Numerical solutions of some models of diabetes by using physics informed deep learning methods

By Baptiste Guilbery
Advisor: Aziz Belmiloudi

October 4, 2024

Contents

Abstract: In this paper, we will study two mathematical models linked to diabetes. Diabetes is chronic disease increasing the blood glucose concentration. This disease can lead to other dangerous diseases (cardiovascular problem...). The goal of this study is to analyse these models behaviours for the simulation of the complex dynamics of diabetes. We also modify these models, and observe the modifications of their behaviours. This part could lead to improvements in the diabetes models and a better understanding of the models. To make these simulations, we will use Physics Informed Neural Network (PINN) to approximate solutions for several systems of differential equations. This method allows to use the knowledge about the dynamic of the system of equation (governing equation, conservation of energy,...) with a neural network to solve a system of differential equation. In the first part of this paper, we will define a global framework for representing a system of differential equation and how the classical PINN's algorithm works, we will explain two modifications of this classical algorithm that will be used in this paper. In the second, part we will use some simple systems of equations to validate our approach. And in the third part we will use our approach to analyse the two models. We will also modify them with the addition of delayed term or the use of Caputo fractional derivative in order to modify the behaviour of the solutions (apparition of oscillations, change of the limits...). We will find that these modifications only change the limits of the system of differential equation.

1 Introduction

Diabetes is a chronic disease, which represents a major public health challenge worldwide. Diabetes is characterized by persistent elevation of blood glucose levels, it can lead to serious complications, including cardiovascular problems, kidney damage, visual impairment and peripheral circulation problems. There are two main forms of diabetes: type 1 diabetes, resulting from insufficient insulin production, and type 2 diabetes, associated with insulin resistance linked to lifestyle factors such as diet and physical activity. In the world, more than five hundred millions of people are affected by diabetes and lead to two million death per year.

Diabetes poses many challenges. In terms of individual health, the management of the disease requires constant vigilance, including regular blood glucose monitoring, dietary adjustments and insulin administration. Also, the costs of healthcare, treatment and complications associated with diabetes are considerable. Understanding diabetes, is essential for developing effective prevention and treatment strategies.

In this paper we will consider two models used to understand diabetes. The first one is the Diabetes Complication (DC) model introduced by AlShurbaji M. et al. in [AlS23] is used to modeling the evolution of the population with diabetes. The people are divided into two groups one without diabetic complications (group D) and the other with diabetic complication (group C). We will also study the Bergman control model, which models, the evolution of the glucose and insulin concentration in the blood of a person with a type 1 diabetes.

These two models are modeled by a system of differential equations. To solve these systems, we will use a Deep-Learning algorithm called Physics Informed Neural Network (PINN). This algorithm use a Neural Network and informations about the system (governing equation, energy conservation,...) to compute the solution. This method has the main advantage of being able to take a wide variety of differential equation (with fractional derivatives, non-linear,...) and use the exact same algorithm to find a solution. The PINN could be a useful tool for a non-specialist to easily solve systems of differential equations.

A goal of our study consist is to modify these systems of differential equations by adding delay term in the equation or by using Caputo fractional derivatives. We expect to modify the behaviour of the solutions (modify the limits, create a periodic pattern,...).

In the first part we will define the mathematical framework and show how the PINN will compute the solutions. After that we will introduce two modifications to the PINN and evaluate these modifications performances with simple tests. Finally, we will study the two models linked to diabetes and modify these models with delayed terms or Caputo fractional derivatives and observe their new behaviours.

2 Methodology

In this first part we will describe the mathematical framework, the definition of a system of differential equations, the classical definition of the PINN and its training. We will after introducing a new algorithm called mPINN which came from two modifications of the classical algorithm.

2.1 Formulation of the problem

In this work, we will consider the following general form of a fractional order system of delay differential equations (for $\alpha \in (0;1]$ and $\tau \geq 0$)

$$\begin{aligned} &(D_t^\alpha y)(t) = f(t, y(t), y(t-\tau)), \text{ for } t > 0, \\ &y(t) = g(t), \text{ for } t \in [-\tau; 0], \end{aligned} \tag{1}$$

where

$$(D_t^\alpha y)(t) = \begin{cases} y'(t) & \text{if } \alpha = 1, \\ ({}_0^C D_t^\alpha y)(t) = \frac{1}{\Gamma(1-\alpha)} \int_0^t (t-s)^{-\alpha} y'(s) ds & \text{if } \alpha < 1 \end{cases}$$

and

$$\begin{aligned} y \colon \mathbb{R} &\to \mathbb{R}^p \\ t &\mapsto y(t). \end{aligned}$$

Now let's introduce the following "dynamic function"

$$\mathcal{F}(t, y(t), y(t-\tau), (D_t^\alpha y)(t)) = (D_t^\alpha y)(t) - f(t, y(t), y(t-\tau)).$$

Then, the system (SDE) becomes

$$\begin{aligned} &\mathcal{F}(t, y(t), y(t-\tau), (D_t^\alpha y)(t)) = 0, \text{ for } t > 0, \\ &y(t) = g(t), \text{ for } t \in [-\tau; 0]. \end{aligned} \tag{SDE}$$

2.2 Principle of the classical algorithm

To solve this differential system of equation, we can use PINN as proposed by Baty H. et al. in [Bat23]. The PINN is a function depending of input data and parameters. In this section, we present the classical definition of the PINN function and the algorithm used to choose his parameters in order to find a solution to the system of equation (SDE):

We begin by considering the following function:

$$\begin{aligned} \mathcal{N}^{n,m} \colon \mathbb{R}^n \times \mathbb{R}^{m\times n} \times \mathbb{R}^m &\to \mathbb{R}^m \\ (z, V, c) &\mapsto tanh(V \times z + c) \end{aligned}$$

where the function *tanh* is applied element-wisely.
We define the weights and biases (for $l \in \mathbb{N}$):

$$\begin{array}{l} N = \{n_i\}_{i\in\{0,...,l+1\}} \in \mathbb{N}^{l+2}, n_0 = 1, n_{l+1} = p \\ W = \{W_i\}_{i\in\{1,...,l+1\}}, W_i \in \mathbb{R}^{n_i\times n_{i-1}} \\ B = \{b_i\}_{i\in\{1,...,l+1\}}, b_i \in \mathbb{R}^{n_i} \end{array} \tag{2}$$

$\Theta = (W, B)$ will be called the parameters of PINN. We will assume that there is Ω such that $\Theta \in \Omega$.

We can now define the different layers of PINN ($\forall i \in \{1, 2, ..., l + 1\}$):

$$\forall x \in \mathbb{R}^{n_{i-1}}, \mathcal{N}_i(x, \Theta) = \begin{cases} \mathcal{N}^{n_{i-1},n_i}(x, W_i, b_i) & \text{if } i \leq l \\ W_{l+1} \times x + b_{l+1} & \text{if } i = l + 1 \end{cases}$$

And the PINN function is the following:

$$\tilde{y} \colon \begin{array}{l} \mathbb{R} \times \Omega \to \mathbb{R}^p \\ (t, \Theta) \mapsto \tilde{y}(t, \Theta) = (\mathcal{N}_{l+1} \circ \mathcal{N}_l \circ ... \circ \mathcal{N}_2 \circ \mathcal{N}_1)(t, \Theta) \end{array} \tag{PINN}$$

A function that match this definition will be called in this paper a PINN with l hidden layers.

This function is differentiable and it is possible to compute $\tilde{y}'(t, \Theta)$ (i.e. $(D^\alpha_t \tilde{y})(t, \Theta)$ for $\alpha = 1$). We would like to be able to compute the value of $\forall t \in \mathbb{R}, \Theta \in \Omega, (D^\alpha_t \tilde{y})(t, \Theta)$ for $\alpha < 1$. But we can't compute the exact value of the integral, we need to approximate the solution. To do this Pakdaman M. et al. proposed in [D17] to use the following approximation:

$$\begin{array}{l} \alpha \in (0; 1), t \in \mathbb{R}_+, N_C \in \mathbb{N}, v = \{v_i\}_{i\in\{0,1,...,N_C\}} \in \mathbb{R}^{N_C}, 0 = v_0 < v_1 < .. < v_{N_C} < t \\ (D^\alpha_t \tilde{y})(t, \Theta) \approx \dfrac{1}{2\Gamma(1-\alpha)} \displaystyle\sum_{i=1}^{N_C} (v_i - v_{i-1}) \left((t - v_i)^{-\alpha} y'(v_i, \Theta) + (t - v_{i-1})^{-\alpha} y'(v_{i-1}, \Theta)\right) \end{array} \tag{3}$$

We would like to find a value of Θ such that $\forall t \in \mathbb{R}_+, \tilde{y}(t, \Theta) = y(t)$ (the $\tilde{y}(t, \Theta)$ function is the solution of the system of equation (SDE)). To find this value of Θ we will solve the following minimization problem:

$$\min_{\Theta\in\Omega} L_F(\Theta) + L_{MSE}(\Theta) \tag{4}$$

Where $L_{MSE}(\Theta)$ is called the data loss and is defined as follow:

$$L_{MSE}(\Theta) = \frac{\omega_{MSE}}{N_{data}} \sum_{i=1}^{N_{data}} \|\tilde{y}(s_i, \Theta) - y(s_i)\|^2, \tag{5}$$

where $\omega_{MSE} \in \mathbb{R}_+, N_{data} \in \mathbb{N}, S = \{s_i\}_{i\in\{1,\dots,N_{data}\}} \in \mathbb{R}^{N_{data}}$.

$L_F(\Theta)$ is called the governing equation loss and is defined as follow:

$$L_F(\Theta) = \frac{\omega_F}{N_F} \sum_{i=1}^{N_F} \|\mathcal{F}(t, \tilde{y}(p_i, \Theta), \tilde{y}(p_i - \tau, \Theta), (D_t^\alpha \tilde{y})(p_i, \Theta))\|^2, \tag{6}$$

where $\omega_F \in \mathbb{R}_+, N_F \in \mathbb{N}, P = \{p_i\}_{i\in\{1,\dots,N_F\}} \in \mathbb{R}_+^{N_F}$.

The points p_i and s_i need to be chosen by the user. A uniform sampling is generally used. The sum $L_F(\Theta) + L_{MSE}(\Theta)$ is simply called the loss.

According to the definition of the loss we know that the minimum will be positive or null. If we know a value of Θ such that $\tilde{y}$ is the unique solution of the system of equation (SDE) by definition we know that $L_F(\Theta) = L_{MSE}(\Theta) = 0$.

The resolution of the minimisation problem (4) will be called training. It will be made by an optimisation algorithm such as gradient descent.

2.3 Modified algorithm

In the previous section we presented the classical definition of the PINN function ($\tilde{y}(t, \Theta)$) and how to determine his parameters with an optimisation algorithm. In this section, we will present two modifications to the training that we used to improve the speed of the training of the PINN (i.e. reduce the number of iterations of the algorithm used to minimize the loss). We will call mPINN the PINN function trained with these two modifications.

First when, the differential equation is not linear, the PINN encounters difficulties to converge. This is due to the loss reaching a local minimum (see the figure 1). When it appends the data loss is much bigger than the governing equation loss, it means that the PINN predicts a valid solution of the equation but for different initial conditions (i.e it solves (SDE) with $y(0) = \tilde{y}(0)$ instead of $y(0) = y_0$) (see the figure 2). This is a local minimum because reducing the loss $L_{MSE}(\Theta)$ implies that the loss of $L_F(\Theta)$ increase. To solve this problem, we change the definition of the governing equation loss

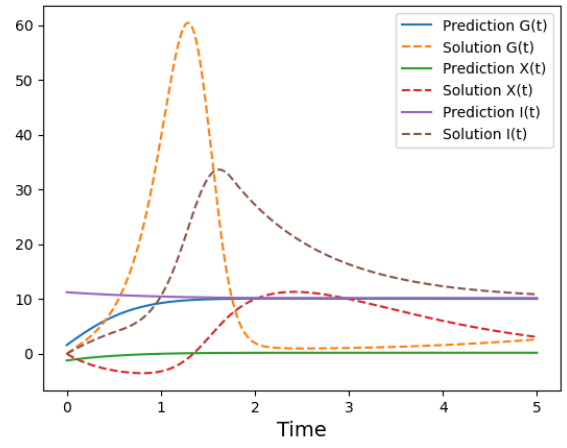

Figure 1: *Plot and the prediction (line) of the mPINN after 20 000 iterations and the solution (dot) for the Bergman minimal model with the initial conditions (0,0,0).*

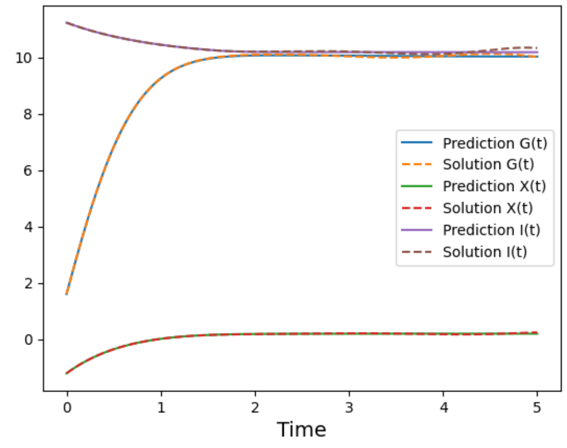

Figure 2: *Plot and the prediction (lines) of the mPINN after 20 000 iterations and the solution (dot) for the Bergman minimal model with the initial conditions (1.6032,-1.2139,11.2437).*

$$L_F(\Theta) = \frac{\omega_F}{N_F}\sum_{i=1}^{N_F}\|\mathcal{F}\left(t, \tilde{y}(p_i,\Theta) - \tilde{y}(0,\Theta) + g(0), \tilde{y}(p_i-\tau,\Theta) - \tilde{y}(0,\Theta) + g(0), (D_t^\alpha \tilde{y})(p_i,\Theta)\right)\|^2 \tag{7}$$

where $\omega_F \in \mathbb{R}_+, N_F \in \mathbb{N}, P = \{p_i\}_{i\in\{1,\dots,N_F\}} \in \mathbb{R}_+^{N_F}$.

This modification allows the PINN to learn the shape of the curve independently of the value of $\tilde{y}(0)$ and prevents the PINN to reach a local minimum.

The second new algorithm concern the approximation of the Caputo fractional derivative. With the classical algorithm (3) when α goes to 1, the value of $\Gamma(1-\alpha)$ becomes much larger that the approximation of the integral and leads to incorrect results (the approximation of the Caputo fractionnal derivative goes to zero when α goes to 1). To obtain accurate results even when α goes to 1 we propose the following approximation:

$$\begin{aligned}
&\alpha \in (0;1), t \in \mathbb{R}_+, N_C \in \mathbb{N}, V = \{v_i\}_{i\in\{0,1,\dots,N_C\}} \in \mathbb{R}^{N_C}, 0 = t_0 < v_1 < .. < v_{N_C} < t,\\
&({}_0^C D_t^\alpha \tilde{y})(t,\Theta) \approx \frac{1}{2\Gamma(1-\alpha)} \sum_{i=1}^{N_C} \left(\tilde{y}'(v_i,\Theta) + \tilde{y}'(v_{i-1},\Theta)\right) \int_{v_{i-1}}^{v_i} (t-s)^{-\alpha} ds,\\
&({}_0^C D_t^\alpha \tilde{y})(t,\Theta) \approx \frac{1}{2\Gamma(1-\alpha)\alpha} \sum_{i=1}^{N_C} \left(\tilde{y}'(v_i,\Theta) + \tilde{y}'(v_{i-1},\Theta)\right) \left((t-v_{i-1})^{1-\alpha} - (t-v_i)^{1-\alpha}\right).
\end{aligned} \tag{8}$$

This approximation uses the trapezoid method with the term $\tilde{y}'(s,\Theta)$. And the step size is replaced by the integral of $\int_{t_{i-1}}^{t_i} (t-s)^{m-1-\alpha} ds$.

In this section we have defined the classical PINN function and the training algorithm used to compute the parameters. And we proposed a new version of the training algorithm with these two modifications called mPINN that should allow to find best parameters of the function.

3 Verification and validation of our approach

In this section we will study the convergence of the mPINN and the approximation of the solutions. As a reminder, mPINN is a variant of PINN (a Deep learning algorithm). This algorithm use a Neural Network and informations about the system (governing equation, energy conservation,...) to compute the solution of systems of differential equations. The main advantage of this method is his permissiveness, this algorithm work with various systems of differential equations (with fractional derivatives, non linear,...). It can be used as a black box algorithm that can solve these systems for a non-specialist users. In this section we will use mPINN with $l = 2$ hidden layers. We will take $N_F = 401, \omega_{MSE} = 1, \omega_F = 0.01$.

In all this paper, all the examples will be computed in python3 with the Pytorch library.

3.1 First ordinary differential equation test

We will first consider the equation:

$$\begin{aligned}
&\frac{dy}{dt}(t) = -sin(t), \text{for } t > 0,\\
&y(0) = 1.
\end{aligned} \tag{9}$$

The exact solution is given by $y(t) = cos(t)$.
For this training we take $N_{data} = 1$ and $s_1 = 0$.

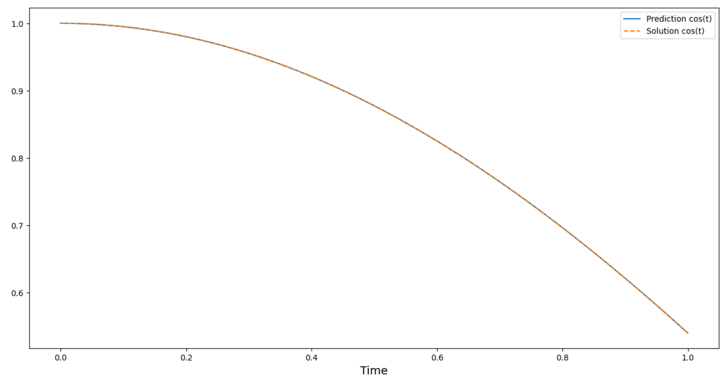

Figure 3: *Plot of the solution and the prediction of the mPINN after 4000 iterations on the interval [0,1].*

We can see that the prediction seems to be the same as the solution. The mean percentage error of the prediction (i.e. $\frac{1}{N}\sum_{i=0}^{N}\frac{|\tilde{y}(t_i)-y(t_i)|}{y(t_i)}$) is smaller than 0.02%. And the quadratic error is 4.7×10^{-4}.

We try with multiple values of N_F and even for $N_F = 10$ the approximation is accurate (the quadratic error is smaller than 10^{-3}). So for this simple problem the mPINN is able to quickly produce a precise approximation of the exact solution.

3.2 Non-linear ordinary differential equation test

We will now consider the Van Der Pol oscillation equation and try to solve it with the mPINN. The VDP equation is the following:

$$\begin{aligned} &\frac{d^2y}{dt^2} + \omega_0 y - \epsilon\omega_0(1-y^2)\frac{dy}{dt}(0) = 0, \text{for } t > 0, \\ &\frac{dy}{dt} = 0, y(0) = 1, \end{aligned} \tag{10}$$

where $\omega_0, \epsilon > 0$.

The governing equation is strongly non linear and the mPINN will require more iterations to converge. We take $N_{data} = 3$ and $s_1 = 0, s_2 = 0.067, s_3 = 0.133$. For this equation there is no exact solution. To compare the results of our approach, we will take the approximation made by a RK2 scheme with a discretisation of the interval in 3000 steps as the reference solution.

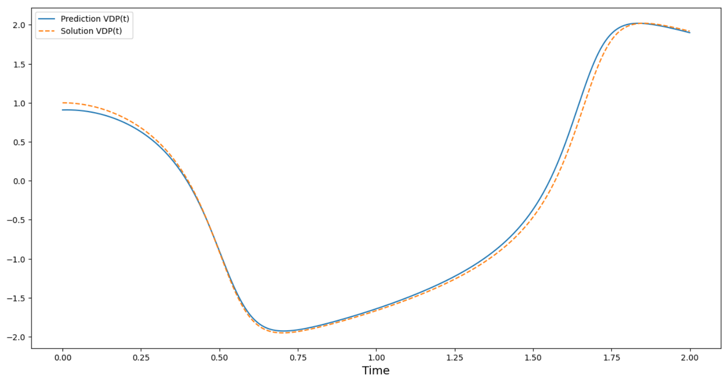

Figure 4: *Plot of the solution (yellow dot) and the prediction (blue line) of the mPINN after 20 000 iterations on the interval [0,2] with $\omega_0 = \pi$ and $\epsilon = 2$.*

For this equation, the mPINN needs more iterations to approximate the solution. After 20 000 iterations, the mean percentage error is 1.54%. This error can be reduced by making more iterations. Also using a shorter period of time ($[0, 1]$ for example) allows the mPINN to learn faster (about 5 000 iterations) this is due to the smoother shape on this shorter interval.

We will now study the impact of the number of collocation point N_F. To do this we will take several values of N_F and compute the error with the quadratic and the infinite norm.

N_F %	Quadratic	Infinite	Time (in seconds)[1]
50	5.32×10^{-2}	1.04×10^{-1}	71.75
100	3.22×10^{-2}	6.32×10^{-2}	75.45
200	1.16×10^{-2}	2.37×10^{-2}	86.26
400	6.68×10^{-3}	1.31×10^{-2}	90.11
1 000	6.39×10^{-3}	1.25×10^{-2}	119.72
2 000	1.01×10^{-3}	2.23×10^{-3}	145.48

Table 1: *Errors and time of computation of the prediction after 30 000 iterations for the VDP for different numbers of collocation points.*

The data in the table below are subject to big variations. For example, we run 14 times the training with the same parameters. The maximum quadratic error is 38 times bigger that the minimum quadratic error in these 14 observations. Also, the median of the 14 observations is 1.1535×10^{-3} and the standard error is 1.7×10^{-3} which is bigger. This is due to the evolution of the loss during the training, sometimes the loss can increase (up to 10 times its previous value) in a single iteration before reaching its older level in about a 1000 iterations. When this phenomenon occurs in the last 1000 iterations the prediction is less accurate. If it appends, by doing a few thousands of additional iterations the loss return to previous values.

[1]computation made with Python 3.11, torch 2.1.0+cu118 on a computer with 16Gb or RAM and an Intel Core i5-9400F CPU

When it appends for the table above, I rerun the training. When the number of collocation points increase the errors decrease as expected but the computation time increase. The infinite error seems to always be twice the quadratic error.

3.3 Delayed differential equation test

We will now study the solution of the following delayed differential equation, $1 > \tau > 0$:

$$\begin{aligned} &y'(t) = y(t) - y(t-\tau) + b, \text{for } t > 0, \\ &y(t) = 0, \text{for } t \in [-\tau, 0]. \end{aligned} \tag{11}$$

This equation admits the following unique global solution $y(t) = \frac{b}{1-\tau}t$.

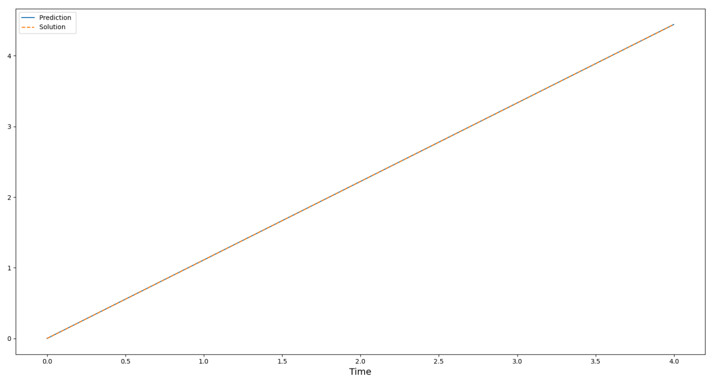

Figure 5: *Plot of the solution and the prediction of the mPINN after 5 000 iterations on the interval [0,4].*

This delayed differential equation is perfectly fits by the network in only 5 000 iteration.

3.4 Fractional differential equation test

Let us consider the following fractional differential equation(FDE):

$$\begin{aligned} &, ({}_0^C D_t^\alpha x)(t) = -x(t) + t^2 \frac{2t^{2-\alpha}}{\Gamma(3-\alpha)}, \text{for } t \in (0,1], \\ &x(0) = 0. \end{aligned} \tag{12}$$

This FDE admits the following solution $x(t) = t^2$.

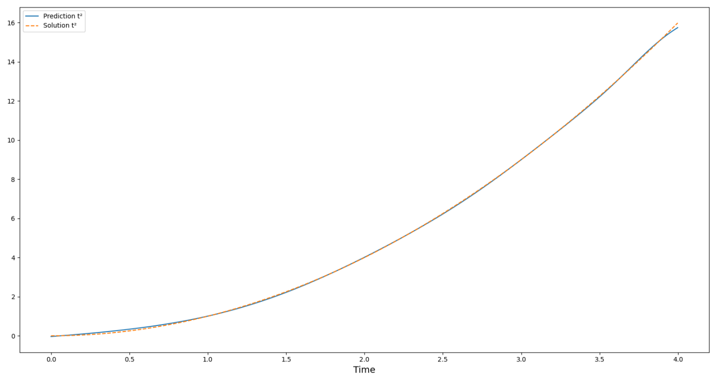

Figure 6: *Plot of the solution and the prediction of the mPINN after 5 000 iterations on the interval [0,4] for* $\alpha = 0.2$.

We can see on figure 6 that the prediction is accurate. The mean percentage error of the prediction is less than 0.4%.

On this part, we have shown that the mPINN can predict various types of differential equation. Also, the accuracy is lower for the mPINN than for traditional schemes like Euler or Runge Kutta (for the same computation time). The main drawback of the mPINN is its speed. A usual scheme (like Euler or Runge-Kutta) is faster than the training of the mPINN to approximate simple and more difficult differential equations. The mPINN can predict the solution of different types of systems of differential equation (fractionnal, with delay) without any change to the used by non-specialist (biologist, economist,...) to help them modeling and solving systems of differential equation without any restriction to their form.

4 Numerical simulation of some mathematical models of diabetes

In this, section we will use the mPINN to predict the solution of different models linked to diabetes. As a reminder, mPINN is a variant of PINN (a Deep learning algorithm). This algorithm use a Neural Network and information about the system (governing equation, energy conservation,...) to compute the solution of systems of differential equations. The main advantage of this method is its permissiveness, this algorithm work with various systems of differential equations (with fractional derivatives, non-linear,...). It can be used as a black box algorithm that can solve these systems for non-specialist users. The first model will be the DC model introduced in [AlS23]. This model is a simple representation of the evolution of the size of the population with diabetes disease. The goal of this analysis is to study the different behaviours of the models when modified (with fractional derivatives or with the addition of a delay term). We will do the same with the control Bergman model proposed by Penet M. in his thesis [M13] that represents the evolution of Glucose and Insulin rate in the blood depending on time.

We expect to see modifications in the model such as a change in the limits of the system, an apparition of a periodic pattern or a shift of the curve. If these modifications occur, they could be used to modified the models to make them fit better the real data.

4.1 DC Model

In this section we will use Physics Informed Neural Network (mPINN) to predict the evolution of the number of patient with diabetes.

The DC model (introduced in [AlS23]) is created for predicting the evolution of the number of patients with diabetes disease. The patients are divided into two classes C and D. D class is the number of patients without complications. C class is the number of patients with complications. For each class at each moment, a number I of new patients enter each of these classes and a proportion μ of patients die naturally. Also, at each moment a proportion λ of the patient develops a complication (switches from class D to class C). And a proportion γ of the patient with complications heals and returns to the D class. Inside the C class, a proportion δ died due to complication and a proportion ν became permanently disabled.

From this model we can derive the following system of differential equation:

$$\begin{aligned}&D'(t) = I - (\lambda + \mu)D(t) + \gamma C(t), \text{for } t > 0,\\&C'(t) = I - (\gamma + \mu + \delta + \nu)C(t) + \lambda D(t), \text{for } t > 0,\\&C(0) = C_0, D(0) = D_0.\end{aligned} \tag{13}$$

Or equivalently:

$$\begin{aligned}&C'(t) = I - (\theta + \lambda)C(t) + \lambda N(t), \text{for } t > 0,\\&N'(t) = 2I - (\nu + \delta)C(t) - \mu N(t), \text{for } t > 0,\\&C(0) = C_0, N(0) = N_0 = C_0 + D_0.\end{aligned} \tag{DC Model}$$

With $\theta = \gamma + \mu + \delta + \nu$ and $N(t) = C(t) + D(t)$. Also the systems (13) and (DC Model) are locally Lipschitz and thus the problems admit a unique global solution.

For convenience we will write:

$$\begin{aligned}&\mathcal{F}(t, C(t), N(t)) = \left(I - (\theta + \lambda)C(t) + \lambda N(t), 2I - (\nu + \delta)C(t) - \mu N(t)\right), \text{for } t > 0,\\&C(0) = C_0, N(0) = N_0.\end{aligned} \tag{14}$$

4.1.1 DC Model with specified parameters

For this study we will take only one data point at $t = 0$. This single point represents the initial conditions. We take $N_{data} = 0, N_F = 50, \forall i, p_i \in [0 : 20]$ and $\omega_{MSE} = 5\times 10^{-4}, \omega_F = 2.25\times 10^{-3}$.

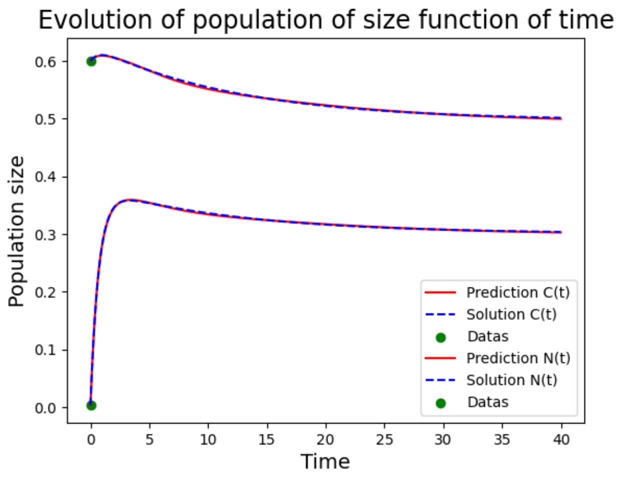

Figure 7: *Plot of the solution (blue dot) and the prediction of mPINN (red lines) for the model (DC Model) after 1 500 iterations for the parameters* $C_0 = 0.03$, $N_0 = 0.6$, $I = \mu = 0.02$, $\lambda = 0.85$, $\gamma = 0.5$ *and* $\nu = \delta = 0.05$

After 1 500 iterations, the approximation of the solution seems accurate. Even for t>20, outside the training range of the collocations points, we cannot see any difference between the prediction and the solution in figure 8. For t=10, we can see a small difference of the two curves that will be solved after 5 000 iterations but at the price of a lesser approximation outside of the learning range.

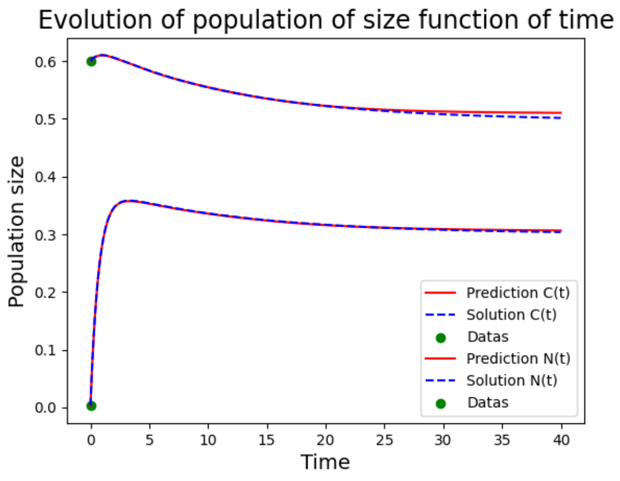

Figure 8: *Plot of the solution (blue dot) and the prediction of mPINN (red lines) for the model (DC Model) after 5 000 iterations for the parameters* $C_0 = 0.03$, $N_0 = 0.6$, $I = \mu = 0.02$, $\lambda = 0.85$, $\gamma = 0.5$ *and* $\nu = \delta = 0.05$

We can see on the table 2 that the predictions of the mPINN are accurate. A error of 0.12% is a very good approximation of the solution. The maximum error is of 6% and is for the initial

Difference %	mean	median	max
C	0.12	0.08	6.05
N	0.16	0.15	0.31

Table 2: *Percentage of error of mPINN predictions for the model (DC Model) after 5 000 iterations for the parameters* $C_0 = 0.03$*,* $N_0 = 0.6$*,* $I = \mu = 0.02$*,* $\lambda = 0.85$*,* $\gamma = 0.5$ *and* $\nu = \delta = 0.05$ *for t<20*

point, it means an error smaller that 2×10^{-3}. It is normal that this point has a bigger error because the associated loss is 4×10^{-6} which is of the same order of magnitude as the total loss. By resuming the training or setting ω_{MSE} to a higher value, we can reduce this error.

Difference %	mean	median	max
C	0.68	0.67	1.30
N	0.43	0.49	0.55

Table 3: *Percentage of error of mPINN predictions for the model (DC Model) after 5 000 iterations for the parameters* $C_0 = 0.03$*,* $N_0 = 0.6$*,* $I = \mu = 0.02$*,* $\lambda = 0.85$*,* $\gamma = 0.5$ *and* $\nu = \delta = 0.05$ *for* $t \in [20; 40]$

4.1.2 DC Model without specified parameters

On this part we will try to create a mPINN without having the need to specify all the parameters of the DC model (initial conditions or parameters such as $\lambda, I, ...$). This modification is also useless for the DC model because the exact solution can be computed. But it could be useful for more complex models if a single mPINN were able to predict the solution of multiple models.

To make it works we need to change the definition of the system of differential equations (SDE) by introducing a new parameter $v \in \mathbb{R}^q, q \in \mathbb{N}$ as follow:

$$\begin{aligned} &\mathcal{F}(t, y(t;v), y(t-\tau;v), (D_t^\alpha \tilde{y})(t;v); v) = (D_t^\alpha \tilde{y})(t;v) - f(t, y(t;v), y(t-\tau;v); v) = 0, \text{for } t > 0, \\ &y(t;v) = g(t;v), \text{for } t \in [-\tau; 0]. \end{aligned} \tag{mSDE}$$

v represents the parameters of the models (for the DC model $v = (\theta, \lambda, \nu, \mu, C_0, N_0)$).

The PINN function becomes:

$$\begin{aligned} \tilde{y}\colon\ &\mathbb{R} \times \mathbb{R}^q \times \Omega \to \mathbb{R}^p \\ &(t, \Theta; v) \mapsto \tilde{y}(t, \Theta; v) = (\mathcal{N}_{l+1} \circ \mathcal{N}_l \circ ... \circ \mathcal{N}_2 \circ \mathcal{N}_1)(t; v) \end{aligned} \tag{pPINN}$$

And finally the two losses will become:

$$L_{MSE}(\Theta) = \frac{\omega_{MSE}}{N_{data}} \sum_{i=1}^{N_{data}+N_{data0}} \|\tilde{y}(s_i, \Theta; v_i) - y(s_i; v_i)\|^2, \tag{15}$$

where $\omega_{MSE} \in \mathbb{R}_+, N_{data}, N_{data0} \in \mathbb{N}, S = \{s_i\}_{i \in \{1,...,N_{data}+N_{data0}\}} \in \mathbb{R}^{N_{data}+N_{data0}}, V_{MSE} = \{v_i\}_{v \in \{1,...,N_{data}+N_{data0}\}} \in \mathbb{R}^{q \times (N_{data}+N_{data0})}$.

$$L_F(\Theta) = \frac{\omega_F}{N_F}\sum_{i=1}^{N_F} \|\mathcal{F}(t, \tilde{y}(p_i, \Theta; v_i), \tilde{y}(p_i - \tau, \Theta; v_i), (D_t^\alpha \tilde{y})(p_i, \Theta; v_i); v_i)\|^2, \tag{16}$$

where $\omega_F \in \mathbb{R}_+, N_F \in \mathbb{N}, P = \{p_i\}_{i\in\{1,\dots,N_F\}} \in \mathbb{R}_+^{N_F} V_F = \{v_i\}_{v\in\{1,\dots,N_{data}\}} \in \mathbb{R}^{q\times N_F}$.

N_{data} training data points are generated by computing random parameters t_i and v_i and generating the solution for the model with the parameters v_i at time t_i. And N_{data0} points are generated the same way but for $t_i = 0$. If we don't add data points with $t_i = 0$, the initial condition are poorly predicted by the network. The collocation point are generated the same way that the data points.

In this part consider the DC model where we fix the parameters $N_0 = 0.5, I = \mu = 0.02, \gamma = 0.5$ and $\nu = \delta = 0.05$. So $v = (\lambda, I, C_0)$.

For the training, we take $N_{data} = 1400, N_0 = 600, N_F = 6000$ for the first 20000 iterations and we take $\omega_{MSE} = 5 \times 10^{-4}, \omega_F = 5 \times 10^{-3}$ and for the next 10000 iterations $\omega_{MSE} = 5 \times 10^{-5}, \omega_F = 5 \times 10^{-4}$.

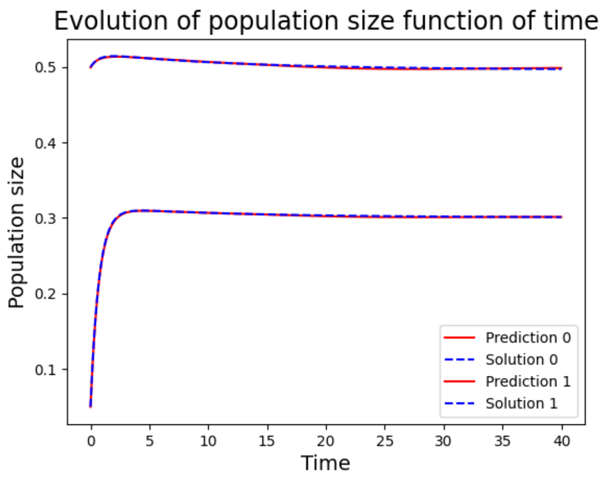

Figure 9: *Plot of the solution (blue dot) and the prediction of mPINN (red lines) for the model (DC Model) after 30 000 iterations for the parameters* $C_0 = 0.05$, $N_0 = 0.5$, $I = \mu = 0.02$, $\lambda = 0.85$, $\gamma = 0.5$ *and* $\nu = \delta = 0.05$.

We can see in figure 9 that the prediction of the mPINN seems to be really accurate. We can only see that between $t = 15$ and $t = 30$ the prediction is slightly below the solution.

The network conserves errors with the same order of magnitude as the network from the previous part. But this is not always the case, one can consider $I = 0.04, C_0 = 0.45$ and $\lambda = 0.2$.

Difference %	mean	median	max
C	0.16	0.10	0.42
N	0.16	0.13	0.37

Table 4: *Percentage of error of mPINN predictions for the model (DC Model) after 30 000 iterations for the parameters* $C_0 = 0.05$*,* $N_0 = 0.5$*,* $I = \mu = 0.02$*,* $\lambda = 0.85$*,* $\gamma = 0.5$ *and* $\nu = \delta = 0.05$ *for* $t \in [0; 40]$

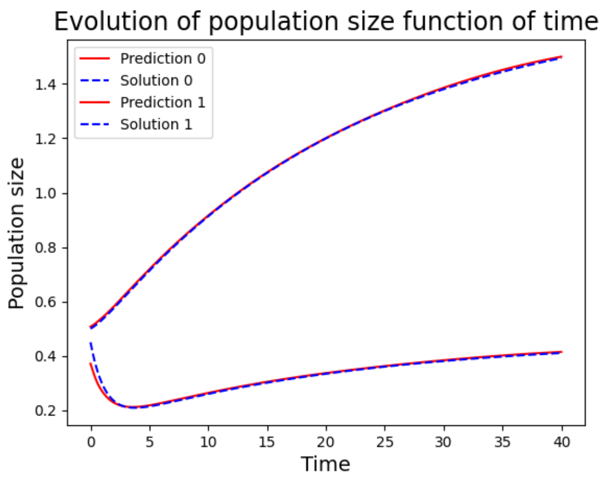

Figure 10: *Plot of the solution (blue dot) and the prediction of mPINN (red lines) for the model (DC Model) after 30 000 iterations for the parameters* $C_0 = 0.45$*,* $N_0 = 0.5$*,* $I = 0.04, \mu = 0.02$*,* $\lambda = 0.2$*,* $\gamma = 0.5$ *and* $\nu = \delta = 0.05$*.*

We can see in the figure 10, when $t = 0$, the initial data are not well predicted. For $t < 3$ there is a noticeable difference between the prediction and the solution. After that the prediction is really accurate.

Difference %	mean	median	max
C	0.54	0.33	7.75
N	0.23	0.23	0.96

Table 5: *Percentage of error of mPINN predictions for the model (DC Model) for the model DC Model after 30 000 iterations for the parameters* $C_0 = 0.45$*,* $N_0 = 0.5$*,* $I = \mu = 0.04$*,* $\lambda = 0.2$*,* $\gamma = 0.5$ *and* $\nu = \delta = 0.05$ *for* $t \in [0; 40]$

Difference %	mean	median	max
C	0.34	0.31	1.01
N	0.21	0.21	0.42

Table 6: *Percentage of error of mPINN predictions for the model (DC Model) after 30 000 iterations for the parameters* $C_0 = 0.45$*,* $N_0 = 0.5$*,* $I = \mu = 0.04$*,* $\lambda = 0.2$*,* $\gamma = 0.5$ *and* $\nu = \delta = 0.05$ *for* $t \in [2.7; 40]$

The difference between C_0 and the prediction is important (more than 7%). but if we consider the time interval $[2.7; 40]$ (see figure 6) the maximum percentage error becomes 1% and the mean error decreases.

This new network is interesting because it can accurately predict the solution of the DC model for different sets of parameters. This ability allows the user to compute different population evolution scenarios by changing the parameters without needing to train a new mPINN. The main downside is that the initial training is very slow because the dataset is bigger than the one used for the mPINN without parameter tuning. We also need to make more iterations to obtain the same accuracy. And for certain parameters, in the beginning of the time interval the prediction might be inaccurate.

4.1.3 DC with fractional derivative

In this section we will modify the DC model. To do that we will use Caputo fractional derivatives ($\alpha < 1$ in SDE) instead of the usual derivatives. The fractional derivatives are used to models complex phenomena with long-term memory. We can expect to observe a shift to the right of the solution, a change of the limits and even the apparition of a periodic pattern (long-term memory).

We can define the new DC model as follow:

$$\begin{aligned} &({}_0^C D_t^\alpha C)(t) = I - (\theta + \lambda)C(t) + \lambda N(t), \text{for } t > 0, \\ &({}_0^C D_t^\alpha N)(t) = 2I - (\nu + \delta)C(t) - \mu N(t), \text{for } t > 0, \\ &C(0) = C_0, N(0) = N_0 = C_0 + D_0. \end{aligned} \tag{Fractional DC Model}$$

Where $0 < \alpha < 1$.

To find a solution to this equation, we will also use mPINN. We can't directly use normal ODE algorithm like Runge-Kutta with this model. We can still use mPINN to predict the solution. To do this we will use the scheme presented in the section 1 in order to compute the value of $({}_0^C D_t^\alpha \tilde{y})(t, \Theta)$ where Θ is the set of parameters of our mPINN and $\tilde{y}(t, \Theta) = (\tilde{C}(t, \Theta), \tilde{N}(t, \Theta))$ represent our mPINN evaluated at the time t with the parameters Θ.

We can see in figure 11 that the modification of α changes the limits of the model and the smoothness of the curve. Also for low values of α the curve seems to converge faster to its limit.

We can clearly see that the change of α modifies the behaviour of the model. But this change of behaviour do not seems to add any shift.

4.1.4 DC with delay

Another way to modify the model is to use delayed differential equation instead of ordinary differential equation. Delayed differential equations are used to induce delay in a response to a signal or induce periodic oscillation in the solution. They are widely used to in biology or in engineering. We can expect to observe a change of the limits, a modification of the speed of convergence and even the apparition of periodic oscillations.

We will modify the DC system by adding two new terms. We define $\tau > 0$.

$$\begin{aligned} &C'(t) = I - (\theta + \lambda)C(t) + \lambda N(t) + a_C C(t - \tau), \text{for } t > 0, \\ &N'(t) = 2I - (\nu + \delta)C(t) - \mu N(t) + a_N N(t - \tau), \text{for } t > 0, \\ &C(t) = C_0, N(t) = N_0, \text{for } t \in [-\tau; 0]. \end{aligned} \tag{Delayed DC Model}$$

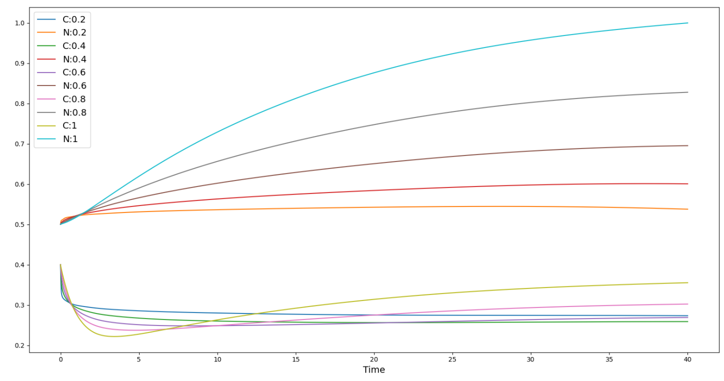

Figure 11: *Plot of the prediction of mPINN for the model (Fractional DC Model) for different values of α for the parameters $C_0 = 0.4$, $N_0 = 0.5$, $I = 0.03, \mu = 0.02$, $\lambda = 0.3$, $\gamma = 0.5$ and $\nu = \delta = 0.05$.*

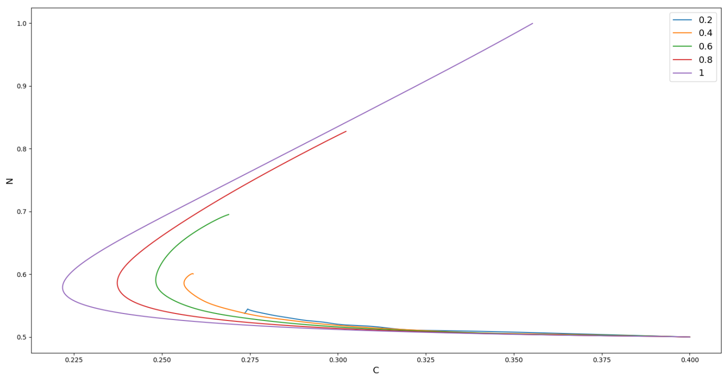

Figure 12: *Phase portrait of the prediction of mPINN for the model (Fractional DC Model) for different values of α for the parameters $C_0 = 0.4$, $N_0 = 0.5$, $I = 0.03, \mu = 0.02$, $\lambda = 0.3$, $\gamma = 0.5$ and $\nu = \delta = 0.05$.*

On this model the parameters a_N and a_C belong to $\mathbb{R}$.

We can see on these figures 14 and 13 that the behaviour of the model strongly depends on the values of a_N and a_C. When $a_N = -0.05$ and $a_C = -0.5$ we can see that a shift to the right seems to appear for $C(t)$. The main modification of this model is the change of the asymptotic values of the model.

For the DC Model the addition of the fractional derivative or delay mainly modifies the limits of the system and the speed of the convergence to its limits. Without real data we cannot compare the models together.

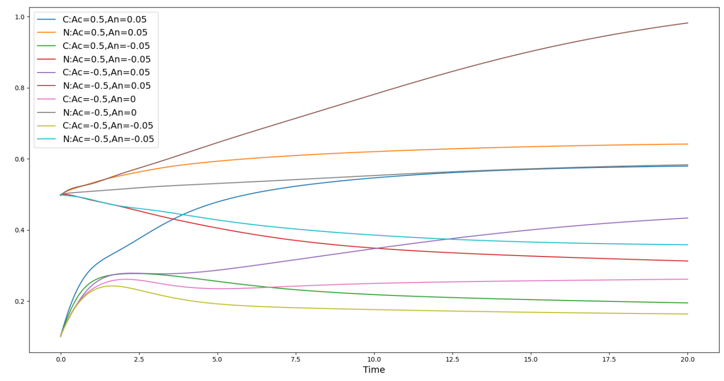

Figure 13: *Plot of the prediction of mPINN for the model (Delayed DC Model) for different values of* a_N, a_C *for the parameters* $C_0 = 0.4$, $N_0 = 0.5$, $I = 0.03, \mu = 0.02$, $\lambda = 0.3$, $\gamma = 0.5$, $\nu = \delta = 0.05$ *and* $\tau = 2$.

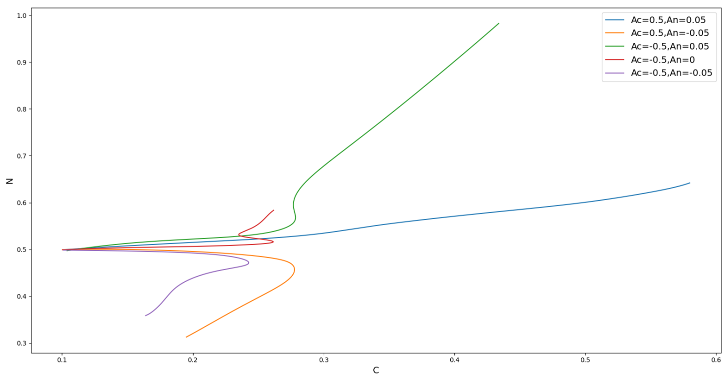

Figure 14: *Phase portrait of the prediction of mPINN for the model (Delayed DC Model) for different values of* a_N, a_C *for the parameters* $C_0 = 0.4$, $N_0 = 0.5$, $I = 0.03, \mu = 0.02$, $\lambda = 0.3$, $\gamma = 0.5$, $\nu = \delta = 0.05$ *and* $\tau = 2$.

4.2 Bergman Control Model

In this section we will study a new model presented by Penet M. in [M13]. This model represents the evolution of the glucose insulin depending on time. This model is suitable to a patient with a type 1 diabetes. For this patient, the input of insulin comes from a subcutaneous injection. Here we make the hypothesis that the subcutaneous flow is constant over time (with value u). This model takes the form of a non-linear ordinary differential equation. The objective of this section is to study the impact of the modification of the derivatives to fractional derivatives and the use of delayed differential equation. To do the computation we will use mPINNs.

This model can be written as:

$$
\begin{aligned}
&\frac{dG}{dt} = -(P_1 + X)G + P_1 G_b + R_a(t), \text{for } t > 0,\\
&\frac{dX}{dt} = -P_2 X + P_3(I - I_b), \text{for } t > 0,\\
&\frac{dI}{dt} = -k_f I + b_f U, \text{for } t > 0,\\
&\frac{dU}{dt} = -k_s U + u, \text{for } t > 0,\\
&(G, X, I, U)(0) = (G_0, X_0, I_0, U_0).
\end{aligned} \tag{BCM}
$$

The variable G represents the blood glucose concentration, the variable I represents the blood insulin, X represents time-lag of the insulin consumption on glucose and finally U stands for the flow of the subcutaneous insulin. R_a stands for the glucose flow in the blood, and the parameters $P_1, P_2, P_3, G_b, k_s, k_f, b_f, u, I_b$ are positive constant parameters of the model.

On this model, the most interesting parameter is the function R_a. The choice of this function can lead to very different behaviours of the solution.

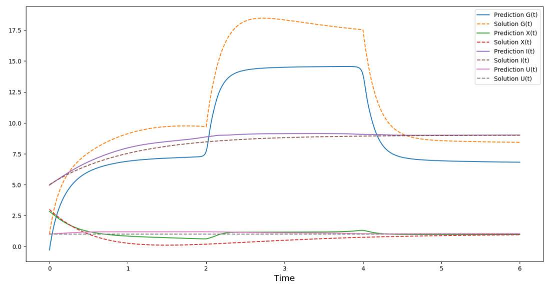

Figure 15: *Plot of the prediction of mPINN for the model (BCM) after 5 000 iterations for the parameters $Ra(t) = \mathbb{1}_{[2;4]}(t)$, $P_1 = 5$, $P_2 = P_3 = 1, G_b = 10, I_b = 8$ and $B_f = K_f = K_s = u = 1$.*

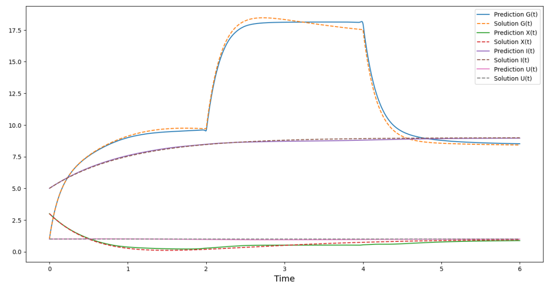

Figure 16: *Plot of the prediction of mPINN for the model (BCM) after 10 000 iterations for the parameters $Ra(t) = \mathbb{1}_{[2;4]}(t)$, $P_1 = 5$, $P_2 = P_3 = 1, G_b = 10, I_b = 8$ and $B_f = K_f = K_s = u = 1$.*

As shown in the figures 15,16 and 17, the mPINN takes about 30 000 iterations to approximate the solution with $\omega_{MSE} = 2, \omega_F = 5$ and $N_F = 300$. The high values of the ω_F and ω_{MSE} are

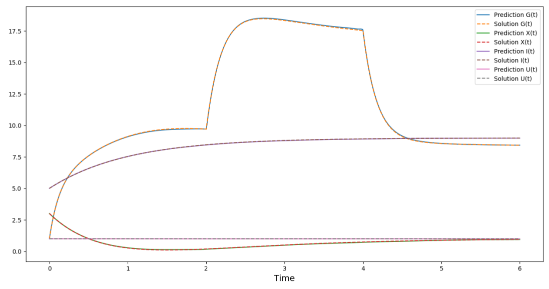

Figure 17: *Plot of the prediction of mPINN for the model (BCM) after 30 000 iterations for the parameters* $Ra(t) = \mathbb{1}_{[2;4]}(t)$*,* $P_1 = 5$*,* $P_2 = P_3 = 1, G_b = 10, I_b = 8$ *and* $B_f = K_f = K_s = u = 1$*.*

used to increase the speed of the convergence when the the value of the loss become small. The number of iteration used to obtain a good approximation is bigger than the number iterations made for the DC model. This increase can be explained by the non-linearity of the Bergman control model.

4.2.1 Bergman model with fractional order derivative

In this section we will modify the model FBCM. To do that we will use Caputo fractional derivatives ($\alpha < 1$ in (SDE)) instead of the usual derivatives. The fractional derivatives, are used to models complex phenomena with long-term memory. We can expect to observe a shift to the right of the solution a change of the limits and even the apparition of a periodic pattern(long-term memory).

The new model follow this system: (where $0 < \alpha < 1$)

$$\begin{aligned}
&({}_0^C D_t^\alpha G)(t) = -(P_1 + X)G + P_1 G_b + R_a(t), \text{for } t > 0,\\
&({}_0^C D_t^\alpha X)(t) = -P_2 X + P_3(I - I_b), \text{for } t > 0,\\
&({}_0^C D_t^\alpha I)(t) = -k_f I + b_f U, \text{for } t > 0,\\
&({}_0^C D_t^\alpha U)(t) = -k_s U + u, \text{for } t > 0,\\
&(G, X, I, U)(0) = (G_0, X_0, I_0, U_0).
\end{aligned} \tag{FBCM}$$

We can see in figure 18, that for $\alpha = 0.2$ or $\alpha = 0.4$ the curve converge quickly to its limit. Also the solution of the new model do not seems to shift the solution to the right but it changes the limits of this models.

4.2.2 Delayed Bergman model

In this section we will study the delayed version of the control Bergman Model the model is the following:

$$\begin{aligned}
&\frac{dG}{dt} = -(P_1 + X)G + P_1 G_b + R_a(t) + c_G G(t - \tau), \text{for } t > 0,\\
&\frac{dX}{dt} = -P_2 X + P_3(I - I_b) + c_X X(t - \tau), \text{for } t > 0,\\
&\frac{dI}{dt} = -k_f I + b_f U + c_I I(t - \tau), \text{for } t > 0,\\
&\frac{dU}{dt} = -k_s U + u + c_U U(t - \tau), \text{for } t > 0,\\
&(G, X, I, U)(t) = (G_0, X_0, I_0, U_0), \text{for } t \in [-\tau; 0],
\end{aligned} \tag{DBCM}$$

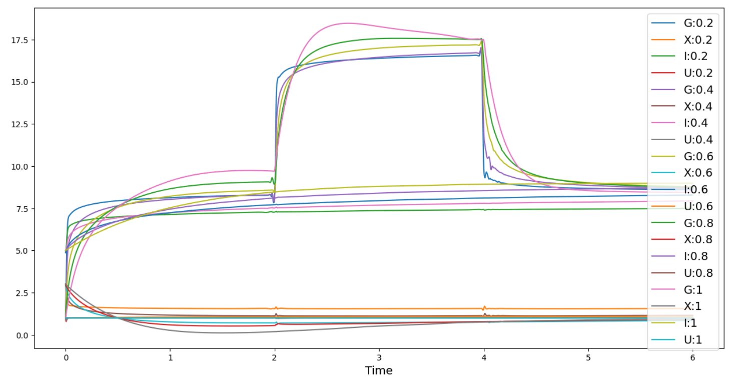

Figure 18: *Plot of the prediction of mPINN for the model (FBCM) for different values of α for the parameters $Ra(t) = \mathbb{1}_{[2;4]}(t)$, $P_1 = 5$, $P_2 = P_3 = 1, G_b = 10, I_b = 8$ and $B_f = K_f = K_s = u = 1$.*

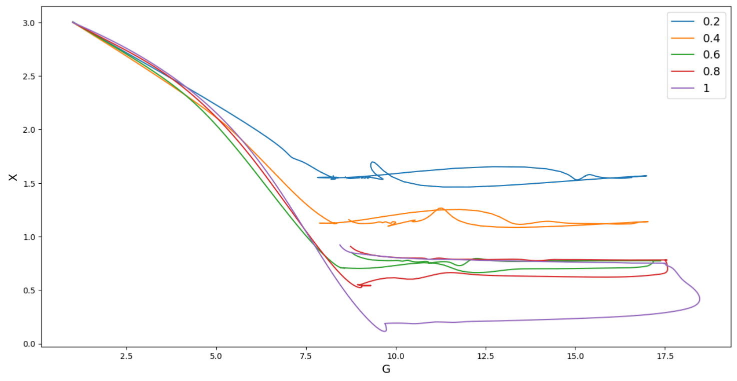

Figure 19: *Plot of the prediction of mPINN for the model (FBCM) for different values of α for the parameters $Ra(t) = \mathbb{1}_{[2;4]}(t)$, $P_1 = 5$, $P_2 = P_3 = 1, G_b = 10, I_b = 8$ and $B_f = K_f = K_s = u = 1$.*

In this study, for simplicity of the analysis we use the same values of $c_G = c_X = c_I = c_U = c$. In the following analysis we will take $N_F = 501$, $P_1 = 5, P_2 = P_3 = b_f, k_f = k_s = u = 1, G_b = 20, I_b = 8$ and $R_a(t) = 50(\mathbb{1}_{]3,4[} + \mathbb{1}_{]6,7[})$. The time interval will be $[0, 10]$.

We will first study the impact of τ on the behaviour of the solution we will take $c = 1$.

We can see in figure 20 that the modification of τ strongly modifies the solution of the equation. A larger value of τ seems to add more instability to the the solution but do not shift of the solution appears.
Also for $I(t)$ the addition of the delay change the concavity of the solution which is now convex.
Now we will take $\tau = 2$ and study the impact of c.

We can see in figure 21 that the value of c also affects the behaviour of the system. A larger

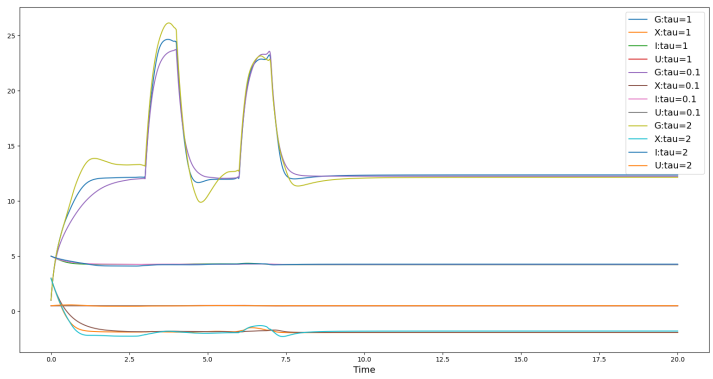

Figure 20: *Plot of the prediction of mPINN for the model (DBCM) for different values of* τ*.*

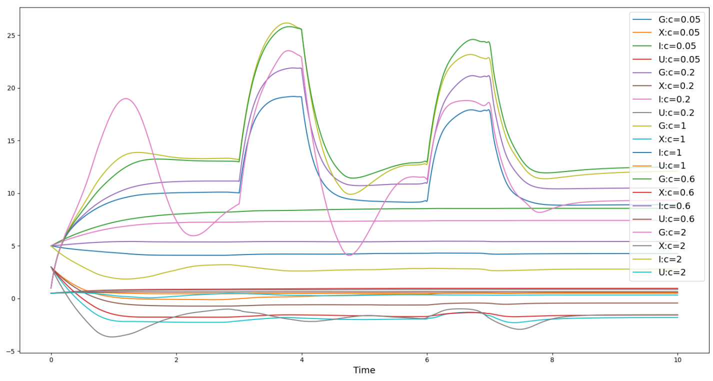

Figure 21: *Plot of the prediction of mPINN for the model (DBCM) for different values of* c*.*

value of c seems to add oscillations. Also considering the G component of system, the values of this component depends of the value of c. For $c = 0.6$ G as its maximum values. Also we can see that for $c \in \{0.05, 0.2, 0.6\}$ the $I(t)$ curve is concave and convex for $c \in \{1, 2\}$.

On the phase portrait 22 we can clearly observe the change of convexity depending of the values of c.

4.3 Analysis conclusion

In this section we have studied two models. The modification made to this two models mainly changes the values of the system and the speed of the convergence to the system limits but do not add any shift. With the delayed differential equation, we manage to add oscillations to the solution. Without any data it is difficult to conclude on the use of these modifications in real case applications. As for the previous part, the mPINN have been slow to make prediction but is versatile in its adaptation to different type of models. This versatility could be used to make

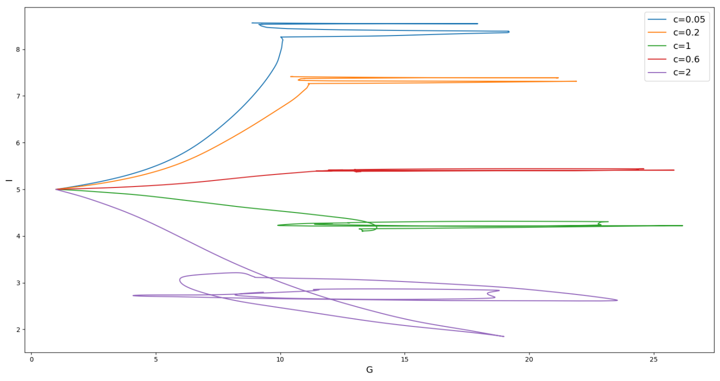

Figure 22: *Plot of the phase portrait (with respect to G and I) of mPINN for the model (DBCM) for different values of c.*

a unique algorithm that could solve different type of system of differential equation.

5 Conclusion

In this paper, we have used mPINN to compute the solutions of different systems of differential equations. We have shown in this experimentation that the mPINN is not efficient to compute solutions for ordinary or delayed differential equations. But he can be used to compute the solutions of fractional differential equations, whereas usual schemes (like Runge-Kutta) cannot solve them. Also mPINN can be used to predict different type of differential equation. This property can be used to make a simple software which can take every type of differential equation and use mPINN to solve it.

We also improve the training. First, we decorrelate the two losses to stop their sum from reaching a local minimum. Secondly we used a better approximation of the Caputo fractional derivative which works with values of α close to 1. We also quickly compare the impact of the number of collocation points for the computation time and the error.

Finally, we use the mPINNs to study two models related to diabetes. We also modify them and see the impacts. We have been able to train a single mPINN to fits solutions of multiple DC model (with different parameters). By using fractional differential equation and delayed equation in our models, we obtain different modifications such as a change of the limits, speed of the convergence and the creation of oscillations. But we did not manage to induce a shift or the apparition of a periodic pattern as wanted.

It could be interesting for a future research to continue to improve the algorithm and study if changing the number of collocation points during the training could lead to less computation time and more accurate results.

Bibliography

[M13] Penet M. “Commande Prédictive Nonlin´eaire Robuste par Méthode de Point Selle en Optimisation sous Contraintes : Analyse de Stabilité et Application au Diabète de Type 1”. 2013.

[D17] Pakdaman M. Ahmadian A. Effati S. Salahshour S. Baleanu D. “Solving differential equations of fractional order using an optimization technique based on training artificial neural network”. In: *Fast Gradient-Based Algorithms for Constrained Total Variation Image Denoising and Deblurring Problems* (2017).

[Ang21] Shelyag S. Angelova M. “Delay-differential equations for glucose-insulinregulation”. In: *Springer* (2021).

[AlS23] Mortula M. Husseini G. A. AlShurbaji M. Abdul Kader L. Hannan H. “Comprehensive Study of a Diabetes Mellitus Mathematical Model Using Numerical Methods with Stability and Parametric Analysis”. In: *International Journal of Environmental Research and Public Health* (2023).

[Bat23] Baty L. Baty H. “Solving differential equations using physics informed deep learning: a hand-on tutorial with benchmark tests”. In: *arXiv* (2023).