

Implementation of Neural Models for Accurate Prediction of Robot Trajectories in Hamiltonian Spaces: A Graph Neural Network Approach

Implementierung von neuronalen Modellen zur präzisen Vorhersage von Trajektorien in Hamiltonschen Räumen: Ein Ansatz mit Graph-Neuronalen Netzwerken

Master thesis by Denis Andrić (Student ID: 2486004)

Date of submission: October 6, 2024

1. Review: Georgia Chalvatzaki

2. Review: An Thai Le
Darmstadt



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Denis Andrić, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 6. Oktober 2024

D. Andrić

Contents

1	Introduction	8
2	Background	12
2.1	Classical Mechanics	12
2.1.1	Harmonic Oscillator	13
2.1.2	Analytical solution of Ordinary Differential Equations	14
2.1.3	Numerical solution of Ordinary Differential Equations	17
2.1.4	Solving the harmonic oscillator	19
2.1.5	Langrange and Hamiltonian Equations	20
2.1.5.1	Lagrange Equations	20
2.1.5.2	Hamiltonian Equations	24
2.2	Neural Network Architectures	25
2.2.1	Deep forward Network - Multilayer Perceptron	25
2.2.2	Recurrent Neural Networks	26
2.2.3	Activation functions and Loss functions	27
2.2.4	Optimizers	29
2.2.5	Backpropagation of Neural Models	30
3	Methods	31
3.1	New paradigm - NeuralODE	31
3.2	Recurrent time steppers	32
3.3	Hamiltonian Neural Network	32
3.4	Graph Neural Network	33
3.4.1	Convolutional Graph Neural Network	33
3.4.2	Gated Attention Network	34
3.5	Graph Neural Hamiltonian Network	35
3.6	Gated Recurrent Graph Hamiltonian Neural Network - GRUGHNN	35

4	Datasets	36
4.1	Oscillator	36
4.2	N-body problem	37
4.2.1	Symplectic numerical methods to solve N-body problem	38
4.2.2	twobody problem	39
4.2.3	Three body problem	41
4.3	N Pendulum	44

List of Figures

1.1	PINN model for Laplace Equation	9
1.2	HNN model with integration solver step	11
2.1	Multilayer perceptron[2]	26
2.2	Recurrent neural unit(many to many)[7]	27
2.3	Gated recurrent unit[15]	28
2.4	ReLU activation	28
2.5	Tanh activation	29



List of Tables

+

Abstract

The introduction of the adjoint method for NeuralODEs[4] has created new ways for physics-based trajectory predictions. In the subsequent chapters of this thesis, we look at deep learning models and methods targeted at predicting physical systems' behavior. We use the guiding light of Hamiltonian equations to dive into systems such as the harmonic oscillator and the three-body problem. We are intrigued with exploring graph neural networks' ability to be used in physics informed models: in the N-body problem and in the N-pendulum. This work tests the ability of graph neural networks to generalize the behavior from single realizations of elements at the node level of the graph. In this way, we hope to exemplify how such models can capture highly dynamic physical processes.

1 Introduction

The intersection of physics and deep learning has never been more pronounced than it is today. With advancements in hardware and computational capabilities, we are now positioned to predict physical phenomena with unprecedented precision using deep learning techniques, particularly through physics-informed neural networks.

Physics-Informed Neural Networks (PINN) are neural networks that encode model equations, like Partial Differential Equations (PDE), as a component of the neural network itself[.]. For example let us take Laplace equation which is defined as

$$\Delta u(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega, \mathbf{x} \in R^2 \quad (1.1)$$

$$u = S \quad \mathbf{x} \in \partial\Omega \text{ Dirichlet boundary condition,} \quad (1.2)$$

$$\frac{\partial u}{\partial \mathbf{n}} = f(\mathbf{x}) \quad \mathbf{x} \in \partial\Omega \text{ Neumann boundary condition.} \quad (1.3)$$

The Operator Δ is called as Laplace Operator and it defines $\Delta = \sum_i \frac{\partial^2}{\partial^2 x_i}$.

To build a PINN model we will use an Neural Network which will learn the solution u and from that output we will calculate all needed derivatives, residuals and other parts for the optimization.

In figure 1 we can observe the architecture of such physics informed model. The Loss functions that we need to use to properly train our network are

$$\text{Loss}_s(\Theta) = \frac{1}{N} \sum_i^N (u_{\Theta}(\mathbf{x}_i) - u_i)^2, \quad (1.4)$$

$$\text{Loss}_r(\Theta) = \frac{1}{R} \sum_i^R \left(\frac{\partial u_{\Theta}(\mathbf{x}_i)}{\partial x_i} - f(\mathbf{x}_i) \right)^2, \quad (1.5)$$

$$\text{Loss}_l(\Theta) = \Delta u_{\Theta}, \quad (1.6)$$

$$\text{Loss}(\Theta) = \omega_s \text{Loss}_s + \omega_r \text{Loss}_r + \omega_l \text{Loss}_l. \quad (1.7)$$

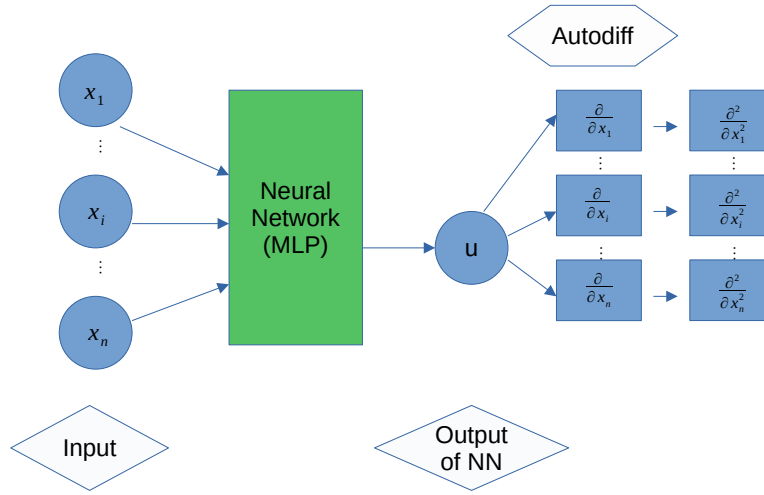


Figure 1.1: PINN model for Laplace Equation

The $\omega_s, \omega_r, \omega_l$, in this case are coefficients $0 \leq \omega \leq 1$ and they are there to improve optimization capabilities which is defined as

$$\Theta^* = \arg \min \text{Loss}(\Theta). \quad (1.8)$$

The Θ are learnable parameters which can be updated with Θ^* through optimization algorithm.

Such model wouldn't be possible without Automatic Differentiation[] which are offered from machine learning library `pytorch`{}. In this example we need big bunch of data and the optimization of the model could be very slow in addition if we don't have a computation capable hardware for it. Obtaining the data could be done only with exactly measurement or using some numerical solvers like FEM or Isogeometric Analysis. Don't

forget, that there are Poission Equation, Wave Equation which are dependent on time t . Those PDEs have mostly application in electrotechnical(Electro-magnetical fields) and civil Engineering(static in construction). In this thesis we won't work on the PDEs but similar to this topic we will try to train a model to predict the Dynamics of some physical phenomena and non-/holonomic systems of the robots.

The Dynamics for Robots are trivially defined as simple Second Level Ordinary Differential Equation[] .

$$\mathbf{M}\ddot{\mathbf{x}}(t) + \mathbf{D}(\dot{\mathbf{x}}(t)) + \mathbf{C}(\mathbf{x}(t)) = 0, \quad (1.9)$$

but there are other forms, as Langrange Equations of Motion if our system is holonomic[]. It is based on Lagrange Function which is based on Kinetik T and Potential U Energy

$$\mathcal{L} = T - V, \quad (1.10)$$

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = 0. \quad (1.11)$$

The Lagrange Equations are used in Delan[] Physics informed network and in simillar manner, but for us are most interesting to use Hamiltonian Equations of Motion. They work for holonomic and non-holonomic systems and they are capable to build First Level Ordinary Equation

$$\dot{x} = f(x, t) \quad (1.12)$$

or in Hamiltonian Form where $\mathcal{H} = T + V$

$$\dot{\mathbf{z}} = \mathbf{J} \frac{\partial \mathcal{H}}{\partial \mathbf{z}} \quad (1.13)$$

where $\mathbf{z} = [\mathbf{q}, \mathbf{p}]^T$ and $\mathbf{J} = \begin{bmatrix} 0 & \mathbf{I}_n \\ -\mathbf{I}_n & 0 \end{bmatrix}$

One of such model is introduced in the [] and it is called HNN. it architecture you can observe in figure 1. We will make it more interesting introducing the Graph Neural Network and try to use it as a base of our architecture.

In this thesis we will discuss what are Hamiltonian equations and how to obtain them. In our research we stumbled upon the fact, that there are no quality datasets for training the Neural Networks which are based on physics. With this fact we brew 4 popular datasets which are based on Hamiltonian Equations of Motion.

Next we made new and revisit old neural architectures and test NeuralODE as new Nerual Paradigm into treaining the solutions of ODEs.

In the experiments we tested the capabilites of Graph Neural Networks, how do they adapt

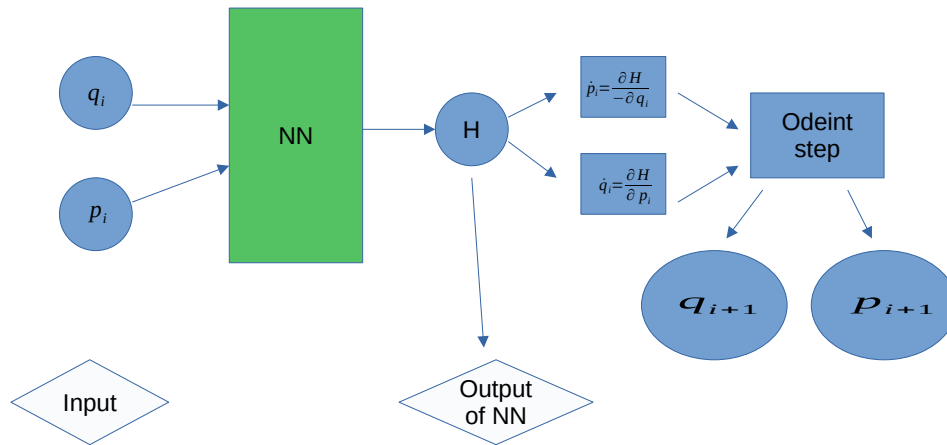


Figure 1.2: HNN model with integration solver step

to diverse training data and their possibility to reduce or increase a degree of freedom in the training. For example how similar is the movement between 3 and 4 bodied pendulum on the same trained set of neural parameters. We hope that we will give the insight in those topics.

2 Background

In this chapter, we revisit some of the important concepts from Classical Mechanics and further develop an understanding of common neural network architectures. This is a critical foundation which will be required to understand how Physics-Informed Neural Networks work, as discussed in greater detail in the Methods chapter.

2.1 Classical Mechanics

Classical Physics presents the basics for understanding the laws presiding over the behavior of objects with mass in our physical reality. This aspect of the science of Classical Mechanics describes how mass moves in space: motion is the result of forces applied to matter. Moving on, Sir Isaac Newton did develop laws that describe and predict the behavior of such objects, which today are called the Newton's Laws of Motion.[14]

- Newton's 1st law: Unless acted on by an outside force the natural motion of an object is constant velocity
- Newton's 2nd law: The effect of an applied force \mathbf{F} upon an object of mass m is to induce an acceleration \mathbf{a} such that

$$\mathbf{F} = m\mathbf{a}$$

If mass is constant we can introduce momentum \mathbf{p} .

$$\mathbf{F} = \frac{d\mathbf{p}}{dt}$$

Derivation of the momentum in time t is a force. With this equation we can obtain momentum as

$$\mathbf{p} = m\mathbf{v}$$

where \mathbf{v} is velocity.

- Newton's 3rd law: If an object applies a force \mathbf{F} on a second object, then the second object applies an equal and opposite force $-\mathbf{F}$ on the first object.
In the physics it is most famous law and it is often referenced as *actio = reactio* from latin.

One of the simplest systems that adheres to these laws is the harmonic oscillator.

2.1.1 Harmonic Oscillator

Harmonic Oscillator is system in classical physics which describes the situation where body is displaced from equilibrium position and it experience the restoring force \mathbf{F} proportional to its displacement.[] This system is described with Ordinary Differential Equation of second order :

$$m\ddot{\mathbf{x}}(t) = -k\mathbf{x}(t), \quad \mathbf{x}(0) = \mathbf{x}_0 \quad (2.1)$$

We will consider this system as one dimensional $\mathbf{x} = x$ in which the harmonic motion is along straight line, the motion is said to be simple harmonic motion [3]. To show the simple harmonic motion let us solve the equation.

First we will build the the linear system of equations to get the first order ordinary differential equation in form:

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{b}. \quad (2.2)$$

We are beginning with parameter z :

$$z_1 = x \quad (2.3)$$

$$z_2 = \dot{x} \quad (2.4)$$

$$\dot{z}_1 = \dot{x} = z_2 \quad (2.5)$$

$$\dot{z}_2 = \ddot{x} = -\frac{k}{m}z_1 \quad (2.6)$$

With this we got:

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \quad (2.7)$$

We got differential ordinary equation which is homogeneous ($\mathbf{b} = 0$). To solve this Problem we need to refresh the knowledge of solving the ODEs

2.1.2 Analytical solution of Ordinary Differential Equations

A Differential Ordinary Equation of the first order is defined as:

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{b}u, \quad \mathbf{z}(0) = \mathbf{z}_0 \quad (2.8)$$

Before we show analytical solution of mentioned linear system, let us solve first the simplest case:

$$\dot{z}(t) = az(t) + bu, \quad z(0) = z_0 \quad (2.9)$$

Every solution of ODE has homogeneous z_{hom} and particular solution z_{par}

$$z(t) = z_{hom} + z_{par}. \quad (2.10)$$

When we calculate de homogeneous solution we set $b = 0$ and we precede with simple integration.

$$\int \frac{dz}{z} = \int a dt \quad (2.11)$$

$$z_{hom} = c \exp(at) \quad (2.12)$$

To find particular solution we will create it with variation of the constants method. We define:

$$z_{part} = c(t) \exp(at) \quad (2.13)$$

and we will put it in the differential equation

$$\dot{z}_{part}(t) = az_{part}(t) + bu(t), \quad (2.14)$$

$$a \exp(at)c(t) + \exp(at)\dot{c}(t) = a \exp(at)c(t) + bu(t), \quad (2.15)$$

$$\dot{c}(t) = \exp(-at)bu(t). \quad (2.16)$$

After integration of \dot{c} we get:

$$z_{part} = c(0) \exp(at) + \int_0^t \exp(at - \tau)bu(\tau)d\tau \quad (2.17)$$

With inclusion of the Initial condition $z(0) = z_0$ we get final form of the solution

$$z = \underbrace{z_0 \exp(at)}_{\text{homogeneous}} + \underbrace{\int_0^t \exp(a(t - \tau))bu(\tau)d\tau}_{\text{particular}}. \quad (2.18)$$

Let so go back and solve the equation 2.8.

Obtaining the solution goes similar to simple case but it is vectorized.

The solution is made by homogeneous and particular solution as above.

$$\dot{\mathbf{z}} = \mathbf{z}_{hom} + \mathbf{z}_{part} \quad (2.19)$$

Lets assume from the example what we have done previously that our homogeneous solution is defined as:

$$\mathbf{z}_{hom} = \exp(\mathbf{A}t)\mathbf{c} \quad (2.20)$$

This is easy to prove trough Taylor series of the $\exp(\mathbf{A}t)$.

$$\exp(\mathbf{A}t) = \sum_{i=0}^{\infty} \frac{(\mathbf{A}t)^i}{i!} = \mathbf{I} + \mathbf{A}t + \mathbf{A}^2 \frac{t^2}{2!} + \mathbf{A}^3 \frac{t^3}{3!} + \mathcal{O}(t^4) \quad (2.21)$$

$$\frac{d \exp(\mathbf{A}t)}{dt} = \mathbf{A} + \mathbf{A}^2 t + \mathbf{A}^3 \frac{t^2}{2!} + \mathcal{O}(t^3) = \quad (2.22)$$

$$\mathbf{A} \left(\mathbf{I} + \mathbf{A}t + \mathbf{A}^2 \frac{t^2}{2!} \right) + \mathcal{O}(t^3) \quad (2.23)$$

$$\mathbf{A} \exp(\mathbf{A}t) \quad (2.24)$$

Lets put it in the equation

$$\dot{\mathbf{z}}_{hom} = \frac{d \exp(\mathbf{A}t)}{dt} \mathbf{c} = \quad (2.25)$$

$$\mathbf{A} \exp(\mathbf{A}t) \mathbf{c} = \quad (2.26)$$

$$\mathbf{A} \mathbf{z}_{hom} \quad (2.27)$$

and the homogeneous solution

$$\boxed{\mathbf{z}_{hom} = \exp(\mathbf{A}t)\mathbf{c}} \quad (2.28)$$

fulfills the equation.

With the variation of the constant method we obtain the particular solution:

$$\exp(\mathbf{A}t)\mathbf{c}(0) + \int_0^t \exp(\mathbf{A}(t - \tau))\mathbf{b}u(\tau)d\tau \quad (2.29)$$

With inclusion of the initial condition $z(0) = z_0$ we get final form of the solution

$$\mathbf{z} = \underbrace{\exp(\mathbf{A}t)\mathbf{x}_0}_{\text{homogeneous}} + \underbrace{\int_0^t \exp(\mathbf{A}(t-\tau))bu(\tau)d\tau}_{\text{particular}}. \quad (2.30)$$

In our work the most important part is homogeneous solution and to obtain that we should use standard practice to solving it.

- Find eigenvalues and eigenvectors- Every matrix with full rank is diagonalizable

$$\mathbf{A} = \mathbf{VDV}^{-1} \quad (2.31)$$

$$\mathbf{D} = \text{diag}(\{\lambda_1, \dots, \lambda_i, \dots, \lambda_n\}) \text{ Eigenvalues} \quad (2.32)$$

$$\mathbf{V} = [\mathbf{v}_1 | \dots | \mathbf{v}_i | \dots | \mathbf{v}_n] \text{ Eigenvectors} \quad (2.33)$$

It is easily obtained through characteristic polynomial $p(\lambda) = \det(\mathbf{A} - \lambda \mathbf{I}) = 0$

- To find general solution we need to transform the $\dot{\mathbf{z}} = \mathbf{A}\mathbf{z}$

$$\dot{\mathbf{z}} = \mathbf{VDV}^{-1}\mathbf{z} \quad (2.34)$$

$$\underbrace{\mathbf{V}^{-1}\dot{\mathbf{z}}}_{\dot{\boldsymbol{\Theta}}} = \mathbf{D}\underbrace{\mathbf{V}^{-1}\mathbf{z}}_{\boldsymbol{\Theta}} \quad (2.35)$$

$$\dot{\boldsymbol{\Theta}} = \mathbf{D}\boldsymbol{\Theta} \quad (2.36)$$

This system is now trivial to solve.

- Set it in general solution $\Theta_i = c_i \exp(\lambda_i t)$ In case of repeated eigenvalue we should use $\Theta_i = c_i \frac{t^r}{r!} \exp(\lambda_i t)$ where r is number of repetitions. It will came up from the eigenvectors.
- back-substitute \mathbf{z}

$$\mathbf{z} = \mathbf{V}\boldsymbol{\Theta} = \sum_{i=1}^n \mathbf{v}_i \Theta_i \quad (2.37)$$

$$\mathbf{z} = \sum_{i=1}^n \mathbf{v}_i \underbrace{c_i \exp(\lambda_i t)}_{\Theta_i} \quad (2.38)$$

- apply initial conditions

$$\mathbf{z}(0) = \sum_{i=1}^n \mathbf{v}_i c_i = \mathbf{V}\mathbf{c} = \mathbf{z}_0 \quad (2.39)$$

$$\mathbf{c} = \mathbf{V}^{-1}\mathbf{z}_0 \quad (2.40)$$

In this procedure we obtained the homogeneous solution. The particular solution is in some cases trivial but in some not solvable analytically due to $\mathbf{b}u(t)$. It is possible that it is nonlinear function. In those cases we use numerical methods to obtain the solutions.

2.1.3 Numerical solution of Ordinary Differential Equations

For every every ODE there is solution, sometimes there is no analytical solution but we can obtain numerical one trough numerical integrators. Still numerical solutions are only approximations of the analytical solution. Due to their stability we differ two categories of one-step methods and that are explicit and implicit methods.

- explicit methods are trivial to calculate because the next step is dependent only on the previous step:

$$x(t_{i+1}) = x(t_i) + h\Phi(x(t_i)), t_i) \quad (2.41)$$

- implicit methods are challenging because the next step is dependent on itself.

$$x(t_{i+1}) = x(t_i) + h\Phi(x(t_{i+1})), t_{i+1}) \quad (2.42)$$

To obtain the step it is needed to know function Φ and solving a next step trough some method like for example Einstein-Raphson method.

To assure the stability and correctness of the methods we need to set some properties. The properties that the every one-step method should have, are Consistency and Convergence.

The Consistency are proven over local discretisation error. Let say that for $(x, y) \in \mu = \mu(\eta)$ is a solution for the ODE $\mu' = f(\eta, \mu)$, $\mu(x) = y$, then the discretisation error is

$$le(x, y; h) = \mu(x + h) - \mu x - h\Phi(x, y; h) \quad (2.43)$$

and discretisation error per step

$$\Delta(x, y; h) = \frac{le(x, y; h)}{h}. \quad (2.44)$$

A one-step method is consistent if

$$\lim_{h \rightarrow 0} \Delta(x, y; h) = 0. \quad (2.45)$$

For the method to be convergent need to be consistent. The general definition of convergency is

$$\lim_{h \rightarrow 0} \Phi(x, y; h) = f(x, y) \quad (2.46)$$

To prove this we should look for the consistency and convergency rank.

- Consistency rank p

$$||\Delta(x, y; h)|| \leq Kh^p \quad (2.47)$$

- Convergency rank p

$$E(h) = \max_{j=0,1,\dots,N} ||y_j - y(x_j)|| \text{ global discretisation error} \quad (2.48)$$

$$\lim_{h \rightarrow 0} E(h) = 0 \text{ Convergence} \quad (2.49)$$

$$E(h) \leq Kh^p \quad (2.50)$$

This properties are highly important to know because we will use for the experiments mostly the Runge-Kutta methods which are based on those properties. Most important for us are

- Euler method

$$x_{i+1} = x_i + hf(x_i, t_i) \quad (2.51)$$

- Runge-Kutta 4 Method(RK4)

$$K_1 = f(x_i, t_i) \quad (2.52)$$

$$K_2 = f(x_i + \frac{h}{2}K_1, t_i + \frac{h}{2}) \quad (2.53)$$

$$K_3 = f(x_i + \frac{h}{2}K_2, t_i + \frac{h}{2}) \quad (2.54)$$

$$K_4 = f(x_i + hK_3, t_i + h) \quad (2.55)$$

$$x_{i+1} = x_i + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \quad (2.56)$$

- Dormand- Prince method is most accurate method for obtaining the solutions of ODEs[1]. It is widely used in experiments.

2.1.4 Solving the harmonic oscillator

In one of the previous subsections we obtained the first order ODE for harmonic oscillator and we described how it should be solved.

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}, \mathbf{z}_0 = \begin{bmatrix} x_0 \\ v_0 \end{bmatrix} \quad (2.57)$$

Through the system Matrix $A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & 0 \end{bmatrix}$ we got complex eigenvalues, which are $\lambda_1 = i\sqrt{\frac{k}{m}}$ and $\lambda_2 = -i\sqrt{\frac{k}{m}}$.

General solution of this problem is

$$z_1 = x = c_1 \exp(i\sqrt{\frac{k}{m}}t) + c_2 \exp(-i\sqrt{\frac{k}{m}}t) \quad (2.58)$$

With applying of the euler formula:

$$\exp(iat) = \cos(at) + i \sin(at) \quad (2.59)$$

We get simplified solution

$$z_1 = x = (c_1 + c_2) \cos\left(\sqrt{\frac{k}{m}}t\right) + (c_1 - c_2)i \sin\left(\sqrt{\frac{k}{m}}t\right) \quad (2.60)$$

$$z_2 = \dot{x} = -(c_1 + c_2)\sqrt{\frac{k}{m}} \sin\left(\sqrt{\frac{k}{m}}t\right) + (c_1 - c_2)\sqrt{\frac{k}{m}}i \cos\left(\sqrt{\frac{k}{m}}t\right). \quad (2.61)$$

Now we can apply initial conditions

$$x(0) = x_0 = c_1 + c_2, \quad (2.62)$$

$$\dot{x}(0) = v_0 = i(c_1 - c_2). \quad (2.63)$$

The velocity in our cases is an real number which we can assume that constants c_1 and c_2 are equal. This simplifies our equations and we can substitute $x_0 = c_1 + c_2$

$$z_1 = x = x_0 \cos \left(\sqrt{\frac{k}{m}} t \right), \quad (2.64)$$

$$z_2 = \dot{x} = -x_0 \sqrt{\frac{k}{m}} \sin \left(\sqrt{\frac{k}{m}} t \right) \quad (2.65)$$

The is harmonic at we can se that clearly because it is periodic and we can read an angle velocity of periodic movement $\omega = \sqrt{\frac{k}{m}}$,

$$z_1 = x = x_0 \cos (\omega t) \quad (2.66)$$

$$z_2 = \dot{x} = -x_0 \omega \sin (\omega t) \quad (2.67)$$

2.1.5 Langrange and Hamiltonian Equations

In robotics, research and model creation of the robot models are not only one-bodied system. Mostly they are are rigid-body systems like a manipulator which has N degrees of freedom. Every of the joints and links have own properties like mass, friction, etc.

The Kinematic of the model and irreversibility is possible to calculate with mathematic methods. Kinematic is just establishing the function which with correct input like angles of joints q give as an output in form of target position of the endeffector. This easy easy to establish with Denavit–Hartenberg convention. On the other hand determining Dynamics is not trivial task. The Dynamic creates trajectory which is in form of ODE.

There is one most popular method to solve and establish Dynamics of the model and that are Lagrange Equations.

2.1.5.1 Lagrange Equations

Langrage Equations are derived from D’Alambert Principle which is an alternative to the Newton’s second law[14]:

$$\mathbf{F} = m\mathbf{a} \quad (2.68)$$

In the case of the rigid body:

$$\mathbf{F}_i = \sum_i m_i \mathbf{a}_i \quad (2.69)$$

To derive Lagrangian Equations which describes dynamics we need first to define constraints. Let us define homonomic constraints. For N bodies $B = \{\mathbf{r}_1, \dots, \mathbf{r}_i, \dots, \mathbf{r}_N\}$; $\mathbf{r}_i = \mathbf{r}_i(q_1, \dots, q_i, \dots, q_n, t)$

$$\boxed{G_k(\mathbf{r}_1, \dots, \mathbf{r}_i, \dots, \mathbf{r}_n, t) = 0 \text{ for } k = 1, \dots, M} \quad (2.70)$$

Where the q_i are generalized coordinates and the parameters are also constrained as $3 \cdot M \cdot N = n$.

In case of two dimensional N-Pendulum some of that constraints would look like:

$$G_i := ||\mathbf{r}_i - \mathbf{r}_{i+1}|| - d_i = 0 \quad (2.71)$$

This says that the distance between two neighboring joints are constant. holonomic constraint defines the relation between bodies i on some configuration space \mathcal{A} . Lets go back to the second newton law and lets apply holonomic constraint. With holonomic constraint we can spilt \mathbf{F} on constraint force \mathbf{F}^c and applied force \mathbf{F}^a .

$$\mathbf{F} = \mathbf{F}^c + \mathbf{F}^a \quad (2.72)$$

Now we will apply virtual Work principle, which comes from orthogonality of virtual displacement and constraint forces [18]

$$\boxed{\sum_i \mathbf{F}_i^c \cdot \underbrace{\delta \mathbf{r}_i}_{\text{virtual displacement}} = 0} \quad (2.73)$$

to the constrained Newton's second law we get following formula which is D'Alembert principle[8]:

$$\mathbf{F}_i^c = \mathbf{F}_i - \mathbf{F}_i^a \quad (2.74)$$

$$\sum_i \mathbf{F}_i^c \cdot \delta \mathbf{r}_i = \sum_i (\mathbf{F}_i - \mathbf{F}_i^a) \cdot \delta \mathbf{r}_i \quad (2.75)$$

$$\sum_i (\mathbf{F}_i - m_i \ddot{\mathbf{r}}_i) \cdot \delta \mathbf{r}_i = 0 \text{ D'Alembert Principle} \quad (2.76)$$

If we need virtual displacement for the generalized coordinates:

$$\delta \mathbf{r}_i = \sum_j^n \frac{\partial \mathbf{r}_i}{\partial q_j} \cdot \delta q_j \quad (2.77)$$

Now we can derive Langrange Equations applying last equation to D’Alambert principle:

$$\begin{aligned}
\sum_i^N \mathbf{F}_i \cdot \delta \mathbf{r}_i &= \\
\sum_i^N \mathbf{F}_i \sum_j^n \frac{\partial \mathbf{r}_i}{\partial q_j} \cdot \delta q_j &= \\
\sum_j^n \delta q_j \underbrace{\sum_i^N \mathbf{F}_i \frac{\partial \mathbf{r}_i}{\partial q_j}}_{Q_j} &= \boxed{\sum_j^n \delta q_j \cdot Q_j} \\
\sum_i^N m_i \ddot{\mathbf{r}}_i \cdot \delta \mathbf{r}_i &= \\
\sum_i^N m_i \ddot{\mathbf{r}}_i \sum_j^n \frac{\partial \mathbf{r}_i}{\partial q_j} \cdot \delta q_j &= \sum_j^n \delta q_j \sum_i^N m_i \ddot{\mathbf{r}}_i \frac{\partial \mathbf{r}_i}{\partial q_j} \\
\text{with substitution of: } \frac{\partial \dot{\mathbf{r}}_i}{\partial \dot{q}_j} &= \frac{\partial}{\partial \dot{q}_j} \left(\frac{d\mathbf{r}_i}{dt} + \sum_i^N \frac{\partial \mathbf{r}_i}{\partial q_j} \dot{q}_j \right) = \frac{\partial \mathbf{r}_i}{\partial q_j} \\
\sum_i^N m_i \ddot{\mathbf{r}}_i \sum_j^n \frac{\partial \mathbf{r}_i}{\partial q_j} \cdot \delta q_j &= \sum_j^n \delta q_j \sum_i^N m_i \ddot{\mathbf{r}}_i \frac{\partial \mathbf{r}_i}{\partial \dot{q}_j} \\
\text{with: } \sum_i^N \frac{d}{dt} m_i \dot{\mathbf{r}}_i \frac{\partial \mathbf{r}_i}{\partial \dot{q}_j} &= \sum_i^N m_i \ddot{\mathbf{r}}_i \frac{\partial \mathbf{r}_i}{\partial \dot{q}_j} + m_i \dot{\mathbf{r}}_i \frac{d}{dt} \frac{\partial \mathbf{r}_i}{\partial \dot{q}_j} \\
\sum_j^n \delta q_j \sum_i^N m_i \ddot{\mathbf{r}}_i \frac{\partial \mathbf{r}_i}{\partial \dot{q}_j} &= \sum_j^n \delta q_j \sum_i^N \frac{d}{dt} \left(m_i \dot{\mathbf{r}}_i \frac{\partial \mathbf{r}_i}{\partial \dot{q}_j} \right) - m_i \dot{\mathbf{r}}_i \frac{\partial \dot{\mathbf{r}}_i}{\partial \dot{q}_j}
\end{aligned}$$

We get this equation

$$\boxed{\sum_i^N m_i \ddot{\mathbf{r}}_i \cdot \delta \mathbf{r}_i = \sum_j^n \delta q_j \left[\frac{d}{dt} \frac{\partial}{\partial \dot{q}_j} \left(\sum_i^N \frac{1}{2} m_i \dot{\mathbf{r}}_i^2 \right) - \frac{\partial}{\partial \dot{q}_j} \left(\sum_i^N \frac{1}{2} m_i \dot{\mathbf{r}}_i^2 \right) \right]} \quad (2.78)$$

With those two made equations we can finally put it together in D'Alembert Principle formula:

$$\sum_j^n \delta q_j \left[\frac{d}{dt} \frac{\partial}{\partial \dot{q}_j} \left(\sum_i^N \frac{1}{2} m_i \dot{\mathbf{r}}_i^2 \right) - \frac{\partial}{\partial q_j} \left(\sum_i^N \frac{1}{2} m_i \dot{\mathbf{r}}_i^2 \right) - Q_j \right] = 0 \quad (2.79)$$

This Equation is almost complete, we need to define the energies and their specifications

- Kinetic Energy - Energy possessed by bodies due to their motion.
The Formula for kinetic energy is

$$T(q, \dot{q}) = \sum_i^N \frac{1}{2} m_i \dot{\mathbf{r}}_i^2 \quad (2.80)$$

- Potential Energy - Energy held by body because of its relative position to other bodies or other target.
There is many types of potential energy:
Pendulum - $U(q = h) = mgh$
Oscillator - $U(q) = \frac{kq^2}{2}$

In the Formula \mathbf{Q} is generalized force and it is created from the gradient of the potential energy and it is conservative :

$$\mathbf{Q}_j = - \sum_i^N \nabla_{\mathbf{r}_i} U(q) \frac{\partial \mathbf{r}_i}{\partial q_j} = - \frac{\partial U}{\partial q_j} \quad (2.81)$$

Potential Energy is only dependent on position of the bodies $\frac{\partial U}{\partial \dot{q}} = 0$.

The Lagrange Equations for holonomic constraints and conservative force is defined as:

$$\frac{d}{dt} \frac{\partial}{\partial \dot{q}_j} (T - U) - \frac{\partial}{\partial q_j} (T - U) = 0 \quad (2.82)$$

$$\frac{d}{dt} \frac{\partial}{\partial \dot{q}_j} \mathcal{L} - \frac{\partial}{\partial q_j} \mathcal{L} = 0 \quad (2.83)$$

\mathcal{L} is called Lagrangian.

2.1.5.2 Hamiltonian Equations

To derive Hamiltonian Equations we need an Lagrangian \mathcal{L} . We have two famous derivation of Langrangian and that are:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{q}} = \dot{\mathbf{p}} \quad (2.84)$$

$$\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}} = \mathbf{p} \quad (2.85)$$

Through Legendre Transformation we can obtain Hamiltonian \mathcal{H}

$$\mathcal{H}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{p}, t) = \sum_i^N \dot{\mathbf{q}}_i \cdot \mathbf{p}_i - \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}, t) \quad (2.86)$$

With total derivation[16] we get

$$d\mathcal{H} = \sum_i^N \left[\dot{\mathbf{q}} d\mathbf{p}_i + \mathbf{p}_i d\dot{\mathbf{q}}_i - \frac{\partial \mathcal{L}}{\partial \mathbf{q}_i} d\mathbf{q}_i - \frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}_i} d\dot{\mathbf{q}}_i \right] - \frac{\partial \mathcal{L}}{\partial t} dt \quad (2.87)$$

$$= \sum_i^N [\dot{\mathbf{q}} d\mathbf{p}_i + \mathbf{p}_i d\dot{\mathbf{q}}_i - \dot{\mathbf{p}}_i d\mathbf{q}_i - \mathbf{p}_i d\dot{\mathbf{q}}_i] - \frac{\partial \mathcal{L}}{\partial t} dt \quad (2.88)$$

$$= \sum_i^N [\dot{\mathbf{q}} d\mathbf{p}_i - \dot{\mathbf{p}}_i d\mathbf{q}_i] - \frac{\partial \mathcal{L}}{\partial t} dt \quad (2.89)$$

$$(2.90)$$

On the other hand we want hamiltonian which is dependent on canoical moment and generalized coordinates $\mathcal{H} = \mathcal{H}(\mathbf{q}, \mathbf{p}, t)$ In that case the total derivation of Hamiltonian \mathcal{H} is

$$d\mathcal{H} = \sum_i^N \left[\frac{\partial \mathcal{H}}{\partial \mathbf{q}_i} d\mathbf{q}_i + \frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} d\mathbf{p}_i \right] + \frac{\partial \mathcal{H}}{\partial t} dt \quad (2.91)$$

With comparing both equivalent derivations we can define the equations of motion:

$$\dot{\mathbf{p}} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} \quad (2.92)$$

$$\dot{\mathbf{q}} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}} \quad (2.93)$$

$$\frac{\partial \mathcal{H}}{\partial t} = -\frac{\partial \mathcal{L}}{\partial t} \quad (2.94)$$

Hamiltonian equations are very useful because in the case $\frac{\partial \mathcal{H}}{\partial t} = 0$ our Total Energy $\mathcal{H} = E = T + U$ is conserved and often we can derive time independent motion.

2.2 Neural Network Architectures

At the beginning of our research we experimented with feedforward networks and recurrent networks.

2.2.1 Deep forward Network - Multilayer Perceptron

Feedforward networks like Multilayer perceptron approximates a certain function $\mathbf{f}(\mathbf{x})$. As MLP function we will show it as a mapping $\mathbf{f}(\mathbf{x}|\Theta)$.

The main goal of feedforward network is to fit the model so that the we get best approximation. Fit the model means to learn the parameters of the model trough optimizing procedure.

Multilayer perceptron is fully connected feedforward neural network. The nodes of the MLP are often called neurons and they build a layer. A minimal MLP has three consecutive layers: which form complete bipartite graph.

Mathematical expression of one neuron:

$$y = \sigma\left(\sum_i w_i \cdot x_i + b\right) \quad (2.95)$$

Mathematical expression of the feedforward layer network:

$$\mathbf{h}^{(k+1)} = \sigma\left(\mathbf{W}^{(k)}\mathbf{h}^{(k)} + \mathbf{b}^{(k)}\right) \quad (2.96)$$

and we can see graphical representation in 2.1 In this equation σ is called activation function, \mathbf{W} as weights and \mathbf{b} as bias which are our parameters $\Theta = \{\mathbf{W}, \mathbf{b}\}$.

The activation function in the network introduces non linearity to improve approximation capability.

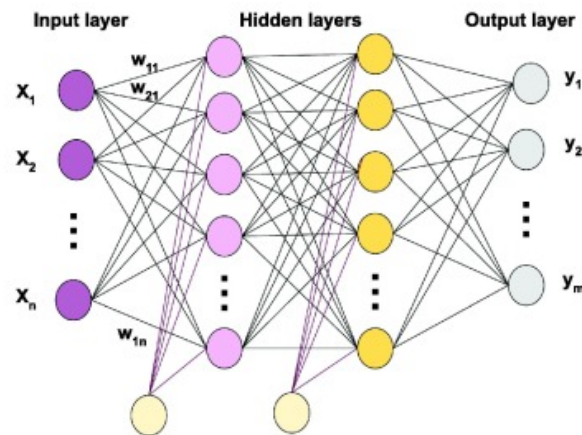


Figure 2.1: Multilayer perceptron[2]

2.2.2 Recurrent Neural Networks

Recurrent Neural Networks are a very interesting concept of Neural Network architecture. It is mostly used for speech recognition but for a time series[13].

The architecture of this Neural Network depends on the data we work with. There are three types which we can use[5]:

- many to many
This type is used for speech recognition where we use a dataset which has sentences and it outputs some values which could be understood from another model. For example translation or video captioning
- one to many
used for time series data from one input we can create recurrence that represents a time point with new output. From one input we get more outputs. Time-series can be for example music generation.
- many to one
from many inputs we get only one output. This is a very good model for example spam detection

Let say that we have a sequence $S = \{\mathbf{x}^0, \mathbf{x}^1, \dots, \mathbf{x}, \dots, \mathbf{x}^n\}$. the first element \mathbf{h}^0 will be initialized as 0. The architecture of the RNN-cell[19] described in mathematical formulation

$$\begin{aligned}\mathbf{h}^{(i)} &= \sigma(\mathbf{W}_{hh} \cdot \mathbf{h}^{(i-1)} + \mathbf{W}_{hx} \mathbf{x}^{(i-1)} + \mathbf{b}_h) \\ \mathbf{y}^{(i)} &= \sigma(\mathbf{W}_{yh} \cdot \mathbf{h}^{(i)} + \mathbf{b}_y)\end{aligned}$$

and graphical appearance is shown in 2.2

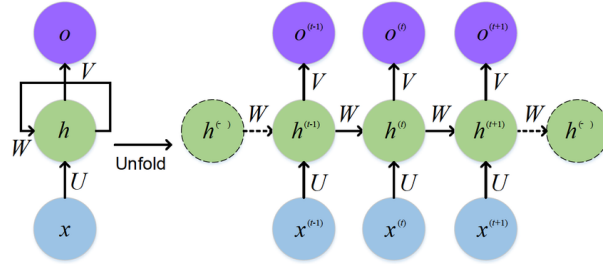


Figure 2.2: Recurrent neural unit(many to many)[7]

There is similar and better model then RNN and that is Gated Recurrent Unit called GRU. We use it in same way as RNN but it his own unique architecture[6]:

$$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}) \quad (2.97)$$

$$z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}) \quad (2.98)$$

$$n_t = \tanh(W_{in}x_t + b_{in} + r_t \odot (W_{hn}h_{(t-1)} + b_{hn})) \quad (2.99)$$

$$h_t = (1 - z_t) \odot n_t + z_t \odot h_{(t-1)} \quad (2.100)$$

and we can see it in plot ?? The GRU unit is specific because it has reset gate r_i and update gate u_i . Such architecture give better performance then at simple RNN.

2.2.3 Activation functions and Loss functions

In every neural architecture there is a activation function. Most used activation functions are:

- ReLU

$$\text{ReLU}(x) = \max(x, 0) \quad (2.101)$$

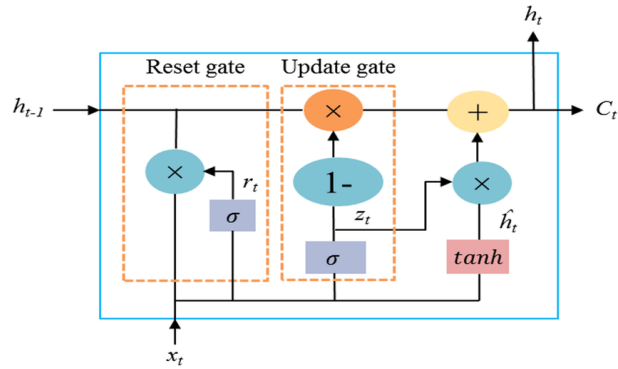


Figure 2.3: Gated recurrent unit[15]

This function passes only positive values.

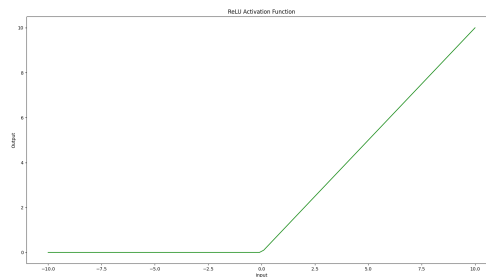


Figure 2.4: ReLU activation

- tanH - It gives an output $y \in [-1, 1]$

Output layer has no activation function. it is connected to the loss function. The most used loss functions are:

- Mean Absolute Error(MAE)

$$\text{MAE}(y, \hat{y}) = \frac{1}{N} \sum_i^N |y_i - \hat{y}_i| \quad (2.102)$$

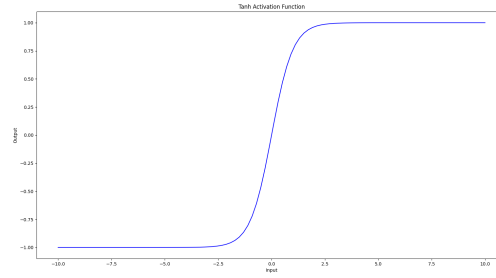


Figure 2.5: Tanh activation

- Mean Squared Error(MSE)

$$\text{MSE}(y, \hat{y}) = \frac{1}{N} \sum_i^N (y_i - \hat{y}_i)^2 \quad (2.103)$$

- HuberLoss

$$\text{HuberLoss}(y, \hat{y}; \delta) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{if } |(y - \hat{y})| < \delta \\ \delta((y - \hat{y}) - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \quad (2.104)$$

.

2.2.4 Optimizers

To do backpropagation we need an optimizer to update our learnable parameters. The easiest optimizer is stochastic gradient descent.

$$\mathbf{W}^{(i+1)} = \mathbf{W}^{(i)} - \mu \frac{\partial \text{Loss}(z)}{\partial \mathbf{W}^{(i)}} \quad (2.105)$$

There are more better one and for our use case we will use AdamW[11].

2.2.5 Backpropagation of Neural Models

After forward pass and calculated loss, we need to do just next steps Backward pass and optimization step.

In backward pass we are computing the gradients of our learnable parameters to use it for the optimization of the model.

For example we will use stochastic gradient descent(SGD)

$$\mathbf{W}_{i+1} = \mathbf{W}_i - \gamma \cdot \frac{\partial \text{Loss}(z)}{\partial \mathbf{W}_i} \quad (2.106)$$

$$\mathbf{b}_{i+1} = \mathbf{b}_i - \gamma \cdot \frac{\partial \text{Loss}(z)}{\partial \mathbf{b}_i} \quad (2.107)$$

where γ is a learning rate and one layered MLP without bias \mathbf{b} .

$$\mathbf{z}_1 = \mathbf{W}_0 \mathbf{y}_0 \quad (2.108)$$

$$\mathbf{y}_1 = \sigma(\mathbf{z}_1) \quad (2.109)$$

$$\text{Loss}(y_1, \hat{y}) = \text{MSE}(y_1, \hat{y}) \quad (2.110)$$

The obtaining the gradients of the weights is called backpropagation because for its calculation we need to apply chain rule for derivation.

$$\frac{\partial \text{Loss}(z)}{\partial \mathbf{W}_0} = \frac{\partial \text{Loss}(z)}{\partial \mathbf{y}_1} \cdot \frac{\partial \mathbf{y}_1}{\partial \mathbf{z}_1} \cdot \frac{\partial \mathbf{z}_1}{\partial \mathbf{W}_0} \quad (2.111)$$

Recurrent neural networks often fall as a victim of gradient vanishing or gradient exploding. In this manner the models doesn't learn because the derivations calculated through backpropagation are too small or too big. It is important that gradients slowly decrease through learning that whole model can generalize the data.

3 Methods

After previous chapters we should achieve basic understanding of neural network models and physics. With this knowledge we will introduce the methods which are capable to learn from dataset which are created through dynamics of the systems.

3.1 New paradigm - NeuralODE

Neural ODE is very interesting and important concept for solving the Ordinary Differential Equations and training the neural networks with trajectory data.

The default situation is described as:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t | \Theta) \quad (3.1)$$

Where $\mathbf{f}(\mathbf{x}, t | \Theta)$ represents the multilayer perception which creates vector field. This method has its history from residual networks where their feed-forward network looks like euler step

$$\mathbf{h}^{(i+1)} = \mathbf{h}^{(i)} + \mathbf{f}(\mathbf{h}^{(i)} | \Theta) \quad (3.2)$$

In the paper [4] it is introduced a adjoint method which solves problem with vanishing gradients.

This is achieved through continuous backpropagation which is stated in the paper. We can compare it

	residual network	adjoint method
a_t or $a(t)$:	$\frac{\partial L}{\partial z}$	$\frac{\partial L}{\partial z(t)}$
forward-pass:	$z_{i+1} = z_t + hf(z_t)$	$z(t+1) = z(t) + \int_t^{t+1} f(t)dt$
backward-pass:	$a_t = a_{t+1} + ha_{t+1} \frac{\partial f(z_t)}{\partial z_t}$	$a(t) = a(t+1) + \int_{t+1}^t a(t) \frac{\partial f(z(t))}{\partial z(z)} dt$
gradients:	$\frac{\partial L}{\partial \theta} = ha_{t+h} \frac{\partial f(z(t), \Theta)}{\partial \Theta}$	$\frac{\partial L}{\partial \theta} = \int_t^{t+1} a(t) \frac{f(z(t), \Theta)}{\partial \Theta}$

3.2 Recurrent time steppers

In chapter?? we introduced recurrent models which are RNN and GRU. Those models are very good at learning the time series data. To make it more physics informed we applied an euler method which it makes to time stepper models:

$$[\mathbf{v}_i, \mathbf{h}_{i+1}] = \text{RNN}(\mathbf{x}_i, \mathbf{h}_i) \text{ or } \text{GRU}(\mathbf{x}_i, \mathbf{h}_i) \quad (3.3)$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_i \cdot dt \quad (3.4)$$

We describe it this procedure as a rollout. This model has one big static parameter dependency which is fixed dt . We suggest to not to change this parameter after training. In Chapter experimentation we will show the performance of RNN and GRU time stepper models.

3.3 Hamiltonian Neural Network

This model comes from the paper [9] it is one of the physics informed models because it uses differentiation technique and applies the Hamiltonian equations. In this case the $\mathbf{x} = [\mathbf{q}, \mathbf{p}]$ are canonical or natural coordinates(Cartesian space). The forward pass looks like:

$$H_{\Theta} = \mathbf{f}(\mathbf{x}|\Theta) \quad (3.5)$$

$$\frac{d\mathbf{x}}{dt}(\mathbf{x}) = \mathbf{J} \left[\frac{\partial H_{\Theta}}{\partial \mathbf{p}}(\mathbf{x}), -\frac{\partial H_{\Theta}}{\partial \mathbf{q}}(\mathbf{x}) \right]^T \quad (3.6)$$

To make trajectory we use ODE solver

$$\mathbf{x} = \text{odeint} \left(\frac{d\mathbf{x}}{dt}, \mathbf{x}_0, t \right) \quad (3.7)$$

The odeint operator in this equation is a solver for ODEs like euler or RK4 method. In the experiments due the limitation of framework `pytorch` it is impossible to us ode solver based on adjoint method.

3.4 Graph Neural Network

Graph neural Networks are very new neural architecture which is available today. It is used broadly in social sciences and in chemistry like generating new chemical compounds. The main part of GNN is a graph \mathbf{G} . It is defined as a collection of vertices and edges $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$ where $\mathbf{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_i, \dots, \mathbf{v}_n\}$ has n vertices and $\mathbf{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_i, \dots, \mathbf{e}_m\}$ has m edges.

The edge is connection between to nodes it can be directed(only one direction between the nodes) or undirected(both direction between the nodes). This is very important because the graphs has many properties which we can use building it as Graph Neural Model. This connectivity can be represented with adjacency matrix \mathbf{A} . Adjacency matrix is denoted as $\mathbf{A} \in \{0, 1\}^{n \times n}$

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if } (\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \quad (3.8)$$

Most important property of Graphs is creating the normalized Laplacian matrix. This is foundation of the idea about graph neural networks and it is defined as:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{\frac{1}{2}} \quad (3.9)$$

where \mathbf{D} is degree matrix. Degree of a node is the number of nodes that are adjacent to the node in question. This matrix has only diagonal elements.

We recognize 3 main training types of GNNs and that are graph- , edge and node - oriented learning.

In our work we will use node oriented type.

3.4.1 Convolutional Graph Neural Network

First idea about graph neural network was Spectral Graph Network (SGN) [] in signal proccesing tasks which is based on spectrality of the modified Laplacian in form

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{\frac{1}{2}} \quad (3.10)$$

where we exchange adjecency matrix with learnable weight matrix. Because of cost of computation and hard backpropagation, the ChebNet was introduced. It was dependent of fourier transformation and had high computational comlpexity. First Kipf and Welling used both ideas and simplified Laplacian to create broadly used architecture called Convolutional

Graph Network.

Because of instability of Laplacian Kipf and Welling [1] renormalization trick which stabilize the network:

$$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I} \quad (3.11)$$

$$\tilde{\mathbf{A}} = \sum_i \tilde{\mathbf{A}}_{ii} \quad (3.12)$$

$$\mathbf{L} \rightarrow \tilde{\mathbf{D}}^{\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{\frac{1}{2}} \quad (3.13)$$

This renormalised Laplacian is a base of Convolutional Graph Neural Network that we know today. Forward pass for convolutional graph Network:

$$\mathbf{h}^{(i+1)} = \sigma(\tilde{\mathbf{D}}^{\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{\frac{1}{2}} \mathbf{W} \mathbf{h}^{(i)}) \quad (3.14)$$

This mathematical expression is equivalent to message passing and aggregation from and between the neighbouring nodes. In this case with \sum operator.

With knowledge that the neural network doesn't depend on graph structure but on message passing and aggregation between the nodes, we can build more designs of graph neural networks. We should not forget, that those message passing and aggregation can be parallelized and calculated on GPUs improving performance. In our use cases we will use Framework DGL.

3.4.2 Gated Attention Network

In paper [1] is introduced new type of architecture which incorporates weighting factors. This weighting factors α shows importance of the node j to the node in question. The forward pass of one layer is defined as:

$$\mathbf{z}_i^{(l)} = \mathbf{W}_f^{(l)} \mathbf{h}_i^{(l)} \quad (3.15)$$

$$e_{ij}^{(l)} = \text{LeakyReLU}(\mathbf{W}_a^{(l)T} (\mathbf{z}_i^{(l)} \parallel \mathbf{z}_j^{(l)})) \quad (3.16)$$

$$\alpha_{ij}^{(l)} = \frac{\exp(e_{ij}^{(l)})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik}^{(l)})} \quad (3.17)$$

$$\mathbf{h}_i^{(l+1)} = \sigma \left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)} \mathbf{z}_j^{(l)} \right) \quad (3.18)$$

This architecture is computationally efficient and storage efficient. It is most important fixed which means it doesn't depend on number of nodes in data. For better learning the datasets it is suggested to use dropout at attention.

3.5 Graph Neural Hamiltonian Network

This architecture is inspired on implementation of HNN and paper [] which does similar models to it. This model is physics informed model. It is created exactly as HNN but the base model in this case isn't MLP but the GNN.

$$H_{\Theta} = \mathbf{f}(\mathbf{x}|\Theta) = \text{GNN}(\mathbf{x}) \quad (3.19)$$

$$\frac{d\mathbf{x}}{dt}(\mathbf{x}) = \mathbf{J} \left[\frac{\partial H_{\Theta}}{\partial \mathbf{p}}(\mathbf{x}), -\frac{\partial H_{\Theta}}{\partial \mathbf{q}}(\mathbf{x}) \right]^T \quad (3.20)$$

This model gives trajectory through using of one ode solvers.

3.6 Gated Recurrent Graph Hamiltonian Neural Network - GRUGHNN

In previous section we introduced the GHNN, which is based on a hamiltonian layer and an ode solver. This architecture is not different but improved with a GRU unit at the beginning of the input. This model is equivalent to the GRU stepper but we added a GHNN in it.

$$[\mathbf{a}_i, \mathbf{h}_{i+1}] = \text{GRU}(\mathbf{x}_i, \mathbf{h}_i) \quad (3.21)$$

$$H_i = \text{GNN}(\mathbf{a}_i) \quad (3.22)$$

$$\mathbf{v}_i = \mathbf{J} \left[\frac{\partial H_i}{\partial \mathbf{p}}(\mathbf{x}), -\frac{\partial H_i}{\partial \mathbf{q}}(\mathbf{x}) \right]^T \quad (3.23)$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_i \cdot dt \quad (3.24)$$

4 Datasets

In this Chapter we will discuss some Physics cases which totally obey conservative property of total energy(Hamiltonian). In previous chapter we discussed about oscillator which is trivial model and how we found a solution of the dynamics of the model. In this chapter we will not only show how oscillator dataset should be created but we will discuss about 4 models which are:

- N body problem
- two body problem
- threebody problem
- N pendulum

4.1 Oscillator

As we already shown the oscillator solution we could show solution of the oscillator through hamiltonian equations. For this we will use Kinetic Energy and Potential energy $U = \frac{1}{2}kx^2$ Kinetic energy is defined as $T = \frac{1}{2}mv^2$ or if we use momentum $p = mv$ we get $T = \frac{1}{2m}p^2$. Our Hamiltonian will look like:

$$\mathcal{H} = \frac{1}{2m}p^2 + \frac{1}{2}kx^2 \quad (4.1)$$

This case is trivial. We could get \dot{x} and \dot{p} through hamiltonian equations $\dot{q} = \frac{\partial \mathcal{H}}{\partial p}$ and $\dot{p} = -\frac{\partial \mathcal{H}}{\partial q}$ Or we could compare Hamiltonian with ellipse formula

$$1 = \frac{x^2}{a^2} + \frac{y^2}{b^2} \quad (4.2)$$

and set the components in parametric equation

$$(x, y) = (a \cos(\phi), b \sin(\phi)) \rightarrow (p, q) = \left(\sqrt{2Hm} \cos(\phi), \sqrt{\frac{2H}{k}} \sin(\phi) \right) \quad (4.3)$$

of the ellipse. This equation represents kinematic of the oscillator. To find periodicity let say that $\phi = \omega t + \Phi$ and we know that the $p = mx$, with this equation we can calculate ω .

$$p = m\dot{x} = \sqrt{2Hm} \cos(\omega t + \Phi) \quad (4.4)$$

$$\dot{x} = \sqrt{\frac{2H}{m}} \cos(\omega t + \Phi) \quad (4.5)$$

$$x = \int \sqrt{\frac{2H}{m}} \cos(\omega t + \Phi) dt = \frac{1}{\omega} \sqrt{\frac{2H}{m}} \sin(\omega t + \Phi) = \sqrt{\frac{2H}{k}} \sin(\omega t + \Phi) \quad (4.6)$$

$$\omega = \sqrt{\frac{k}{m}} \quad (4.7)$$

The oscillator has harmonic, periodic movement which means that after some time T the movement will be repeated. With that property we can calculate the period over circle circumference:

$$T = \frac{2\pi}{\omega} \quad (4.8)$$

With this we can create a dataset within the program code. For coding we used framework pytorch to calculate this dataset numerically.

For the obtain the solution of ODE we used package torchdiffeq to solve numerical equation with dopri5 method which is most accurate method for solving ODEs.

To create unique trajectories for the dataset we just need to specify energy region.

4.2 N-body problem

N-body problem is most complex problem in physics. It describes a movement of bodies in the free space. There are no conditions and there are multitude of solutions, because this problem if it isn't deterministic it goes under the quantum theory. In Newtonian formulation the N-body problem is

$$m_i \ddot{\mathbf{x}}_i = - \sum_{i \neq j, j=1}^n G \frac{m_i m_j (\mathbf{x}_i - \mathbf{x}_j)}{||\mathbf{x}_i - \mathbf{x}_j||^3} \quad (4.9)$$

and its Hamiltonian:

$$\mathcal{H} = \sum_i^n \frac{1}{2m_i} \|\mathbf{p}_i\|^2 - \sum_{1 \leq i, j \leq n, i \neq j} \frac{Gm_i m_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} \quad (4.10)$$

for $i \in [1, n]$. In our use case to create dataset we needed to make some conditions that our trajectories have if possible constant hamiltonian.

The biggest issue in this dataset was the crashes between the bodies. This can be resolved with simple trick adding the ϵ to fix maximal potential energy

$$- \sum_{1 \leq i, j \leq n, i \neq j} \frac{Gm_i m_j}{\|\mathbf{x}_i - \mathbf{x}_j\| + \epsilon}. \quad (4.11)$$

We secured that the value in dominator is never 0. This is important because we use numerical solvers to create trajectories. If we have energy which achieves very big difference in values in one time step, our solution will be inaccurate. For calculation of such trajectories we are forced not to make Hamiltonian equations, but to calculate the trajectories using acceleration, velocity and position of the bodies trough some symplectic numerical methods.

4.2.1 Symplectic numerical methods to solve N-body problem

Symplectic numerical methods are designed for numerical solution of Hamilton Equations. It posses conserved quality to approximate the Total Energy the Hamiltonian. They have two Forms. One in canonical coordinates(q,p) and in langrangian coordinates(x,v,a). Most used ones are in langrangian form and we will use it to create trajectory for N-body problem.

For the N-body problem we used two numerical solvers. In first step we used velocity verlet

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{v}_i + \mathbf{a}_i \frac{h^2}{2} \quad (4.12)$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + (\mathbf{a}_i + \mathbf{a}_{i+1}) \frac{h}{2} \quad (4.13)$$

and for the next steps with timestep h we used more stable beeman method

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h\mathbf{v}_i + \frac{h^2}{6}(4\mathbf{a}_i - \mathbf{a}_{i-1}) \quad (4.14)$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \frac{h}{6}(2\mathbf{a}_{i+1} + 5\mathbf{a}_i - \mathbf{a}_{i-1}). \quad (4.15)$$

This needed to be done because the beeman method needs acceleration of the next and previous step to be calculated.

4.2.2 twobody problem

Two body problem is one of the most easiest dataset which could be created from N-body problem. It has one trivial solution which is Binary star movement. Two stars trough their movemnt maintain constant distance r and their center of mass \mathbf{p}_m is static at the origin. The newtonian form of twobody Problem is written as

$$m_1 \ddot{\mathbf{q}}_1 = -G \frac{m_1 m_2 (\mathbf{q}_1 - \mathbf{q}_2)}{\|\mathbf{q}_1 - \mathbf{q}_2\|^3} \quad (4.16)$$

$$m_2 \ddot{\mathbf{q}}_2 = -G \frac{m_1 m_2 (\mathbf{q}_2 - \mathbf{q}_1)}{\|\mathbf{q}_2 - \mathbf{q}_1\|^3} \quad (4.17)$$

From those equations we can get a statement about third newton law:

$$m_1 \ddot{\mathbf{q}}_1 - m_2 \ddot{\mathbf{q}}_2 = 0 = \mathbf{F}_{ij} - \mathbf{F}_{ji} \quad (4.18)$$

and its hamiltonian

$$\mathcal{H} = \frac{\|\mathbf{p}_1\|^2}{2m_1} + \frac{\|\mathbf{p}_2\|^2}{2m_2} - G \frac{m_1 m_2}{\|\mathbf{q}_1 - \mathbf{q}_2\|} \quad (4.19)$$

We said that we have static center mass point. center mass point is obtained using following formula

$$\mathbf{q}_m = \frac{1}{m_1 + m_2} (m_1 \mathbf{q}_1 + m_2 \mathbf{q}_2) \quad (4.20)$$

if we set the center mass point in origin the equation simplifies nd we get one about velocity

$$m_1 \mathbf{q}_1 + m_2 \mathbf{q}_2 = 0 \quad (4.21)$$

$$m_1 \dot{\mathbf{q}}_1 + m_2 \dot{\mathbf{q}}_2 = \mathbf{p}_1 + \mathbf{p}_2 = 0 \quad (4.22)$$

. From now on we will write $\mathbf{p} = \mathbf{p}_1 = -\mathbf{p}_2$. Lets make some formulas about distance.

$$\mathbf{q}_2 = \frac{m_1}{m_2} \mathbf{q}_1 \quad (4.23)$$

$$\|\mathbf{q}_2\| =_{\mathbf{q}_m=0} \frac{m_1}{m_2} \|\mathbf{q}_1\| \quad (4.24)$$

$$r_2 = \frac{m_1}{m_2} r_1 \quad (4.25)$$

With this we got interesting property

$$\frac{r_1}{r_2} = \frac{m_2}{m_1} \quad (4.26)$$

. This can be used to calculate constant distance between the bodies

$$r = r_1 \left(1 + \frac{m_1}{m_2} \right) \quad (4.27)$$

With those equations we can simplify hamiltonian

$$\mathcal{H} = \|\mathbf{p}\|^2 \left(\frac{1}{m_1} + \frac{1}{m_2} \right) - G \frac{m_1 m_2}{r} \quad (4.28)$$

. Still to make a dataset with twobody trajectories we need a period T . Let assume that our solution as describe is radial. the bodies moves on a circular path

$$\mathbf{q}_1 = r_1 \begin{bmatrix} \cos(\Phi) \\ \sin(\Phi) \end{bmatrix} \quad (4.29)$$

$$\mathbf{q}_2 = r_2 \begin{bmatrix} \cos(\Phi + \pi) \\ \sin(\Phi + \pi) \end{bmatrix} \quad (4.30)$$

The momentum is tangential in dependency of orientation of position

$$\mathbf{p}_1 = \|\mathbf{p}\| \begin{bmatrix} \sin(\Phi) \\ -\cos(\Phi) \end{bmatrix} \quad (4.31)$$

$$\mathbf{p}_2 = \|\mathbf{p}\| \begin{bmatrix} \sin(\Phi + \pi) \\ -\cos(\Phi + \pi) \end{bmatrix} \quad (4.32)$$

with $\Phi \in [0, 2\pi]$. Let say that $\Phi = \omega t$ and we want to find a period of movement. With the equations

$$\dot{\mathbf{q}}_1 = r_1 \omega \begin{bmatrix} \sin(\omega t) \\ -\cos(\omega t) \end{bmatrix} \quad (4.33)$$

$$= \frac{\mathbf{p}_1}{m_1} = \frac{\|\mathbf{p}\|}{m_1} \begin{bmatrix} \sin(\omega t) \\ -\cos(\omega t) \end{bmatrix} \quad (4.34)$$

$$\|\mathbf{p}\| = r_1 m_1 \omega \quad (4.35)$$

, we calculated the value of the momentum. This momentum we can get from Physics and newton third law. If some body moves around some point in radial path, we can calculate a centripetal Force and this Force should be equal to Gravitational Force

$$F_c = \frac{\|\mathbf{p}\|^2}{m_1 r_1} = \frac{G m_1 m_2}{r^2} = F_g \quad (4.36)$$

Substituting $\|\mathbf{p}\|$ we get ω .

$$\omega = \frac{1}{r} \sqrt{G \frac{m_2}{r_1}} \quad (4.37)$$

In two body problem to create some diverse dataset but with equivalent bodies we vary the distance between the bodies. We need to be careful because with grater distance r the period T will be longer.

The mathematical formulation of the solution which is used for code is taken from hamiltonian equations

$$\dot{\mathbf{x}} = J \frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}) = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{1}{m_1} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{m_1} \\ 0 & 0 & 0 & 0 & -\frac{1}{m_2} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{m_2} \\ -\mu_{1,2} & 0 & \mu_{1,2} & 0 & 0 & 0 \\ 0 & -\mu_{1,2} & 0 & \mu_{1,2} & 0 & 0 \end{bmatrix} \begin{bmatrix} q_{1x} \\ q_{1y} \\ q_{2x} \\ q_{2y} \\ p_x \\ p_y \end{bmatrix} \quad (4.38)$$

where $\mu_{ij} = \frac{G m_i m_j}{r^3}$

4.2.3 Three body problem

Three body problem is from formulation similar to two body problem, but finding the periodic solutions is very complex task. Fortunately the M. Šuvlakov and V. Dmitrašinović have found many of the initial values to create periodic three body trajectory and made data public over the website for everyone to use.[20][17]. But under some definitions such like the gravitational constant G and all masses m are 1. Even today there are scientists which are documenting initial values for more periodic solutions [10]. Having those conditions we need just Hamiltonian equations which are easy to obtain.

Obtaining the formula for straight-forward calculation the trajectories of the threebody problem we need to define a function

$$\Xi(\mathbf{q}_i, \mathbf{q}_j) = \Xi_{i,j} = \frac{G m_i m_j}{\|(\mathbf{q}_i - \mathbf{q}_j)\|^3}. \quad (4.39)$$

In following consider $\mathbf{m}_i^{-1} = \text{diag}(\frac{1}{m_i}, \frac{1}{m_i})$ and coordinates are two dimensional.

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & \mathbf{m}_1^{-1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{m}_2^{-1} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{m}_3^{-1} \\ -(\Xi_{1,2} + \Xi_{1,3}) & \Xi_{1,2} & \Xi_{1,3} & 0 & 0 & 0 \\ \Xi_{1,2} & -(\Xi_{1,2} + \Xi_{2,3}) & \Xi_{2,3} & 0 & 0 & 0 \\ \Xi_{1,3} & \Xi_{2,3} & -(\Xi_{1,3} + \Xi_{2,3}) & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} \quad (4.40)$$

The three-body problem have three interesting properties due its defined potential energy. Their potential energy is defined over gravitational force between the bodies. In another words we can translate the trajectory in space and rotate it. Let us prove this over Hamiltonian Energy

$$\mathcal{H} = \frac{\|\mathbf{p}_1\|^2}{2m_1} + \frac{\|\mathbf{p}_2\|^2}{2m_2} + \frac{\|\mathbf{p}_3\|^2}{2m_3} - G \frac{m_1 m_2}{\|\mathbf{q}_1 - \mathbf{q}_2\|} - G \frac{m_2 m_3}{\|\mathbf{q}_2 - \mathbf{q}_3\|} - G \frac{m_1 m_3}{\|\mathbf{q}_1 - \mathbf{q}_3\|} \quad (4.41)$$

- Translation:

Lets define $\mathbf{q}_i = \lim_{\mathbf{q}_t \rightarrow \mathbf{0}}(\mathbf{q}_i + \mathbf{q}_t)$ where \mathbf{q}_t is translation vector. When we substitute \mathbf{q}_i . The hamiltoinian acts as follows

$$\begin{aligned} \mathcal{H} &= \frac{\|\mathbf{p}_1\|^2}{2m_1} + \frac{\|\mathbf{p}_2\|^2}{2m_2} + \frac{\|\mathbf{p}_3\|^2}{2m_3} \\ &\quad - \lim_{\mathbf{q}_t \rightarrow \mathbf{0}} G \frac{m_1 m_2}{\|\mathbf{q}_1 + \mathbf{q}_t - (\mathbf{q}_2 + \mathbf{q}_t)\|} - \lim_{\mathbf{q}_t \rightarrow \mathbf{0}} G \frac{m_2 m_3}{\|\mathbf{q}_2 + \mathbf{q}_t - (\mathbf{q}_3 + \mathbf{q}_t)\|} \\ &\quad - \lim_{\mathbf{q}_t \rightarrow \mathbf{0}} G \frac{m_1 m_3}{\|\mathbf{q}_1 + \mathbf{q}_t - (\mathbf{q}_3 + \mathbf{q}_t)\|} \\ &= \frac{\|\mathbf{p}_1\|}{2m_1} + \frac{\|\mathbf{p}_2\|}{2m_2} + \frac{\|\mathbf{p}_3\|}{2m_3} \\ &\quad - G \frac{m_1 m_2}{\|\mathbf{q}_1 - \mathbf{q}_2 + (\mathbf{q}_t - \mathbf{q}_t)\|} - G \frac{m_2 m_3}{\|\mathbf{q}_2 - \mathbf{q}_3 + (\mathbf{q}_t - \mathbf{q}_t)\|} - G \frac{m_1 m_3}{\|\mathbf{q}_1 - \mathbf{q}_3 + (\mathbf{q}_t - \mathbf{q}_t)\|} \\ &= \frac{\|\mathbf{p}_1\|}{2m_1} + \frac{\|\mathbf{p}_2\|}{2m_2} + \frac{\|\mathbf{p}_3\|}{2m_3} - G \frac{m_1 m_2}{\|\mathbf{q}_1 - \mathbf{q}_2\|} - G \frac{m_2 m_3}{\|\mathbf{q}_2 - \mathbf{q}_3\|} - G \frac{m_1 m_3}{\|\mathbf{q}_1 - \mathbf{q}_3\|} \end{aligned}$$

- Rotation: Lets define $\mathbf{q}_i = r_i \begin{bmatrix} \cos(\beta + \alpha) \\ \sin(\beta + \alpha) \end{bmatrix}$ and $\mathbf{p}_i = p_i \begin{bmatrix} -\sin(\beta + \alpha) \\ \cos(\beta + \alpha) \end{bmatrix}$ where $\beta \in (0, 2\pi]$ but it is defined and $\alpha \in (0, 2\pi]$ which is free parameter.

Let substitute this in Hamiltonian

$$\begin{aligned}\mathcal{H} = & \frac{\left\| p_1 \begin{bmatrix} -\sin(\beta + \alpha) \\ \cos(\beta + \alpha) \end{bmatrix} \right\|^2}{2m_1} + \frac{\left\| p_2 \begin{bmatrix} -\sin(\beta + \alpha) \\ \cos(\beta + \alpha) \end{bmatrix} \right\|^2}{2m_2} + \frac{\left\| p_3 \begin{bmatrix} -\sin(\beta + \alpha) \\ \cos(\beta + \alpha) \end{bmatrix} \right\|^2}{2m_3} \\ & - G \frac{m_1 m_2}{\left\| r_1 \begin{bmatrix} \cos(\beta + \alpha) \\ \sin(\beta + \alpha) \end{bmatrix} - r_2 \begin{bmatrix} \cos(\beta + \alpha) \\ \sin(\beta + \alpha) \end{bmatrix} \right\|} \\ & - G \frac{m_2 m_3}{\left\| r_2 \begin{bmatrix} \cos(\beta + \alpha) \\ \sin(\beta + \alpha) \end{bmatrix} - r_3 \begin{bmatrix} \cos(\beta + \alpha) \\ \sin(\beta + \alpha) \end{bmatrix} \right\|} \\ & - G \frac{m_1 m_3}{\left\| r_1 \begin{bmatrix} \cos(\beta + \alpha) \\ \sin(\beta + \alpha) \end{bmatrix} - r_3 \begin{bmatrix} \cos(\beta + \alpha) \\ \sin(\beta + \alpha) \end{bmatrix} \right\|}\end{aligned}$$

From trigonometry we know that $\cos(\phi)^2 + \sin(\phi)^2 = 1$ which means $\left\| \begin{bmatrix} \cos(\beta + \alpha) \\ \sin(\beta + \alpha) \end{bmatrix} \right\| = \left\| \begin{bmatrix} \cos(\beta) \\ \sin(\beta) \end{bmatrix} \right\| = 1$.

Applying this theorem we get

$$\mathcal{H} = \frac{p_1^2}{2m_1} + \frac{p_2^2}{2m_2} + \frac{p_3^2}{2m_3} - G \frac{m_1 m_2}{|r_1 - r_2|} - G \frac{m_2 m_3}{|r_2 - r_3|} - G \frac{m_1 m_3}{|r_1 - r_3|} \quad (4.42)$$

which is equivalent to the previous definition.

For the dataset creation we wanted that our trajectory mass middle point is fixed. We needed only rotation around it. For such rotation we used formula for such transformation

$$\hat{\mathbf{q}} = \begin{bmatrix} 1 & 0 & 0 & x_m \\ 0 & 1 & 0 & y_m \\ 0 & 0 & 1 & z_m \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_m \\ 0 & 1 & 0 & -y_m \\ 0 & 0 & 1 & -z_m \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_q \\ y_q \\ z_q \\ 1 \end{bmatrix} \quad (4.43)$$

which for two dimensional cases can be rewritten as:

$$\hat{\mathbf{q}} = \mathbf{R}\mathbf{q} - \mathbf{R}\mathbf{q}_m + \mathbf{q}_m \quad (4.44)$$

where $\mathbf{R} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$.

4.3 N Pendulum

The N Pendulum is straight forward model. To make it in canonical coordinates which are angles of the joints, still we need to define Kinematic of the model.

$$x_i = \sum_i^n l_i \sin(\Theta_i) \quad (4.45)$$

$$y_i = - \sum_i^n l_i \cos(\Theta_i) \quad (4.46)$$

Their temporal derivations are:

$$\dot{x}_i = \sum_i^n \dot{\Theta}_i l_i \cos(\Theta_i) \quad (4.47)$$

$$\dot{y}_i = \sum_i^n \dot{\Theta}_i l_i \sin(\Theta_i) \quad (4.48)$$

From this kinematic model we just calculated the Hamiltonian and proceed with autodifferentiation from pytorch framework to calculate the values of hamiltonian equations

$$\mathcal{H} = \frac{1}{2} \sum_i^n m_i (\dot{x}_i^2 + \dot{y}_i^2) + \sum_i^n m_i g_i y_i \quad (4.49)$$


. We set a canonical coordinates (Θ, p_Θ) and we can calculate $p_\Theta = \frac{\partial \mathcal{H}}{\partial \dot{\Theta}}$. Making the Hamiltonian only dependent on Θ and p_Θ we can get hamiltonian equations:

$$\dot{\Theta}_i = \frac{\partial \mathcal{H}}{\partial p_{\Theta_i}} \quad (4.50)$$

$$p_{\dot{\Theta}_i} = - \frac{\partial \mathcal{H}}{\partial \Theta} \quad (4.51)$$

At the Dataset creation we made randomised dataset with $\Theta \in [-\frac{\pi}{4}, \frac{\pi}{4}]$ and starting $p_{\Theta_i} = 0$.

We can create a dataset with observation. With interest to creating of urdf files, we create the framework which through python coding creates accurate urdf model. This was used to create pendulum with n number of joints. This model was observed in pybullet. From the use case we suggest if you are working with pendulums where $N < 10$ please use



the pytorch numerical technique, otherwise use pybullet observation. This is due the numerical stability. More joints or degrees of freedom, hamiltonian even if it should be conservative can't hold stability. In that case using pybullet and decide if Hamiltonian of that trajectory conservative enough trough calculation the mean and standard deviation of the energy sample.

Bibliography

- [1] M. Calvo, J.I. Montijano, and L. Randez. “A fifth-order interpolant for the Dormand and Prince Runge-Kutta method”. In: *Journal of Computational and Applied Mathematics* 29.1 (1990), pp. 91–100. ISSN: 0377-0427. DOI: [https://doi.org/10.1016/0377-0427\(90\)90198-9](https://doi.org/10.1016/0377-0427(90)90198-9). URL: <https://www.sciencedirect.com/science/article/pii/0377042790901989>.
- [2] Kit Yan Chan et al. “Deep neural networks in the cloud: Review, applications, challenges and research directions”. In: *Neurocomputing* 545 (2023), p. 126327. ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2023.126327>. URL: <https://www.sciencedirect.com/science/article/pii/S0925231223004502>.
- [3] Suresh Chandra, Mohit Kumar Sharma, and Monika Sharma. “Harmonic Oscillator”. In: *Fundamentals of Mechanics*. Foundation Books, 2014, pp. 150–174.
- [4] Ricky T. Q. Chen et al. *Neural Ordinary Differential Equations*. 2019. arXiv: 1806.07366 [cs.LG]. URL: <https://arxiv.org/abs/1806.07366>.
- [5] Susmita Das et al. “Recurrent Neural Networks (RNNs): Architectures, Training Tricks, and Introduction to Influential Research”. In: *Machine Learning for Brain Disorders*. Ed. by Olivier Colliot. New York, NY: Springer US, 2023, pp. 117–138. ISBN: 978-1-0716-3195-9. DOI: 10.1007/978-1-0716-3195-9_4. URL: https://doi.org/10.1007/978-1-0716-3195-9_4.
- [6] Rahul Dey and Fathi M. Salem. “Gate-variants of Gated Recurrent Unit (GRU) neural networks”. In: *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)* (2017), pp. 1597–1600. URL: <https://api.semanticscholar.org/CorpusID:8492900>.
- [7] Weijiang Feng et al. “Audio visual speech recognition with multimodal recurrent neural networks”. In: May 2017, pp. 681–688. DOI: 10.1109/IJCNN.2017.7965918.

-
- [8] Mattias Flygare. “Holonomic versus nonholonomic constraints”. In: 2012. URL: <https://api.semanticscholar.org/CorpusID:118677177>.
- [9] Sam Greydanus, Misko Dzamba, and Jason Yosinski. *Hamiltonian Neural Networks*. 2019. arXiv: 1906.01563 [cs.NE]. URL: <https://arxiv.org/abs/1906.01563>.
- [10] Ana Hudomal. “New periodic solutions to the three-body problem and gravitational waves”. In: *Master of Science Thesis at the Faculty of Physics, Belgrade University* (2015).
- [11] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG]. URL: <https://arxiv.org/abs/1711.05101>.
- [12] Michael Lutter, Christian Ritter, and Jan Peters. *Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning*. 2019. arXiv: 1907.04490 [cs.LG]. URL: <https://arxiv.org/abs/1907.04490>.
- [13] Yao Ma and Jiliang Tang. “Foundations of Deep Learning”. In: *Deep Learning on Graphs*. Cambridge University Press, 2021, pp. 43–72.
- [14] Joseph D. Romano Matthew J. Benacquista. *Classical Mechanics*. 2018. ISBN: 978-3-319-68779-7.
- [15] Saeed Mohsen. “Recognition of human activity using GRU deep learning algorithm”. In: *Multimedia Tools and Applications* 82 (May 2023). DOI: 10.1007/s11042-023-15571-y.
- [16] B. W. Montague. “Basic Hamiltonian mechanics”. In: *CERN Accelerator School: Course on Advanced Accelerator Physics (CAS)*. 1993, pp. 1–14.
- [17] Institute of Physics Belgrade. *THREE-BODY GALLERY*. URL: <http://three-body.ipb.ac.rs/>.
- [18] Subhankar Ray and J. Shamanna. *Virtual Displacement in Lagrangian Dynamics*. 2004. arXiv: physics/0410123 [physics.ed-ph]. URL: <https://arxiv.org/abs/physics/0410123>.
- [19] Robin M. Schmidt. *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview*. 2019. arXiv: 1912.05911 [cs.LG]. URL: <https://arxiv.org/abs/1912.05911>.
- [20] Milovan Šuvakov and V. Dmitrašinović. “Three Classes of Newtonian Three-Body Planar Periodic Orbits”. In: *Physical Review Letters* 110.11 (Mar. 2013). ISSN: 1079-7114. DOI: 10.1103/physrevlett.110.114301. URL: <http://dx.doi.org/10.1103/PhysRevLett.110.114301>.