CentraleSupélec

université PARIS-SACLAY

Big Data Management & Analytics Master

Differential Equations Solver using NFTM for Burger Equations

Samuel CHAPUIS
samuel.chapuis@student-cs.fr

Lucia Victoria FERNANDEZ SANCHEZ
lucia-victoria.fernandez@student-cs.fr

Alexandra PERRUCHOT-TRIBOULET RODRIGUEZ
alexandra.perruchot-triboulet-rodriguez@student-cs.fr

Github Link

Advisor: Nacéra SEGHOUANI - Nacera.Seghouani@centralesupelec.fr
Advisor: Akash MALHOTRA - akash.malhotra@centralesupelec.fr

Contents

Chapter 1

Introduction

1.1 General Context: Partial Differential Equations in Science and Engineering

Partial differential equations (PDEs) constitute the fundamental mathematical framework for describing continuous phenomena across virtually all domains of science and engineering. From the quantum mechanics governing subatomic particles to the fluid dynamics shaping atmospheric patterns, PDEs provide the rigorous language through which we understand, predict, and control the physical world.

1.1.1 Mathematical Framework of PDEs

A partial differential equation is a relationship involving an unknown function u of multiple independent variables and its partial derivatives. The general form of a PDE can be expressed as:

$$F\left(x_1, x_2, \ldots, x_n, t, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \ldots, \frac{\partial^2 u}{\partial x_1^2}, \ldots, \frac{\partial^k u}{\partial x_1^k \partial x_2^l \cdots}\right) = 0 \tag{1.1}$$

where:

- $u = u(x_1, x_2, \ldots, x_n, t)$ is the unknown field variable
- $x_1, x_2, \ldots, x_n$ are spatial coordinates
- t is time
- F is a functional relationship
- $k, l, \ldots$ denote the order of partial derivatives

Classification by Order. The order of a PDE is determined by the highest derivative present. For a second-order PDE in two dimensions:

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu + G = 0 \tag{1.2}$$

The discriminant $\Delta = B^2 - 4AC$ determines the PDE type:

- **Elliptic** ($\Delta < 0$): Steady-state problems (Laplace, Poisson equations)
- **Parabolic** ($\Delta = 0$): Diffusion processes (Heat equation)
- **Hyperbolic** ($\Delta > 0$): Wave propagation (Wave equation)

1.1.2 Fundamental PDEs in Physics

1. Conservation of Mass (Continuity Equation). The principle that mass cannot be created or destroyed leads to:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{1.3}$$

where $\rho(x, y, z, t)$ is density and $\mathbf{u}(x, y, z, t)$ is the velocity field.

2. Conservation of Momentum (Navier-Stokes Equations). Newton's second law applied to a fluid element yields:

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} \right) = -\nabla p + \mu \nabla^2 \mathbf{u} + \rho \mathbf{g} \tag{1.4}$$

where:

- $p(x, y, z, t)$ is pressure [Pa]
- μ is dynamic viscosity [Pa·s]
- $\mathbf{g}$ is gravitational acceleration [m·s^{-2}]

3. Conservation of Energy (Heat Equation). Fourier's law of heat conduction combined with energy balance gives:

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + Q \tag{1.5}$$

where:

- $T(x, y, z, t)$ is temperature [K]
- c_p is specific heat capacity [J·kg^{-1}·K^{-1}]
- k is thermal conductivity [W·m^{-1}·K^{-1}]
- Q is heat source term [W·m^{-3}]

4. Wave Propagation. The wave equation governing vibrations, acoustics, and electromagnetic radiation:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u \tag{1.6}$$

where c is the wave speed [m·s^{-1}].

1.1.3 Challenges in Classical PDE Solving

Traditional numerical methods for solving PDEs face several fundamental limitations:

Computational Complexity. For a PDE discretized on an N-dimensional grid with M points per dimension and T time steps, the computational cost scales as:

$$\text{Cost} \sim \mathcal{O}(M^N \cdot T \cdot K) \tag{1.7}$$

where K is the cost per grid point update. For 3D problems with $M = 256$, this becomes prohibitive.

Curse of Dimensionality. The number of grid points grows exponentially with dimension:

$$\text{Grid points} = M^N \tag{1.8}$$

For $N = 3$ dimensions and $M = 100$ points per dimension: 10^6 grid points.

Stability Constraints. Explicit time-stepping schemes require timestep Δt to satisfy the CFL (Courant-Friedrichs-Lewy) condition:

$$\Delta t \leqslant C \cdot \frac{\Delta x}{|\mathbf{u}_{\max}|} \tag{1.9}$$

where $C \leqslant 1$ for stability. For fine grids ($\Delta x \to 0$), this forces $\Delta t \to 0$, dramatically increasing computational cost.

Stiffness. Many physical systems exhibit stiffness, where multiple timescales coexist. The stiffness ratio:

$$S = \frac{\lambda_{\max}}{\lambda_{\min}} \tag{1.10}$$

where λ are eigenvalues of the system operator. For $S \gg 1$, explicit methods require $\Delta t \sim 1/\lambda_{\max}$ even when dynamics of interest occur at timescale $\sim 1/\lambda_{\min}$.

1.2 The Promise of Neural PDE Solvers

1.2.1 Mathematical Framework for Learning Solution Operators

Rather than solving individual PDE instances, neural approaches learn **operators** that map between function spaces.

Operator Learning Formulation. Given a PDE of the form:

$$\mathcal{L}[u] = f \tag{1.11}$$

where $\mathcal{L}$ is a differential operator, we seek to learn the solution operator:

$$\mathcal{G} : f \mapsto u \tag{1.12}$$

such that $u = \mathcal{G}(f)$ satisfies $\mathcal{L}[u] = f$.

Universal Approximation for Operators. The neural operator framework [3] proves that certain neural architectures can approximate continuous operators between Banach spaces to arbitrary accuracy:

$$\|\mathcal{G} - \mathcal{G}_\theta\|_{L^p(\mathcal{X},\mathcal{Y})} < \varepsilon \tag{1.13}$$

where:

- $\mathcal{G}_\theta$ is the neural operator with parameters θ
- $\mathcal{X}$ is the input function space
- $\mathcal{Y}$ is the output function space
- ε is the approximation error

1.2.2 Three Paradigms for Neural PDE Solving

Paradigm 1: Physics-Informed Neural Networks (PINNs). PINNs [5] approximate the solution $u(x,t)$ directly with a neural network $u_\theta(x,t)$ by minimizing:

$$\mathcal{L}_{\text{PINN}} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{IC}} + \mathcal{L}_{\text{BC}} \tag{1.14}$$

where:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_{\text{col}}} \sum_{i=1}^{N_{\text{col}}} \left|\mathcal{L}[u_\theta](x_i,t_i) - f(x_i,t_i)\right|^2 \tag{1.15}$$

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_{\text{IC}}} \sum_{j=1}^{N_{\text{IC}}} \left|u_\theta(x_j,t_0) - u_0(x_j)\right|^2 \tag{1.16}$$

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_{\text{BC}}} \sum_{k=1}^{N_{\text{BC}}} \left|u_\theta(x_{\partial\Omega},t_k) - g(x_{\partial\Omega},t_k)\right|^2 \tag{1.17}$$

Automatic Differentiation for PDE Residuals. The PDE residual is computed via automatic differentiation:

$$\frac{\partial u_\theta}{\partial t} = \frac{\partial}{\partial t}\left(\text{NeuralNet}_\theta(x,t)\right) \tag{1.18}$$

computed exactly using the chain rule through the computational graph.

Paradigm 2: Fourier Neural Operators (FNOs). FNOs [4] learn in the Fourier domain, leveraging the convolution theorem. The operator layer is:

$$v_{\ell+1}(x) = \sigma\left(W_\ell v_\ell(x) + \mathcal{F}^{-1}\left(R_\ell \cdot \mathcal{F}(v_\ell)\right)(x)\right) \tag{1.19}$$

where:

- $\mathcal{F}$ denotes Fourier transform: $\mathcal{F}(v)(k) = \int v(x)e^{-2\pi ik\cdot x}dx$
- $R_\ell(k)$ is a learnable weight in Fourier space
- W_ℓ is a local linear transformation
- σ is a nonlinear activation

Resolution Invariance. FNOs satisfy:

$$\mathcal{G}_\theta(v)|_M \approx \mathcal{G}_\theta(v|_M) \tag{1.20}$$

meaning the operator learned at resolution M generalizes to different resolutions.

Paradigm 3: Neural Field Turing Machines (NFTMs). NFTMs employ an iterative refinement strategy inspired by classical solvers:

$$u^{n+1}(x) = u^n(x) + \text{Controller}_\theta\left(\text{Patch}(u^n, x)\right) \tag{1.21}$$

where:

- u^n is the field at iteration n
- $\text{Patch}(u^n, x)$ extracts a local neighborhood around x
- Controller_θ is a neural network applying learned update rules

This formulation mimics iterative methods like Jacobi or Gauss-Seidel.

Classical Jacobi Iteration:

$$u_i^{n+1} = \frac{1}{2a}\left(f_i - bu_{i-1}^n - cu_{i+1}^n\right) \tag{1.22}$$

NFTM Analog:

$$u_i^{n+1} = u_i^n + \text{NN}_\theta\left([u_{i-k}^n, \ldots, u_i^n, \ldots, u_{i+k}^n]\right) \tag{1.23}$$

1.3 The Problem: Burgers Equation as a Test Case

1.3.1 Mathematical Formulation

The one-dimensional Burgers equation is a nonlinear PDE that serves as the simplest model capturing the essence of fluid dynamics:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{1.24}$$

with:

- Domain: $x \in [0, L]$, $t \in [0, T]$
- Initial condition: $u(x, 0) = u_0(x)$
- Boundary conditions: $u(0, t) = u_L(t)$, $u(L, t) = u_R(t)$
- Kinematic viscosity: $\nu > 0$ $[\text{m}^2 \cdot \text{s}^{-1}]$

1.3.2 Physical Interpretation

Advection Term ($u\frac{\partial u}{\partial x}$). This nonlinear term represents self-advection or transport. A fluid element at position x with velocity $u(x,t)$ carries itself downstream. The nonlinearity leads to **wave steepening**.

Consider a smooth initial profile $u_0(x) = A\sin(kx)$ with $k = 2\pi/L$.

The characteristic curves along which information propagates satisfy:

$$\frac{dx}{dt} = u(x,t) \tag{1.25}$$

Since velocity varies spatially, different parts of the wave travel at different speeds, causing steepening and eventual shock formation.

Diffusion Term ($\nu\frac{\partial^2 u}{\partial x^2}$). This linear term represents viscous dissipation, smoothing gradients:

$$\frac{\partial u}{\partial t} = \nu\frac{\partial^2 u}{\partial x^2} \tag{1.26}$$

has the fundamental solution (Green's function):

$$G(x,t) = \frac{1}{\sqrt{4\pi\nu t}}\exp\left(-\frac{x^2}{4\nu t}\right) \tag{1.27}$$

showing that initially sharp features spread with width $\sigma \sim \sqrt{\nu t}$.

1.3.3 Dimensionless Analysis and Reynolds Number

Introduce characteristic scales:

- Length: L [m]
- Velocity: U [m·s^{-1}]
- Time: $T = L/U$ [s]

Define dimensionless variables:

$$x^* = \frac{x}{L}, \quad t^* = \frac{tU}{L}, \quad u^* = \frac{u}{U} \tag{1.28}$$

Substituting into Burgers equation:

$$\frac{U}{T}\frac{\partial u^*}{\partial t^*} + \frac{U^2}{L}u^*\frac{\partial u^*}{\partial x^*} = \frac{\nu U}{L^2}\frac{\partial^2 u^*}{\partial x^{*2}} \tag{1.29}$$

Dividing by U^2/L:

$$\frac{\partial u^*}{\partial t^*} + u^*\frac{\partial u^*}{\partial x^*} = \frac{1}{\mathrm{Re}}\frac{\partial^2 u^*}{\partial x^{*2}} \tag{1.30}$$

where the **Reynolds number** is:

$$\mathrm{Re} = \frac{UL}{\nu} = \frac{\text{inertial forces}}{\text{viscous forces}} \tag{1.31}$$

Physical Regimes.

1. **Low Reynolds Number** (Re $\ll 1$, ν large):
   - Viscosity dominates
   - Smooth, diffusion-controlled solutions
   - No shock formation
   - Analytical solutions via similarity methods
2. **Moderate Reynolds Number** (Re ~ 1):
   - Balanced advection and diffusion
   - Weak shocks with finite width
   - Transition regime
3. **High Reynolds Number** (Re $\gg 1$, ν small):
   - Inertia dominates
   - Sharp shocks and discontinuities
   - Asymptotically approaches inviscid limit
   - Highly challenging for numerical methods

1.3.4 Shock Formation and the Inviscid Limit

For the inviscid Burgers equation ($\nu = 0$):

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \tag{1.32}$$

the method of characteristics gives:

$$\frac{dx}{dt} = u, \quad \frac{du}{dt} = 0 \tag{1.33}$$

implying u is constant along characteristics: $u = u_0(x_0)$ where x_0 is the initial position.

Shock Formation Time. Consider initial condition $u_0(x) = -\alpha x$ with $\alpha > 0$. Characteristics satisfy:

$$x(t) = x_0 + u_0(x_0)t = x_0 - \alpha x_0 t = x_0(1 - \alpha t) \tag{1.34}$$

Characteristics originating from different x_0 intersect when:

$$t_{\text{shock}} = \frac{1}{\max\left(-\frac{du_0}{dx}\right)} = \frac{1}{\alpha} \tag{1.35}$$

At this time, u becomes multi-valued (unphysical), and a shock discontinuity forms.

Rankine-Hugoniot Jump Condition. Across a shock at position $s(t)$, conservation laws require:

$$\frac{ds}{dt} = \frac{1}{2}(u_L + u_R) \tag{1.36}$$

where $u_L = u(s^-, t)$ and $u_R = u(s^+, t)$ are velocities on left and right sides.

For viscous Burgers with small ν, shocks have finite width:

$$\delta_{\text{shock}} \sim \frac{\nu}{u_L - u_R} \tag{1.37}$$

This rapid variation challenges numerical methods and provides an ideal test for neural solvers.

1.3.5 Cole-Hopf Transformation and Exact Solutions

The Cole-Hopf transformation [1, 2] linearizes Burgers equation:

Let:

$$u(x,t) = -2\nu \frac{\partial}{\partial x} \ln \phi(x,t) \tag{1.38}$$

Substituting into Burgers equation yields the **heat equation**:

$$\frac{\partial \phi}{\partial t} = \nu \frac{\partial^2 \phi}{\partial x^2} \tag{1.39}$$

Solution Procedure.

1. Given $u_0(x)$, compute $\phi_0(x) = \exp\left(-\frac{1}{2\nu}\int_0^x u_0(\xi)d\xi\right)$
2. Solve heat equation: $\phi(x,t) = \int_{-\infty}^{\infty} G(x-\xi,t)\phi_0(\xi)d\xi$
3. Recover $u(x,t) = -2\nu\frac{\partial \ln \phi}{\partial x}$

Example: Gaussian Initial Condition. For $u_0(x) = A\exp\left(-\frac{x^2}{2\sigma^2}\right)$, the exact solution is:

$$u(x,t) = \frac{Ax}{1 + \frac{At}{2\nu}\exp\left(-\frac{x^2}{2\sigma^2(1+\frac{2\nu t}{\sigma^2})}\right)} \tag{1.40}$$

This provides ground truth for validation of neural solvers.

1.3.6 Why Burgers Equation for Neural PDE Research?

The Burgers equation is ideal for benchmarking neural PDE solvers because it:

1. **Captures essential physics:**
   - Nonlinearity (wave steepening)
   - Dissipation (viscous damping)
   - Shock formation
   - Parameter dependence (Reynolds number)

2. **Has analytical solutions:** Via Cole-Hopf, enabling exact validation
3. **Computationally tractable:** 1D allows rapid iteration
4. **Generalizes to Navier-Stokes:** The velocity components in 2D/3D Navier-Stokes satisfy coupled Burgers-like equations
5. **Well-studied benchmark:** Extensive literature for comparison (PINNs, FNOs, classical methods)

1.4 Challenges in Neural PDE Solving

Despite the promise of neural approaches, several fundamental challenges persist.

1.4.1 Accuracy vs. Computational Efficiency Trade-off

Quantitative Analysis. Traditional finite difference methods achieve accuracy:

$$\|u_{\text{numerical}} - u_{\text{exact}}\|_{L^2} = \mathcal{O}(\Delta x^p + \Delta t^q) \tag{1.41}$$

where p, q are the spatial and temporal orders of accuracy (typically 2-4).

For $\Delta x = 10^{-3}$, second-order method: error $\sim 10^{-6}$.

Neural methods currently achieve:

$$\|u_{\text{neural}} - u_{\text{exact}}\|_{L^2} \sim 10^{-3} \text{ to } 10^{-5} \tag{1.42}$$

Computational Cost Comparison. Finite Difference for Burgers 1D:

- Grid points: N
- Time steps: T
- Operations per step: $\mathcal{O}(N)$
- Total cost: $\mathcal{O}(NT)$

For $N = 256$, $T = 10^4$ steps: $\sim 2.5 \times 10^6$ operations

Neural Forward Pass:

- Layer evaluations: L layers
- Operations per layer: $\mathcal{O}(D^2)$ where D is hidden dimension
- Total cost: $\mathcal{O}(LD^2)$

For $L = 5$, $D = 64$: $\sim 2 \times 10^4$ operations per prediction

Speedup Potential. Once trained, neural solver can be 100× faster per forward pass, but with 1-2 orders of magnitude lower accuracy.

1.4.2 Generalization to Unseen Parameter Regimes

Statistical Learning Perspective. Given training set $\mathcal{D} = \{(\nu_i, u_0^{(i)}, u_{\text{target}}^{(i)})\}_{i=1}^N$, we learn mapping:

$$\mathcal{G}_\theta : (\nu, u_0) \mapsto u(t) \tag{1.43}$$

Generalization Error.

$$\mathbb{E}_{(\nu,u_0)\sim p_{\text{test}}} \left[\|u_{\text{pred}} - u_{\text{true}}\|^2 \right] \tag{1.44}$$

For parameter extrapolation ($\nu_{\text{test}} \notin [\nu_{\min}^{\text{train}}, \nu_{\max}^{\text{train}}]$), generalization degrades.
Empirical observations show:

$$\text{Error}(\nu) \approx \text{Error}_{\text{train}} \cdot \left(1 + \alpha \cdot \frac{|\nu - \nu_{\text{train}}|}{\nu_{\text{train}}} \right)^\beta \tag{1.45}$$

with $\beta \approx 2 - 3$ for most neural solvers, indicating rapid degradation outside training distribution.

1.4.3 Long-Time Stability and Error Accumulation

Autoregressive Rollout. For time-dependent PDEs, neural solvers predict iteratively:

$$u^{n+1} = \mathcal{G}_\theta(u^n) \tag{1.46}$$

Error Propagation Analysis. Let $\varepsilon^n = u_{\text{pred}}^n - u_{\text{true}}^n$ be the error at step n.
Linearizing around true trajectory:

$$\varepsilon^{n+1} \approx J\varepsilon^n + \eta^n \tag{1.47}$$

where $J = \frac{\partial \mathcal{G}_\theta}{\partial u}\Big|_{u_{\text{true}}^n}$ is the Jacobian and η^n is per-step error.

Spectral Stability. If eigenvalues of J satisfy $|\lambda_i| \geqslant 1$, errors grow exponentially:

$$\|\varepsilon^n\| \sim \|\varepsilon^0\| \cdot \lambda_{\max}^n + \frac{\|\eta\|}{1 - \lambda_{\max}} \tag{1.48}$$

For $\lambda_{\max} = 1.01$ and $n = 1000$ steps:

$$\text{Error growth} \sim (1.01)^{1000} \approx 20,000\times \tag{1.49}$$

Current State-of-the-Art. Most neural solvers diverge after:

$$n_{\text{stable}} \approx 10 - 100 \times n_{\text{train}} \tag{1.50}$$

This limits practical applicability to short-horizon predictions.

1.4.4 Physical Consistency and Conservation Laws

Conservation Law Violation. Consider mass conservation for Burgers:

$$M(t) = \int_0^L u(x,t)dx \tag{1.51}$$

For periodic BCs, exact evolution satisfies:

$$\frac{dM}{dt} = -\nu \left[\frac{\partial u}{\partial x}\right]_0^L = 0 \tag{1.52}$$

Neural solvers typically violate this:

$$\left|\frac{M^{\text{pred}}(t) - M(0)}{M(0)}\right| \sim 10^{-2} \text{ to } 10^{-1} \tag{1.53}$$

after long rollouts.

Energy Dissipation. For Burgers, kinetic energy:

$$E(t) = \frac{1}{2}\int_0^L u^2(x,t)dx \tag{1.54}$$

satisfies:

$$\frac{dE}{dt} = -\nu \int_0^L \left(\frac{\partial u}{\partial x}\right)^2 dx \leqslant 0 \tag{1.55}$$

Neural solvers often exhibit energy drift, with dE/dt fluctuating in sign, violating second law of thermodynamics.

1.5 Project Objectives

This thesis investigates the application of **Neural Field Turing Machines (NFTMs)** to solve the one-dimensional Burgers equation, with a focus on developing architectural innovations that address the challenges outlined above.

1.5.1 Primary Research Questions

RQ1: Architecture Design. Can a hypernetwork-based NFTM controller, conditioned on problem parameters, achieve accuracy competitive with specialized neural operators while maintaining computational efficiency?

Mathematical Formulation. Design $\text{Controller}_\theta : (\text{Patch}(u^n), \nu) \mapsto \Delta u$ where:

- Hypernetwork generates weights: $\theta = h_\phi(\nu)$
- Multi-scale processing: kernels $k \in \{3, 7, 15\}$
- Target: $\|u_{\text{NFTM}} - u_{\text{FD}}\|_{L^2} < 10^{-4}$

RQ2: Parameter Generalization. Does conditioning on viscosity ν enable extrapolation to unseen parameter values, and what are the quantitative limits of this generalization?

Experimental Protocol.

- Train: $\nu \in [0.01, 0.05]$
- Test: $\nu \in \{0.005, 0.075, 0.10\}$
- Measure: $\text{Error}(\nu_{\text{test}})/\text{Error}(\nu_{\text{train}})$

RQ3: Long-Time Stability. Can architectural choices (residual connections, normalization, multi-scale kernels) improve rollout stability beyond current state-of-the-art?

Stability Metric.

$$T_{\text{stable}} = \max\{t : \|u_{\text{pred}}(t) - u_{\text{true}}(t)\|_{L^2} < \varepsilon_{\text{threshold}}\} \tag{1.56}$$

Target: $T_{\text{stable}} > 50 \times T_{\text{train}}$

RQ4: Computational Efficiency. What is the accuracy-speed Pareto frontier for NFTM compared to classical solvers and alternative neural methods?

Metrics.

- Forward pass time (ms)
- Memory usage (GB)
- Accuracy (L^2 error)
- Amortized cost for parameter sweeps

Chapter 2

Related Work

It is expected to find:

- The research articles related to your defined problem/objective
- Try to present them by category, the main ideas behind and their limitations
- Justify your directions/choices. This part should make a link with the next chapter.

Chapter 3

Background

The objective here is to detail the main concepts/definitions existing algorithms needed to understand your work to be detailed later and to introduce the notations to be used.

Use examples

Don't forget to cite again these existing approaches

Chapter 4

Our Methodology and Approach

- First the overall methodology using a figure: different components
- Explain in detail each component: it's based on the previous chapter, algorithms if an existing algorithm is extended,
- use examples
- no code but pseudo-code
- the link to your Github must appear in first title page (as a footnote). make sure that the supervisors and me have the access. A text file called readme.text must explain ho to run your program, using your dataset

Chapter 5

Experiments and Evaluation

- Objective of Experiments, which measures, which comparisons, evaluations, according to which parameters
- Data description
- Overall program using a figure (API ???) make the link with the components/parts explained in the previous chapter
- no code
- Results/interpretation, each table/curve must be explained in the text

Chapter 6

Conclusion and Perspectives

A summary of your work. More focused on the results

The limitations of the work -> which perspectives/clues to deal with limitations, to improve your work

the last paragraph must be dedicated to the work in team

6.1 Gl remarks

GENERAL :

Each table, figure must be cited and explained in the text.

The references must be complete

Each chapter must start with a paragraph to introduce its content (no need to have a separated for that), except the introduction and the conclusion. In the same manner each chapter must finish with a paragraph to conclude and to make a link with the next one, except the introduction and the conclusion.

6.2 Gl remarks about the presentation

The slides must be numbered

The presentation follows more and less the structure of the report

No too much blabla about the the gl context you need to define the objectives of the project (with examples if possible) ...

Then how your work fits into existing works (some main related works), the overall pipeline, your main contributions in this pipeline

also your main contributions in terms of implementation

the main results

Conclusions and next ...

Then the overall

Appendix A

Appendix

The progress draft must be included in the appendix

Bibliography

[1] Julian D. Cole. On a quasi-linear parabolic equation occurring in aerodynamics. Quarterly of Applied Mathematics, 9:225–236, 1951. (Cited on page 8.)

[2] Eberhard Hopf. The partial differential equation $u_t + uu_x = \mu u_{xx}$. Communications on Pure and Applied Mathematics, 3:201–230, 1950. (Cited on page 8.)

[3] Nikola B. Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. Journal of Machine Learning Research, 24(89):1–97, 2023. (Cited on page 4.)

[4] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In Proceedings of the International Conference on Learning Representations (ICLR), 2021. (Cited on page 4.)

[5] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378:686–707, 2019. (Cited on page 4.)

Abstract:

Keywords: