}) \qquad (2.2)$$

получена как порт-Гамильтонова, восстановить её структуру.

Напомним, что алгоритм, который мы начали объяснять в [4], выглядел (не вдаваясь в подробности) следующим образом.

Алгоритм

Входные данные: система дифференциальных уравнений вида (0.2).

1. Восстановление графа связности ПГС, то есть идентификация узлов и связей между ними.

2. Разметка полученного графа, то есть идентификация или выбор симплектической/Пуассоновой структуры и соответствующего Гамильтониана.

3. Разметка связей между узлами, то есть идентификация портов.

Выходные данные: граф (множество узлов и портов ПГС), размеченный в обозначениях (0.1).

Пункты 2 и 3 были подробно изучены в [4] – после соответствующего анализа они сводятся к символьным вычислениям с дифференциальными формами, векторными и бивекторными полями. В предыдущей статье мы рассматривали примеры с помощью простейших инструментов на языке Python, упомянем здесь лишь, что пакет SageManifolds в SageMath ([7]) оказался удобным и более продвинутым для таких целей.

Что же касается п. 1, в предварительных вычислениях для “proof of concept” мы использовали совершенно прямолинейную реализацию алгоритмов, о которой скажем ниже, но уже тогда было понятно, что вопрос более интересный и заслуживает отдельного рассмотрения. Сформулируем некоторые свойства ПГС, повлиявшие на выбор алгоритма решения этой задачи.

Как уже было упомянуто, важное свойство ПГС – возможность их соединения:

- два соединённых или даже независимых узла могут рассматриваться как один;
- одна переменная может быть узлом;
- вся ПГС может считаться узлом;
- несколько ПГС, соединённых вместе – снова ПГС.

Это значит, что без дополнительных условий задача восстановления структуры ПГС по уравнению вида (2.2) имеет существенно неединственное решение.

Рассмотрим теперь всю ПГС как один узел, иными словами, конфигурацию, когда уравнение (2.2) имеет вид (2.1). В таком случае правые части системы имееют довольно специальный вид. Матрицы J и R отвечают за внутренние степени свободы узла – когда узел сообран из нескольких, они будут иметь блочную структуру: подматрицы меньшего размера будут сворачиваться с градиентом Гамильтониана, зависящего от некоторого подмножества переменных. Переменные w и $\mathbf{u}$, наоборот, будут “перемешивать” блоки и встречаться парами: внутренние для одного – внешние для другого подузла.

3. ПОДХОДЫ К ОПРЕДЕЛЕНИЮ СТРУКТУРЫ ГРАФА

В этой секции мы представим три различных подхода, которые мы рассматривали для определения структуры графа ПГС. Мы опишем в том числе и “неудачный” опыт, а именно стратегии, которые не привели к желаемому или оптимальному результату. В свете замечаний выше, мы объясним различия методов и причины возникающих сложностей. Мы постараемся сделать это с наименьшим количеством технических подробностей. Напомним, цель в сис-

теме уравнений вида (2.2) сгруппировать переменные в узлы и восстановить связи между ними.

Детерминистский подход

Описанная структуры матриц J и R кажется довольно "узнаваемой" – естественно возникает идея восстановить эту структуру *точно*, то есть сформулировать набор критериев, позволяющих классифицировать каждое слагаемое уравнения (2.2). Идея алгоритма могла бы быть следующая: в каждом уравнении системы (2.2) каждое из слагаемых отнести либо к переменным "своего" узла, либо к портам по некоторому признаку, затем проверить совместность результата с учётом Гамильтоновой структуры узлов.

На деле единственное решение, которое получилось формализовать – перебрать *все возможные комбинации* размеров узлов и заполнить их *всеми возможными способами* слагаемыми из (2.2). При этом шаг проверки совместности сводится к описанному в [4] восстановлению Пуассоновой или симплектической структуры (матрицы J) и подбору Гамильтониана, то есть аналитическому решению нелинейной системы дифференциальных уравнений. Только в отличие от оригинального подхода [4], разделение на порты и внутренние переменные в каждом случае будет зафиксировано, что систематически приводит именно к несовместным конфигурациям. Это же ограничение приводит к невозможности корректировать Гамильтониан, если он оказывается слишком сложным и/или сильно нелинейным для восстановления. Все эти явления были заметны на очень простых системах в малой размерности: достаточно рассмотреть гармонические осцилляторы с минимально нелинейным взаимодействием, порт-Гамильтонова структура для которых легко читается невооруженным глазом. Очевидно также, что для больших систем даже перебор размеров блоков становится неразумным. Таким образом даже до попыток настоящей программной реализации стало понятна ограниченность такого подхода.

Причина трудностей каждый раз одна и та же: неоднозначность представления системы вида (2.2) в виде ПГС. В [4] мы объяснили, что именно эта неоднозначность должна облегчать процесс, предоставляя свободу выбора для идентификации геометрических структур. Это же свойство оказалось удобно использовать и для построения узлов в двух методах, описанных далее.

Машинное обучение

Как мы уже не раз упомянули, по-настоящему работающие и удобные методы оказались связаны с довольно стандартными подходами машинного обучения с использованием нейронных сетей. Мы, естественно, не собираемся излагать здесь хоть сколько-то подробно идеи нейронных сетей: даже обзор литературы занял бы неразумно большое пространство. Ограничимся лишь напоминанием ставшего общепринятым подхода. Машинное обучение применяется в задачах принятия решений в случае, когда есть возможность проанализировать множество похожих задач в предварительной фазе "обучения" сети. При этом ответы на них могут быть известны (supervised learning) или нет (unsupervised).

В нашем случае ситуация для применения алгоритмов машинного обучения оказалась очень благоприятной: основным результатом [8], помимо приложений, является программный пакет PyPHS ([9]), позволяющий проводить распределённые вычисления сложных систем, построенных в форме ПГС. Первая его часть (собственно построение) сводится к генерации систем уравнений, соответствующих ПГС с *заданной физикой и топологией взаимодействий*. То есть для задачи восстановления структуры ПГС в нашем распоряжении есть пакет, быстро и эффективно строящий базу данных с известным ответом – эта база используется в двух методах, описанных ниже.

Классический матричный подход

Чтобы свести задачу распознавания структуры ПГС к стандартной задаче машинного обучения, нужно выбрать формат представления данных. В данном случае цель – распределить правые части (2.2) по матрицам в (2.1). Технически для этого нужно выбрать некоторый базис функционального пространства, в котором мы будем работать. Самый простой выбор, естественно, рассмотреть многочлены от переменных, описывающих состояние системы. Он, конечно, не отражает всего разнообразия нелинейностей, возникающих в динамических системах, но для иллюстрации оказывается поучительным.

Таким образом, фаза обучения состоит в построении систем вида (2.1) пакетом PyPHS без дальнейшего их решения. Пакет (с открытым кодом) позволяет зафиксировать все параметры и ограничить пространство функций, а небольшая надстройка над ним обеспечивает построение нужного количества систем с заданным разбросом характеристик. Для нейронной сети входными параметрами будут: топология ПГС (узлы и связи между ними), коэффициенты всех многочленов, заменяющих функции в матрицах и в свёртках. Выходные данные – ана-

логичные коэффициенты разложения правых частей (2.2).

Для данной статьи мы начали с матриц с постоянными коэффициентами (то есть с примеров, соответствующих симплектическим структурам в узлах), а градиент Гамильтониана и другие функции ограничили квадратичными. Даже такой простой класс примеров оказался достаточно репрезентативным. Данные обрабатывались с помощью конволюционных нейронных сетей (convolution neural networks – некоторые подробности в приложении А).

Тестовый эксперимент проводился на небольшой выборке из 1500 элементов, для систем из 10 уравнений с возможной топологией, включающей 4, 5 или 6 рёбер. Даже для таких относительно небольших систем стало понятно, что выбор гиперпараметров играет важную роль, а количество параметров модели, необходимых для разумных результатов (потери при обучениии ~0.13, потери валидации ~0.007) достигает сотен тысяч. На стандартном компьютере (core i5, 2.6GHz) такой счёт занимает примерно 50 минут.

Основная проблема данного подхода в том, что для реальных систем (большого размера с сильными нелинейностями) количество параметров и время исполнения растёт очень быстро и требует либо огромных вычислительных мощностей, либо существенного упрощения модели. Таким образом, приведённый матричный подход в принципе работает как быстрое решение, не требующее существенных трудозатрат, но на практике он недостаточно удобен.

Машинное обучение на графах

Разумной альтернативой описанному выше матричному подходу оказался метод анализа связности графов и группировки вершин в них. Мы снова не пытаемся изложить всё множество алгоритмов анализа структуры графов, известных в современной литературе. Приведём лишь все необходимые “детали” для построения соответствующего алгоритма.

Как и раньше, у нас есть возможность построить базу данных для обучения с помощью пакета PyPHS, с абсолютно аналогичными входными и выходными данными. Но первый шаг алгоритма теперь существенно отличается: вместо заполнения матриц функциональными слагаемыми, мы сохраним лишь данные о взаимодействии переменных. Иными словами, на вход подаются не конретные функции из правых частей (2.2), а информация, от каких переменных (и как) они зависят. На выходе же тоже не требуются конкретные выражения для правых частей (2.1), а только их структура в смысле ПГС.

На языке графов это значит, что исходный строится явно по правым частям системы уравнений (2.2); все переменные X_i объявляются независимыми вершинами, их пары связаны ребром, если одна их них присутствует в правой части уравнения для другой, такие рёбра обычно будут направленными. Конечный граф состоит из объединённых групп вершин с большим количеством попарных связей (узлы ПГС), рёбра же (снова направленные) соответствуют портам. Переход от начального (большого) к конечному (существенно меньшему) графу как раз и осуществляется недетерминистскими методами.

У каждого специалиста по анализу данных есть свой любимый пакет для кластеризации точек или свёртки (pooling) графов/сетей – это, как правило, довольно простые нейронные сети с небольшим количеством слоёв. В нашем случае в основе конечного решения лежит пакет MapEquation ([10]). В остальном подход снова довольно стандартный и похож на предыдущий: обучить сеть на базе данных из PyPHS, проверить на элементах из выборки и вне выборки. Чтобы не перегружать текст техническими подробностями, мы не будем приводить здесь конкретных цифр и параметров. К тому же сравнивать параметры сетей, полученных в этом и предыдущем пункте, не имеет большого смысла, поскольку они решают несколько разные задачи. Анализ графов даёт исключительно представление об узлах ПГС, в то время как матричный подход начинает также решать (а иногда и решает полностью) задачу разметки, восстанавливая Гамильтонову структуру. Таким образом понятно, что подход с графами работает значительно быстрее.

Отметим лишь, что в отличие от предыдущего подхода, данный совершенно не меняется в зависимости от степени нелинейности системы. Он зависит исключительно от её размера, а именно от комбинации количества переменных и связей, что обеспечивает разумные свойства для масштабирования. Это и повлияло на конечный выбор данного подхода для решения задачи восстановления структуры ПГС.

Отметим также, что подход оказался очень гибким. Например, построение исходного графа просто по бинарному свойству наличия или отсутствия соответствующей переменной в правой части может быть уточнено: в зависимости от вида или сложности слагаемого можно присваивать некоторым или всем рёбрам нетривиальный вес. К тому же некоторые типы взаимодействий можно “насильно” относить к узлам или портам – это реализуется добавлением слоя в нейронную сеть.

4. ВМЕСТО ЗАКЛЮЧЕНИЯ

В этой статье мы продолжили изучение порт-Гамильтоновых систем, а именно описали важную часть алгоритма восстановления их структуры – разбиение системы на узлы. Основная цель работы была убедиться, что методы машинного обучения могут применяться для решения данной задачи с минимальными усилиями по адаптации готовых программных пакетов. Методом проб и ошибок мы установили что наиболее разумным в этом смысле подходом является анализ графа связности системы дифференциальных уравнений.

Напомним, что исходная мотивация для представления произвольной системы уравнений в виде ПГС была связана с последующим использованием геометрических интеграторов — численных методов, сохраняющих некоторую геометрическую структуру — для проведения компьютерных вычислений. Таким образом, подход потенциально позволяет расширить область применения как классических ([11, 12]), так и более продвинутых ([13, 14, 15]) интеграторов. Иными словами, реализация описанных алгоритмов позволила превратить теорию ПГС из абстрактной “игры обозначений” в часть большого пакета для моделирования сложных систем с использованием параллельных и распределенных вычислений.

Разработка и описание данного подхода также натолкнули нас на пару технических замечаний, которые мы хотим изучить более подробно в последующих работах.

Во-первых, в разделе о нейронных сетях мы привели некоторые численные значения потерь при обучении и валидации. Важно заметить, что в этой статистике учитывалось точное восстановление структуры исходной ПГС. Но замеченные “ошибки” не значат, что алгоритм сработал совсем неверно. На самом деле, возможно, была построена другая структура ПГС для заданной системы уравнений, что с точки зрения приложений для распределённых вычислений, тоже будет подходящим решением. Иными словами, вполне возможно, критерии “успеха” решения задачи можно существенно ослабить.

Во-вторых, на графах мы рассмотрели использование сетей, которые обучались на порт-Гамильтоновых системах, но концептуально этап обучения можно было бы совсем пропустить. В таком случае система всё равно разобьётся на узлы, которые, правда, могут быть малопригодны для восстановления Гамильтоновой структуры в смысле [4]. Возможно, тем не менее, искомое решение может быть достигнуто не полным переобучением сети, а лишь незначительной её модерацией добавлением одного “ПГС-слоя” или изменением оптимизируемой функции.

Мы также пришли к некоторому концептуальному замечанию, которое уже упомянули несколько раз: мысль анализировать ПГС с помощью стандартных негеометрических алгоритмов можно развернуть. Если на вход нейронной сети, обученной на ПГС, подать систему уравнений общего вида, на выходе будет некоторая структура, похожая на взаимодействие физических систем. Такое восстановление “скрытой физики” системы, с одной стороны, снова позволит применить методы распределённого вычисления. С другой стороны, его можно воспринимать как подход к классификации систем дифференциальных уравнений. Стандартные методы анализа обычно начинаются с линеаризации и рассмотрения добавочных слагаемых особого вида. Мы сами использовали похожий подход в разделе о матричных методах. Понятно, что они очень сильно зависят от выбора исходной системы координат. Вспомним задачи построения нормальных форм Гамильтоновых систем, когда к симплектическим структурам и простым Гамильтонианам добавляются нелинейные слагаемые с малыми параметрами. Если же посмотреть на более сложную систему с точки зрения ПГС, то вместо исходной простой Гамильтоновой системы, будет рассматриваться набор потенциально сильно нелинейных, но консервативных систем, возможно с совершенно не физической “энергией”. А “возмущение”, опять же не обязательно малое, будет соответствовать их взаимодействию.

ПРИЛОЖЕНИЕ А. НЕКОТОРЫЕ ТЕХНИЧЕСКИЕ ПОДРОБНОСТИ

Данные, описанные в матричном подходе секции 3, обрабатывались полной конволюционной нейронной сетью, состоящей из двух конволюционных слоёв, активационного (activation) и отбрасывающего (dropout) слоя, а также слоя нормализации выборки. В качестве оптимизационной была выбрана функция Адама, а функция потерь – среднеквадратичная.

Оптимизатор Адама корректирует веса по правилу:

$$w_t = w_{t-1} - \eta \frac{m_t}{\sqrt{\nu_t} + \varepsilon},$$

где w_t — веса t-го слоя, ν_t – параметр шага.

$$m_t = \frac{m_t}{1 - \beta_1^t},$$

$$\nu_t = \frac{\nu_t}{1-\beta_2^t},$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t,$$

$$\nu_t = \beta_2 \nu_{t-1} + (1-\beta_2) g_t^2,$$

где g_t – градиент по текущей мини-выборке, β_t – гиперпараметры с исходнымии значениями, близкими к 1; переменные m_t, v_t суть среднее и смещённое отклонение градиентов функций потерь.

Количество фильтров для каждого конволюционного слоя было выбрано равным 64128. Количество “тренируемых” параметров равно 175162.

БЛАГОДАРНОСТИ

Я благодарен Всеволоду Сальникову за ценные советы по поводу машинного обучения и описание возможностей и проблем использования нейронных сетей. Некоторые предварительные численные эксперименты, мотивировавшие написание этой статьи, были выполнены Дарьей Лозиенко. Я благодарен Антуану Фалезу за многочисленные обсуждения возможностей пакета PyPHS. Я особенно благодарен Сергею Александровичу Абрамову за бесконечное терпение во время подготовки этой статьи и анонимному рецензенту за полезные замечания.

ИСТОЧНИКИ ФИНАНСИРОВАНИЯ

Данная работа была частично поддержана проектом CNRS PEPS Jeune Chercheur GraNum 2.0, а также проектом ACI Университета Ла-Рошели.

СПИСОК ЛИТЕРАТУРЫ

1. *Salnikov V., Hamdouni A., Loziienko D.*, Generalized and graded geometry for mechanics: a comprehensive introduction // Mathematics and Mechanics of Complex Systems. 2021. V. 9. № 1. 2021.
2. *Salnikov V., Hamdouni A.* Geometric integrators in mechanics: The need for computer algebra tools, Tr. Tret’ei Mezhdun. Konf. “Computer algebra” (Proc. 3rd Int. Conf. Computer Algebra), Moscow, 2019.
3. *Salnikov V.N., Hamdouni A.* Differential geometry and mechanics: A source for computer algebra problems // Program. Comput. Software. 2020. V. 46. P. 126–132.
4. *Salnikov V., Falaize A., Lozienko D.* Learning port-Hamiltonian systems: Algorithms // Comput. Math. Math. Phys. 2023. V. 63. P. 126–134.
5. *Paynter H.M.* Analysis and Design of Engineering Systems // MIT Press, Cambridge, Massachusetts, 1961.
6. *A. van der Schaft*. Port-Hamiltonian systems: an introductory survey // Proceedings of the International Congress of Mathematicians, Madrid, 2006.
7. Sage Manifolds – Differential geometry and tensor calculus with SageMath, https://sagemanifolds.obspm.fr
8. *Falaize A.* Modélisation, simulation, génération de code et correction de systèmes multi-physiques audios: Approche par réseau de composants et formulation hamiltonienne à ports, // PhD thesis, Télécommunication et Électronique de Paris, Université Pierre et Marie Curie, 2016.
9. Modeling, simulation and code-generation of multiphysical Port-Hamiltonian Systems in Python: https://github.com/pyphs/pyphs
10. *Edler D., Holmgren A. Rosvall M.*, Infomap – Network community detection using the MapEquation framework, https://www.mapequation.org/infomap/
11. *Hairer E., Lubich C., Wanner G.*, Geometric Numerical Integration // Springer Series in Computational Mathematics, 2006.
12. *Razafindralandy D., Hamdouni A., Chhay M.*, A review of some geometric integrators // Advanced Modeling and Simulation in Engineering Sciences, SpringerOpen. 2018. V. 5 № 1. P. 16.
13. *Razafindralandy D., Salnikov V., Hamdouni A., Deeb A.* Some robust integrators for large time dynamics // Advanced Modeling and Simulation in Engineering Sciences. 2019. V. 6. № 5.
14. *Cosserat O.*, Symplectic groupoids for Poisson integrators // Journal of Geometry and Physics, 2023. V. 186.
15. *Cosserat O., Laurent-Gengoux C., Salnikov V.* // Numerical Methods in Poisson Geometry and their Application to Mechanics, Preprint: arXiv:2303.15883.

PORT-HAMILTONIAN SYSTEM: STRUCTURE RECOGNITION AND APPLICATIONS

© 2024 V. N. Salnikov[a]

[a] *University of La Rochelle, Av. Michel Crépeau, La Rochelle, 17042, Paris, France*

In this paper, we continue to consider the problem of recovering the port-Hamiltonian structure for an arbitrary system of differential equations. We complement our previous study on this topic by explaining the choice of machine learning algorithms and discussing some details of their application. We also consider the possibility provided by this approach for a potentially new definition of canonical forms and classification of systems of differential equations.

Keywords: geometrization of mechanics, port-Hamiltonian systems, and machine learning methods for ODEs