Simulation and Modeling of Traffic Congestion.

Pedro G. Branquinho, Wei-Liang Qian (钱卫良)

17 December 2021

Contents

1 Abstract

We reproduced numerical method algorithms in this research. The initial aim was at studying partial differential equations and to recreate the simulations of a well known work on *Traffic Flow* (1). Although the end goal was not achieved, the research provided knowledge on the topics of *Numerical Stiffness* and the limitations of *Physics Informed Neural Network* (PINN).

2 Introduction

Throughout the research, we used a `Version Control System` to keep a backup and to track the progress; git and GitHub. The language chosen to model and compute the PDEs was `Julia` 3.3.

The numerical methods explored consisted of Numerical Analysis (von Neumman) (2), Forward methods, and `Physics Informed Neural Network` (PINN) (3).

When the PINN method wasn't sufficient to solve the problem, a study about `Numerical Stiffness` and the analytical numerical analysis became the focus of the study. At this point, one of the programs developed dialed with simulating `Stability Regions` of numerical methods [1].

The equations we intended to simulate are a `non-linear system` 2.

2.1 Analytical standpoint

From a *Analytical* point of view, Partial Differential Equations (PDEs) differs from Ordinary Differential Equations (ODEs) in the number of free variables. This means, in the case of PDEs the model depends on a relationship of many variables and ODEs of only one.

2.1.1 Definition

A `partial differential equation` means a relation, for a given function `u(x,y,...)` (4),

$$F(x, y, \dots, u, u_x, u_y, \dots, u_{xx}, u_{xy}, \dots) = 0 \tag{1}$$

[1] the program was based on <https://github.com/jverzani/ImplicitEquations.jl>

If more than one partial differential equation is needed to describe a model, then these PDEs are called a `system`.

If substituted an $u(\mathbf{x})$ which satisfies $F(\mathbf{x}, \mathbf{u}(\mathbf{x})) = 0$. Then, `u` is a solution of the `PDE` or the `system`.

2.2 Numerical standpoint

The similarity between ODEs and PDEs, from a *Numerical* point of view, is that different approximation methods will result in different errors, in relation to the exact, analytical, solution of the equations.

The need for numerical methods also unite both types of equation. Since, even with ODEs as simple as the one derived from the *Simple Pendulum* do not have an analytical solution, without making simplifying hypothesis (5).

2.3 Methods studied

2.3.1 Ordinary Differential Equation

Euler Forward, Backward and Adams-Moulton (trapezoidal) methods have been studied with the standard test equation,

$$y'(t) = e^{-\lambda t} \tag{2}$$

This was done in other to both get used implementing numerical methods, and get a better empirical sense of how different methods affect the accuracy of the model.

Different variations of this system was simulated, in order to understand the concepts of `Stiffness`, `A-stability` and `L-stability`.

2.3.2 Nth-order methods for partial differentiation

The numerical methods were further explored using the analytical derivation of higher order numerical formulas. These derivations consisting in using Taylor Series and arrive at a formula considering `n` points to calculate a step. E.g., for the second order partial differential of the third order

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} - O(\Delta x^2) \tag{3}$$

2.4 Analytical theory on perturbations

On Kerner's paper, he uses perturbation theory to developed the analytical formulas that implies the existence of a clustering effect - traffic jams - even with extremely initial small perturbations (1).

In order to better understand these derivations, material has been studied on the topic (6), and a derivation of the spring-mass equations for a small pertubation was derived.

3 Bibliography Review

3.1 Development of the field

3.1.1 The dawn of numerical methods

Numerical analysis dawn can be dated to 1820 B.C. (7), with Egyptian's methods for calculating roots.

Although, modern methods for solving ODEs and PDEs were mainly developed after the discovery of *Calculus*, in the 17th century.

3.1.2 Modern methods and Computing

Currently, even though sophisticated mathematics has been developed to accurately simulate virtually any ODE, the mathematics for solving PDEs is still a open field. This is due to the nature of PDEs which do not have a general method or procedure that is efficient, when simulating them.

3.1.3 Physics Informed Neural Networks

There exists methods aimed at using the computational power available in computers to abstract the theoretical knowledge of Numerical Stability away from the problem. *Physics Informed Neural Network* (PINN) is one of these methods. The shortcoming of the method is the loss of the possibility of making small adjustments to the resulting *Numerical Method*, as will be discussed further on the results.

3.1.4 Numerical Instability

Each equation, and it's constant parameters, will have specific method or a coupling of methods suited to the simulation of a PDE.

3.1.5 Grouping of methods and types of PDEs

Broadly, methods are associated with the kind of partial differential equation one is studying. These equations can be either `Hyperbolic`, `Parabolic` or `Elliptic`. Also, equations can be mixed, e.g., `Mixed Parabolic-Hyperbolic`, etc.

3.2 Mathematical categorization of PDEs

3.2.1 Order of equations and systems

"The order of the system is the order of the highest derivative that occurs." (4). In which, irrespective of the free variable, we count the total number of derivatives. E.g., a sixth order equation, with mixed variables.

$$F(\mathbf{x}, \mathbf{u}(\mathbf{x})) = 0 \ \wedge \ F(\mathbf{x}) = \frac{\partial^6 \mathbf{u}}{(\partial x)^2 (\partial y)^2 (\partial z)} + \frac{\partial^3 \mathbf{u}}{(\partial x)^3} \tag{4}$$

3.2.2 Linearity, quasi-linearity and non-linearity

1. Linearity

   Linearity is defined as not having any term in the (1), such that it's a result of a multiplication of two independent terms. The independent terms being $\mathbf{x}$, $u(\mathbf{x})$ and all partial derivatives of $u(\mathbf{x})$.

   For example, the second order linear equation with constant coefficients, `Kolmogorov's equation` (8),

   $$u_t - \sum_{i,j=1}^{n} a^{ij} u_{x_i x_j} + \sum_{i}^{n} b^i u_{x_i} = 0 \tag{5}$$

   We see there is no non-linear terms, as the product $u_{x_i} * u_{x_j}$, etc.

2. Non-linear equations

The physical model we are interested deals with an equation similar to the `Navier-Stokes` equation (1). The `Navier-Stokes` equations are a nonlinear system (4) (1). E.i.,

$$\begin{cases} \mathbf{u}_t + \mathbf{u} \cdot D\mathbf{u} - \nabla \mathbf{u} = -Dp \\ \text{div}(\mathbf{u}) = 0 \end{cases} \tag{6}$$

3. Quasi-linear equations

   A special kind of equations that follow under the category of non-linear equations are the quasi-linear equations. These have the non-linear terms which are of a lesser order than the order of the equation. E.g., the `Korteweg-de Vries` equation (4),

$$u_t + cuu_x + u_{xxx} = 0 \tag{7}$$

3.2.3 Elliptic, hyperbolic and parabolic PDEs

Given the general quasi-linear equation for a function `u`,

$$au_{xx} + 2bu_{xy} + cu_{yy} = d \tag{8}$$

where a, b, c and d are of the form $f(x, y, u, u_x, u_y)$.

We can develop an analysis of how the solution would behave. *A priori*, the solution γ is contained on the xy-plane.

From this consideration, if carried an analysis on the curve itself, we ultimately arise at the condition:

$$\frac{dy}{dx} = \frac{b \pm \sqrt{b^2 - ac}}{a} \tag{9}$$

Then, if $ac - b^2 > 0$ it's `elliptic`; else, if $ac - b^2 < 0$ it's called `hyperbolic`. Finally, if $ac - b^2 = 0$ we call it `parabolic`.

These categories help understand the expected behavior of the solution. But, in nonlinear cases the PDE do not characterize the solution behavior in these categories; and in some linear cases, different regions will have different types of behaviors, regarding the `elliptic`, `hyperbolic` and `parabolic` characterization (4).

3.3 Numerical Methods implemented in Julia

There exists a variety of libraries on *numerical methods* available in modern languages. Most of them with ports of libraries written in C and FORTRAN. To list a few, from the `Julia` documentation on available methods:

- General PDE approximation methods.
- Transform methods.
- Finite difference methods.
- Finite element methods.
- Finite volume methods.
- Spectral element methods.
- Boundary element, Boundary integral methods.
- Mesh free methods and particle methods.
- Virtual element methods.
- Multi-method packages.
- Non-classical methods.
  Source: <https://github.com/JuliaPDE/SurveyofPDEPackages>

3.4 TODO PINN

Physics Informed Neural Networks

4 Materials and Methods

The materials used were:

- Git
- GitHub
- Julia language

4.1 **TODO** Version Control

4.2 **TODO** Julia language

4.3 **TODO** PINNs

4.4 **TODO** Developed code

4.5 **TODO** Stability Regions

5 Results and Discussion

5.1 **TODO** Kerner's reproduction through PINNs

Figure 1: Reproduction try, using PINN. Source: The authors

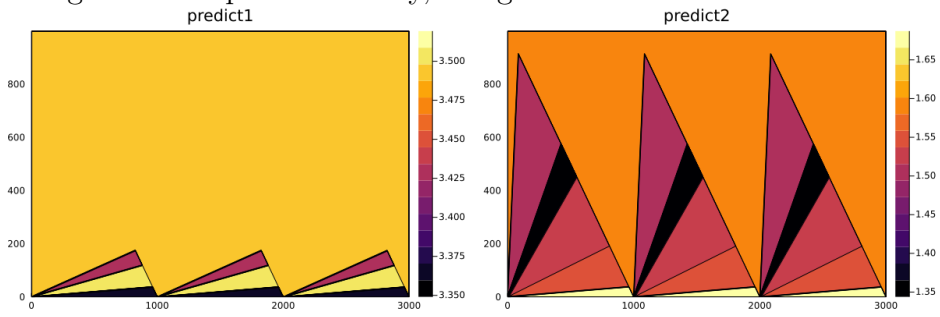
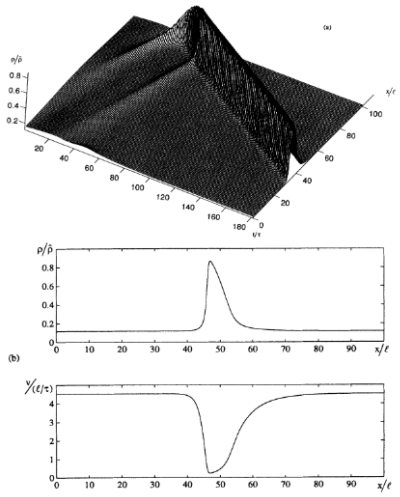


Figure 2: Original simulation. Source: Image from Kerner and Konhäuser (1)



5.2 TODO Steps to derive the Stability Region

5.3 TODO Stiffness

5.4 Nth-order approximation

5.4.1 Third-order approximation of second-order differential equation

By Taylor Expansion

$$\begin{cases} u_{i+1} &= u_i + \Delta x \left.\frac{\partial u}{\partial x}\right|_i + \frac{\Delta x^2}{2!}\frac{\partial^2 u}{\partial x^2} + \dots \\ u_{i-1} &= u_i - \Delta x \left.\frac{\partial u}{\partial x}\right|_i + \frac{\Delta x^2}{2!}\frac{\partial^2 u}{\partial x^2} + \dots \end{cases}$$
$$\sim \begin{cases} u_{i+1} &= u_i + \sum_{n=1}^{M} \left.\frac{(\Delta x)^n}{n!}\frac{\partial^n u(x)}{\partial x^n}\right|_i \\ u_{i-1} &= u_i + \sum_{n=1}^{M} (-1)^n \left.\frac{(\Delta x)^n}{n!}\frac{\partial^n u(x)}{\partial x^n}\right|_i \end{cases} \tag{10}$$

Summing both terms and isolating $\left.\frac{\partial^2 u}{\partial x^2}\right|_i$, we have:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} - O(\Delta x^2) \tag{11}$$

6 Conclusion

The use of PINNs got us closer to reproducing Kerner's results. But, at the same time, the pigtail of using a method that hide us the ability of adjusting the discretization showed itself, once the results were not satisfactory.

After getting stuck on this riddle, we took the path of learning more about *classical methods* on numerical methods for PDEs. Knowledge on the subject of *equation Stiffness* was gained in the process e.g., why and how to categorize methods and to grasp what does it mean to an equation to be stiff.

References

1 KERNER, B. S.; KONHäUSER, P. Cluster effect in initially homogeneous traffic flow. *Physical Review E*, American Physical Society (APS), v. 48, n. 4, p. R2335–R2338, Oct 1993. ISSN 1095-3787. Disponível em: <http://dx.doi.org/10.1103/physreve.48.r2335>.

2 PRESS, W. H. et al. *Numerical recipes*. [S.l.]: Cambridge university press Cambridge, 1986. v. 818.

3 ZUBOV, K. et al. Neuralpde: Automating physics-informed neural networks (pinns) with error approximations. *arXiv preprint arXiv:2107.09443*, 2021.

4 JOHN, F. Partial differential equations. *Applied mathematical sciences*, v. 1, n. 1, 1978.

5 BRAUER, F.; NOHEL, J. A. *The qualitative theory of ordinary differential equations: an introduction*. [S.l.]: Courier Corporation, 1989.

6 TREMBLAY, A.-M. Phy-892 the many-body problem, from perturbation theory to dynamical-mean field theory (lecture notes). 2017.

7 SMITH, D. E. The rhind papyrus. *Bulletin of the American Mathematical Society*, American Mathematical Society, v. 36, n. 3, p. 166–170, 1930.

8 EVANS, L. C. Partial differential equations. *Graduate studies in mathematics*, Rhode Island, USA, v. 19, n. 4, p. 7, 1998.