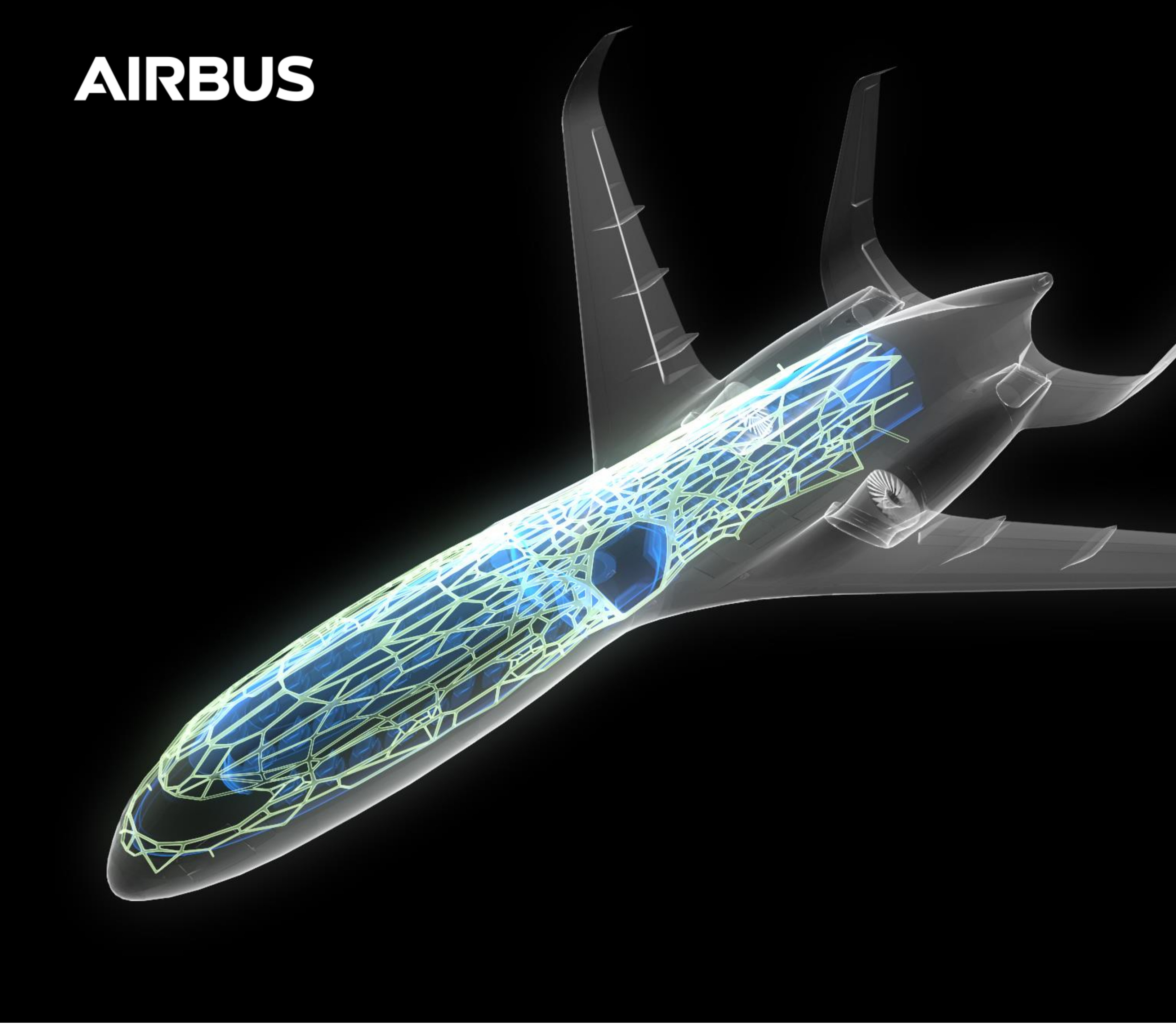

AIRBUS QUANTUM COMPUTING CHALLENGE

Problem Statement n°3

Application of Quantum Computing to surrogate modeling of Partial Differential Equations

1. Problem statement

Partial Differential Equations (PDE) are used to represent a wide variety of physical phenomena such as aerodynamic flows. In this context, PDE are numerically solved by approaches such as the Finite Difference Method (FDM), Finite Volume Method (FVM) or Finite Element Method (FEM). Any of them relies on discretizing the space domain in which the equation is solved by means of a mesh or grid. The solution is then approximated in a discrete set of time instants. Subsequently, the continuous mathematical operators are also approximated using the so-called numerical schemes. Such methods have been deeply studied and developed, but their accuracy and robustness strongly rely on the refinement level of the discretization. The associated compu-tational cost grows as finer meshes and smaller time steps are required. This makes disciplines such as Computational Fluid Dynamics (CFD) very time and resource consuming tasks even for High Performance Computing (HPC) clusters.

Efforts are frequently done in aerodynamics to reduce the cost of numerical simulations by using surrogate modeling. Here we draw attention to a recent approach [1] that consists in using machine learning techniques for solving a PDE problem. This approach is based on building an approximate representation of the solution by means of an artificial Neural Network (NN). From this perspective, solving the PDE is turned into a regression problem for which classical machine learning algorithms can be applied. As it will be detailed, this removes the need of using a mesh as only randomly sampled points in the time and space domains are required. Furthermore, the spatial and time dimensions are treated simultaneously. As a consequence, the classical notions of numerical schemes disappear and are replaced by a new approach.

Nevertheless, this approach of surrogate modelling brings new challenges to overcome. In par-ticular, large neural networks and training times might be required to build a sufficiently accurate representation of the PDE. In this context, we aim at investigating how Quantum Computing (QC) algorithms can be applied to the generation and training of such neural networks. The use case case to assess the quality of the developed QC algorithms is the solution of the Burgers equation, which models some particular cases of aerodynamic flows.

2. Problem formulation

In this section we describe how a PDE problem in its most general formulation can be translated into a machine learning problem.
Let a generic PDE problem be formulated as follows for the scalar variable u(t, x) in the time and n-dimensional space domain ($x \ \epsilon \ \Omega \ \subset \ \Re^n$):

$$\frac{\partial u}{\partial t}(t,x) + \mathcal{L}u(t,x) = 0 \ \ in \ \ (t,x) \ \epsilon \ [0,T] \times \Omega \qquad (1)$$

$$u(0,x) = u_0(x) \ \ in \ \ x \ \epsilon \ \Omega \qquad (2)$$

$$u(t,x) = g(t,x) \ \ in \ \ (t,x) \ \epsilon \ [0,T] \times \partial\Omega \qquad (3)$$

Where $\mathcal{L}$ is an operator on u and its space derivatives that can be non-linear, $u_0(x)$ is the initial condition and $g(t,x)$ stands for the boundary condition of the problem.

The main idea behind the use of the deep learning based algorithms for solving PDE is to turn this problem into a regression problem, for which classical machine learning algorithms can be applied. For this, we assume that the solution of the equation $u(t,x)$ can be approximated by a feed-forward neural network. It is reminded that neural networks can approximate any function with arbitrary accuracy it the learning problem is well-posed [3]. Such neural network is defined by a set of tunable parameters θ also known as *weights* and *biases* and shall predict the value of the function u when fed with a time instant and a spatial location.

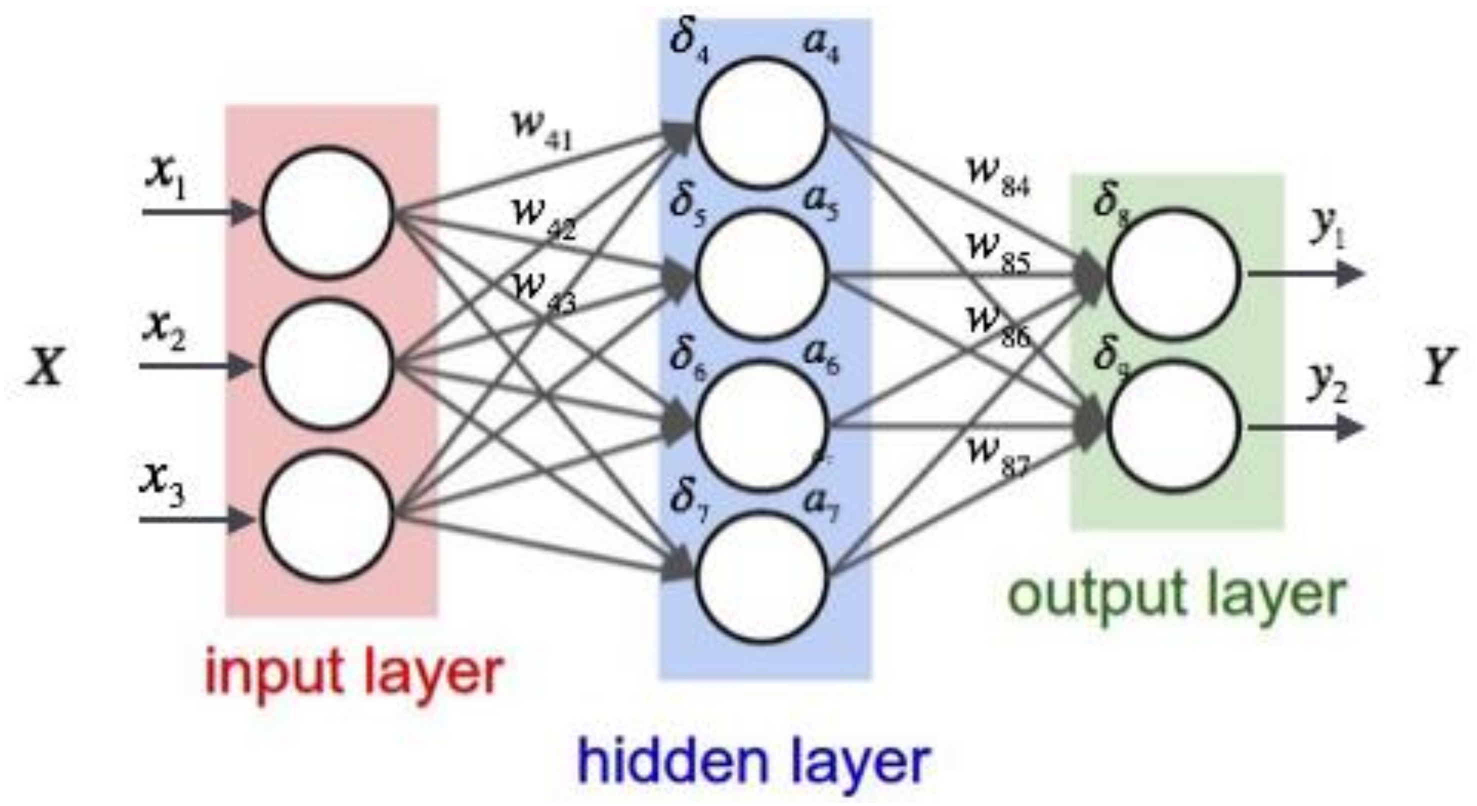

Figure 1: Feedforward Neural Network

In mathematical terms, the regression problem can be formulated as follows:

- Given a PDE problem as expressed in equation 1, we assume that $u(t,x) \approx f(t,x;\theta)$, where $f(t,x;\theta)$ is a feed-forward neural network as displayed in figure 1.
- The approximation error of $f(t,x;\theta)$ is measured by a cost function $J(f)$ that evaluates how the neural network fits the equation, the initial and the boundary conditions in the considered domain. The cost function can be formulated as follows:

$$J(f) = ||\frac{\partial f}{\partial t} + \mathcal{L}f||^2_{2,[0,T]\times\Omega} + ||f - g||^2_{2,[0,T]\times\partial\Omega} + ||f(0,) - u_0||^2_{2,\Omega} \quad (4)$$

- If $J(f) = 0$ then f is a solution of the PDE. Thus, the role of the machine learning algorithm is to find the parameters θ that minimize the value of the cost function, as $J(f) \to 0$, then $f \to u$ meaning that the neural network is an accurate approximation of the solution.

2.1 Machine learning-based reference algorithm

Based on the problem formulation mentioned above, a recent contribution [1] introduced an algorithm for training feedfoward neural networks to solve a PDE using supervised learning principles. This technique is presented here.
Firstly, the neural network parameters θ are initialized randomly. Then, for $n = 1, \dots, N$ steps until a convergence criterion is satisfied:

1. Generate the set of random points: (t_n, x_n) from $[0, T] \times \Omega$, (τ_n, z_n) from $[0, T] \times \partial\Omega$ and w_n from Ω according to random distributions of probability densities ν_1, ν_2 and ν_3 respectively.

2. Calculate the value of the cost function $J(\theta_n)$ over the set of points considered before, defined as:

$$J(\theta_n) = (\frac{\partial f}{\partial t}(t_n, x_n; \theta_n) + \mathcal{L}f(t_n, x_n; \theta_n)^2 + (f(\tau_n, x_n; \theta_n) - g(\tau_n, z_n))^2 + (f(0, w_n; \theta_n) - u_0(w_n))^2 \tag{5}$$

3. Back-propagate the loss function so as to obtain the contribution of each network parameter to the error using a learning rate ϵ and update the parameters accordingly

$$\theta_{n+1} = \theta_n - \epsilon \, \nabla_\theta \, J(\theta_n) \tag{6}$$

A key feature of this algorithm is that no dataset neither mesh is required for training the neural network. Instead, only a subset of points randomly sampled from the time and space domains are needed to assess whether the neural network output fits the PDE, its initial condition and its boundary conditions. Furthermore, the algorithm relies on well-know machine learning methods such as backpropagation and stochastic gradient descent. These methods are usually implemented in standard machine learning libraries such as TensorFlow [2]. In particular, this library also provides tools to automatically differentiate the neural network, making the computation of the network spatial and time derivatives straightforward.

Contrary to these benefits, builing a high fidelity model of a PDE using this approach can be computationally expensive as large neural networks and time-consuming training algorithms might be required. Therefore, the goal of this problem is to explore QC algorithms for PDE surrogate modelling that could alleviate the needs of complexity of the neural networks or provide more efficient training methods.

3. Case study: solving the 1-D Burgers equation

3.1 Mathematical problem

The Burgers equation can be interpreted as a simplified equation of fluid dynamics. In the 1-D case, it represents the evolution of a scalar field $u(t,x)$ following the expressions hereafter:

$$\frac{\partial u(t,x)}{\partial t} + u(t,x)\frac{\partial u(t,x)}{\partial x} = \nu \frac{\partial^2 u(t,x)}{\partial x^2} \quad in \quad (t,x) \in [0,T] \times \Omega \tag{7}$$

$$u(0,x) = u_0(x) \quad in \quad x \in \Omega \tag{8}$$

$$u(t,x) = g(t,x) \quad in \quad (t,x) \in [0,T] \times \partial\Omega \tag{9}$$

Where ν stands for a dissipation coefficient referred to as *viscosity*, u_0 is the initial condition and g is the boundary condition.
In the absence of viscosity, $\nu = 0$, the right hand side of (7) vanishes and the equation reduces to its inviscid form, which is a first order quasi-linear, hyperbolic, conservation equation that develops discontinuities or shock waves.

3.2 Surrogate model for 1-D viscous Burgers equation

We aim at building a surrogate model for the solution of the viscous Burgers equation using machine learning-based techniques previously discussed. The key objective is to develop a QC algorithm that solves the regression problem presented in 2.1, for instance a quantum neural network (a quantum-hybrid neural network might also be envisaged) and its associated training algorithm. For this purpose, the first step consists in reformulating the mathematical problem described in this section as a machine learning problem. Then, a QC algorithm shall be developed to address it.

In order to check the quality of the algorithm, we focus on an initial value problem for which an analytic solution exists so that the numerical solution can be validated straightforward.

Consider the equation (7) in the domain $(t,x) \in [0,T] \times [0,2\pi]$ along with the following initial condition and periodic boundary conditions:

$$u(0,x) = -2\frac{\nu}{\phi(0,x)}\frac{d\phi}{dx} + 4 \quad in \quad x \in [0,2\pi] \tag{10}$$

$$u(t,0) = u(t,2\pi) \quad in \quad t \in [0,T] \tag{11}$$

Under these conditions, Burgers equation presents an analytic solution that reads:

$$u(t,x) = -2\frac{\nu}{\phi(t,x)}\frac{d\phi}{dx} + 4 \quad in \quad (t,x) \in [0,T] \times [0,2\pi] \tag{12}$$

Where:

$$\phi(t,x) = exp(\frac{-(x-4t)^2}{4\nu(t+1)}) + exp(\frac{-(x-4t-2\pi)^2}{4\nu(t+1)}) \quad (13)$$

We consider here the viscosity to be $\nu = 0.05$.
A reference surrogate model of this PDE was generated by means of implementing the algorithm discussed in section 2.1 on CPU hardware resources. The Tensorflow machine learning library [2] was used to train a 6-layer neural network containing 10 neurons each. Training con-vergence was reached after 2,000 epochs. At each epoch, the domain $(t,x) \epsilon [0,0.5] \times [0,2\pi]$ was randomly sampled to obtain 20,000 training instances. Turnaround time for the whole training process was around 10 minutes using a standard laptop PC.

Figure 2 displays the comparison of the surrogate model solution against the analytic solution for evenly spaced time instants ranging from $t = 0\ to\ t = 0.5$. It can be appreciated that this simple neural network allows reproducing the PDE equation with fair accuracy. However, the time spent in the model generation remains large compared to a FDM approach that would take few seconds.

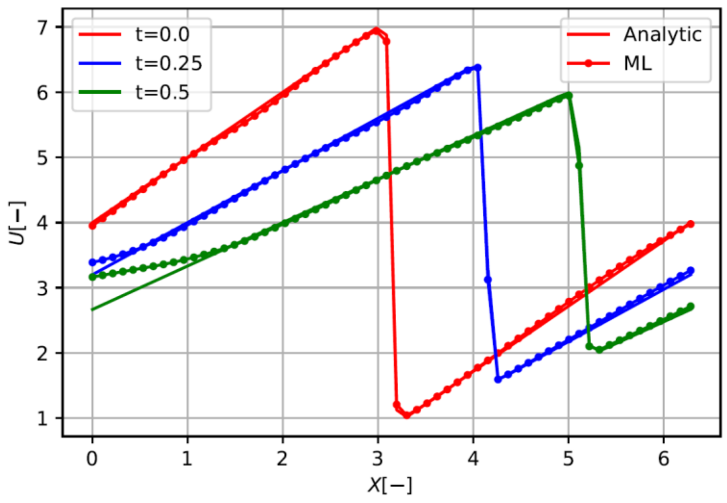

Figure 2: Comparison of the analytic and surrogate model solution of the 1-D Burgers viscous equation

3.3 Surrogate model for 1-D inviscid Burgers equation

This part aims at applying the developed QC algorithms for building a surrogate model of the inviscid version of the Burgers equation. In the absence of viscosity, the Burgers equation becomes a conservation equation that can be written as:

$$\frac{\partial u}{\partial t}(t,x) + \frac{\partial}{\partial x}(F\ (u)) \ = \ 0\ in\ (t,x)\ \epsilon\ [0,T\ ]\ \times\ \Omega\ with\ F\ (u)\ = \frac{1}{2}u^2(t,x) \quad (14)$$

Hereafter we consider the following boundary and initial conditions:

$$u(0,x) = exp(-\frac{(x-b)^2}{2c^2}) \ in \ x \ \epsilon \ [0,1] \tag{15}$$

$$u(0,0) = u(0,1) = 0 \tag{16}$$

The initial condition is then a gaussian wave with parameters $b = 0.5$ and $c = 0.1$ and Dirichlet boundary conditions are considered. Under these conditions, the solution develops a discontinuity.

The same approach as for the viscous version of the PDE was applied to this case to generate a surrogate model. A 6-layer neural network containing 10 neurons each was trained for 2,000 epochs by means of randomly sampling the domain $(t, x) \, \epsilon \, [0,0.5] \times [0,1]$ to obtain 20,000 training instances. Turnaround time for the whole training process was around 10 minutes using a standard laptop PC.

Figure 3 displays the comparison of the surrogate model solution against a FDM-obtained solution using the Lax-Friedrichs numerical scheme (see equation 17) for some time instants between $t = 0$ to $t = 0.5$, featuring the development of a shock wave. It can be appreciated that this simple neural network allows reproducing the PDE equation with fair accuracy. However, the time spent in the model generation remains large compared to the FDM solution that took only few seconds.

$$u_i^{n+1} = \frac{1}{2}(u_{i+1}^n - u_{i-1}^n) - \frac{\Delta t}{2\Delta x}(F(u_{i+1}^n) - F(u_{i-1}^n)) \tag{17}$$

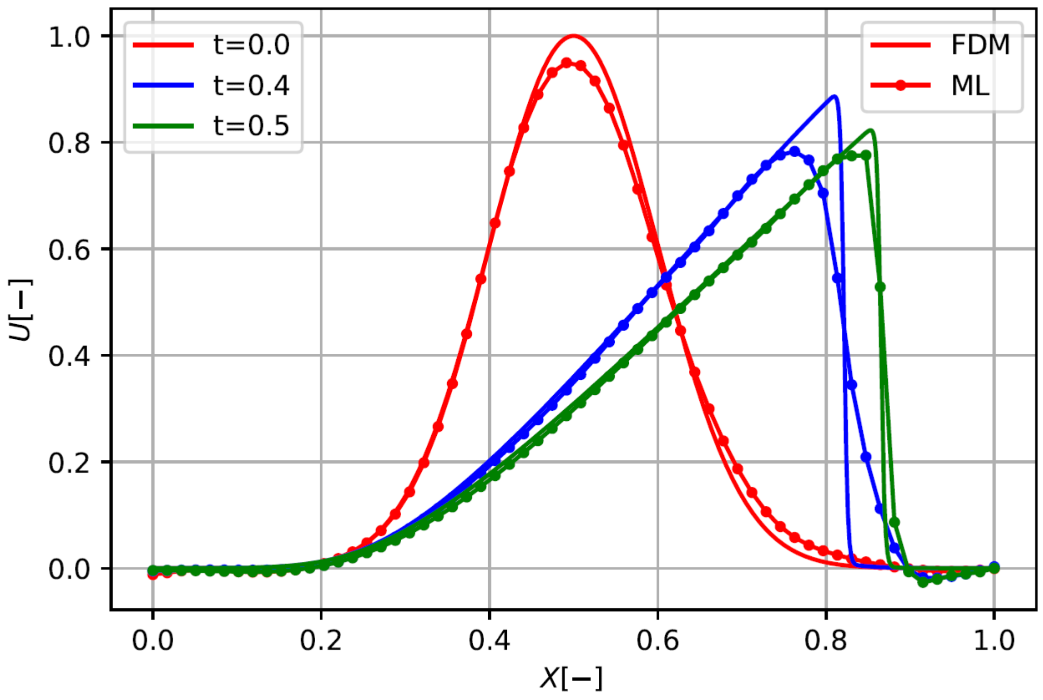

Figure 3: Comparison of the FDM and surrogate model solution of the 1-D Burgers inviscid equation

4. KPI

The assessment of the proposed algorithms will be based on the following criteria:

1. **Problem decomposition.** The choice of the approach to build a neural network and the associated training method using QC algorithms (for instance a Quantum Neural Network or a quantum-hybrid approach).
2. **Algorithm mapping.** The algorithm complexity (namely the size of the neural network) that can be fit in given QC hardware resources (number of qubits)
3. **Algorithm scalability.** The evolution of the computational time required to build more complex surrogate models (i.e., larger neural networks).

The algorithm that provides the best performances based on these criteria will be run in QC hardware and demonstrated in the generation of surrogate models of the viscous and inviscid versions of the Burgers equation.

References

[1] Sirignano, J. and Spiliopoulos, K., *DGM: A deep learning algorithm for solving partial differential equations*, Journal of Computational Physics 2018.

[2] Abadi, M. et al., *TensorFlow: A System for Large-Scale Machine Learning*, 2016.

[3] Hertz, J. et al, *Introduction to the theory of neural computation*, 1991.