Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XL, págs. 1513-1525 (artículo completo)
F.A. Avid, L.C. Bessone, P. Gamazo, J.J. Penco, M.A. Pucheta, M.A. Storti (Eds.)
Concordia, 6-9 Noviembre 2023

INTRODUCTION TO NUMERICAL METHODS IN FLUID MECHANICS WITH JUPYTER NOTEBOOKS

Andrés M. Cimino[a] **and Gustavo J. Krause**[a,b]

[a]*Facultad de Ciencias Exactas Físicas y Naturales, Universidad Nacional de Córdoba, Av. Velez Sarsfield 1611, X5016GCA, Córdoba, Argentina, andres.cimino@unc.edu.ar* https://fcefyn.unc.edu.ar/

[b]*Instituto de Estudios Avanzados en Ingeniería y Tecnología (IDIT–CONICET–UNC), Córdoba, Argentina*

Keywords: Computational Fluid Mechanics, Jupyter, Numerical Methods, Teaching.

Abstract. Introducing the governing equations of fluid dynamics and their discrete numerical counterparts to undergraduate students can be challenging due to their complexity, which sometimes makes it difficult to provide examples or simple solutions. For this reason, we developed a set of interactive notebooks that combine text, equations, and figures like in a textbook with computer code in Python. To achieve this we used the open-source libraries developed by Project Jupyter and the notebooks "12 steps to Navier Stokes" made by Dr. Lorena Barba as a model. Jupyter Notebooks run in any web browser and are compatible with most Python libraries (SciPy, NumPy, SymPy, Matplotlib, etc.), allowing for much flexibility as well as advanced functions. The use of these tools has many advantages, such as obtaining analytical and numerical solutions simply and interactively, or allowing to break down an entire code, explaining and executing each part separately. What is more, students may become more familiar with Python and its libraries, and may reuse the code to solve the assignments of the course. We developed notebooks to cover the governing equations of Fluid Dynamics, an introduction to ordinary and partial differential equations and their numerical and analytical solutions, the Finite Difference Method for Fluid Dynamics, the Finite Volume Method, introduction to pressure-based algorithms for flow problems and gradient and interpolation computations for unstructured meshes. With these tools it was possible to introduce these complex subjects and allow the students to gain insight by performing sensitivity analysis, modifying code or obtaining new solutions based on the existing ones.

1 INTRODUCTION

The governing equations of Fluid Dynamics and other mathematical models based on Partial Differential Equations (PDEs) are sometimes difficult to grasp for students due to their complexity and dependence of many variables, parameters and boundary conditions. Regarding the physical behavior of their solution and boundary conditions they can be classified as equilibrium problems, advance or marching problems or eigenvalue problems. Most textbooks on Computational Fluid Dynamics address these concepts when they introduce the conservation equations and their particular forms. The fact that the same set of equations can represent different physical behaviors is sometimes confusing to the students. For example, an unsteady problem is always a marching problem, whereas a steady state problem may be an equilibrium or a marching problem. However, from a mathematical point of view, it is possible to classify the equations considering whether characteristics curves exist, i.e., if the PDEs can be reduced to ODEs in some regions of the solution plane, which allows the existence of wave-like solutions. For second order PDEs of two independent variables (e.g., x, t), this classification is based on the discriminant formed by the coefficients of greater order terms, giving place to the well known classification of hyperbolic, parabolic and elliptic equations, that is generalized to any PDE (Hoffmann and Chiang, 2000). For this reason in the introduction of our course we classify PDEs on physical and mathematical grounds and relate both categories with practical examples, if possible related to the Navier Stokes equations. It is also useful to introduce different boundary conditions to a given PDE to analyze their effects on the solution. This gives the students the possibility to estimate beforehand the extrema of the solution or to have a qualitative notion of what to expect.

Numerical methods use a discrete and approximated version of the conservation equations. This introduces a difference with the exact solution, which is called Truncation Error (Chapra and Canale, 2015). In most solutions to complex engineering problems using simulations the truncation error can only be estimated because it is not possible to find an analytical solution. However, it is useful for the students to compare analytical solutions to numerical ones to gain insight of how the truncation error grows or is distributed (e.g. with different discretizations or boundary conditions). It is also useful to compare the estimated truncation error given by theory with the true error obtained from the exact solution. Furthermore, some mathematical models have stability issues when solved numerically (Hirsch, 2007). This is also an important topic that is sometimes cumbersome to students.

The simplest PDEs to solve both analytically and numerically are those related to heat conduction (Coleman, 2013), both one and multidimensional, steady and unsteady. The simplest solutions can be extended to more complex ones by adding particular solutions since the heat conduction equation for constant parameters is linear. This allows the student to learn like adding building blocks. The convection-diffusion equation for a scalar ϕ is a good benchmark and starting point to discuss the stability of different numerical methods for fluid flow: it can be solved in one dimension as a steady state solution or in its unsteady version analytically and numerically with the Finite DIfference Method (FDM). It also may be interpreted as some sort of generalization of the heat conduction equation. In addition, it is easy to apply Von Neumann's stability criterion to this solution (Pletcher et al., 2013).

The Finite Volume Method (FVM) is one of the most used techniques to solve engineering and scientific flow problems because it is intuitively related to the conservation laws for a control volume (Versteeg and Malalasekera, 2007). It is easy to apply in the convection-diffusion equation, and later extended to nonlinear equations like the Navier Stokes (NS) equations for

Incompressible Flow. However, the Incompressible NS equations require to solve a zero divergence equation for the conservation of mass. This equation works as a defacto pressure equation linked to the derivative of velocities that creates a coupling problem between both variables. To solve this issue a staggered grid may be used or a interpolation for velocities (Moukalled et al., 2016).

Taking all these issues into account, we started a course on Introduction to Computational Fluid Mechanics for advanced undergraduate and graduate students in 2018. The aim was to emphasize in the implementation and theory behind the models, it was designed for future PhD candidates and MSc students who work developing code for numerical methods. Therefore, after writing some lecture notes we started developing the notebooks as a supplementary tool, so that the students could work in class or at home with complex examples. The idea was to use an environment which is portable and may work in most personal computers, that is easy to use and that is well documented and maintained. We found the lectures of Barba et al. (2014) and took them as a starting point to develop our own. The notebooks developed were then uploaded to GitHub with a Creative Commons License, so that they may be used by anyone (Cimino and Krause, 2023). For the sake of brevity we comment the contents of each notebook, and any interested reader may download them.

2 PYTHON AND ITS LIBRARIES

Python is an interpreted general purpose language specially designed to be easy to read and learn. It is also very flexible, allowing for complex data structures, object oriented and functional programming (Van Rossum et al., 2023). It was developed to be a platform to develop code more quickly and efficiently integrating different libraries and systems. Nowadays it has become very popular for data science, engineering and physics, specially because there are many open source libraries for specific tasks. In addition, there are open source platforms such as Anaconda (Wang et al., 2023) which include an integrated development environment (IDE), the most important libraries for numerical and scientific calculations and visualization, as well as Jupyter notebook. We chose this platform because it has everything we need for our course, it is easy to install in any platform or operating system (OS) and it is supported by a large community of users. We describe briefly the libraries we employed in the following sections.

2.1 Jupyter Notebook

Jupyter Notebook is a web based environment for teaching and scientific computing that can run in most web browsers. It consists of an html document with a particular format, in which there are cell blocks. These can be filled with Markdown, Python code or Raw html code. It was developed initially as Ipython in 2014 but it currently supports many other programming languages, such as R, Julia, C/C++, Matlab, etc. It is an open source project with many stakeholders in enterprises, startups and universities.

The theory and figures are presented using Markdown, a simplified version of html. It allows the simplest features of documents: headings with different hierarchies, bullets, numbering, font size and type, etc. A cheat sheet and extra documentation may be found at Gruber et al. (2023). Equations are written in LaTeX, and special latex commands may be defined as usual. Images can be introduced in a simple form using the command

```
![image_caption](image_file_path.extention)
```

However, this gives no control of its size, location, aspect ratio, etc. Those parameters may be changed using html.

2.2 NumPy

List, Tuples and dictionaries in Python do not follow the same rules as arrays in other scientific programming languages (e.g. homogeneity of elements and dimensions, operations, etc.). Also, most scientific calculations require linear algebra and matricial functions and operations. For this reasons Numpy, an open source numerical calculation library, was created (Olyphant et al., 2023). It supports all basic linear algebra operations for arrays with an implementation of LAPACK, array derived data types like matrices. It also includes advanced linear algebra functions like condition number, matrix factorizations, as well as statistical functions, Fourier transforms and input-output. It is also more computationally efficient since it uses pre-compiled vectorized C code.

2.3 SymPy

To perform symbolic calculations to obtain analytical solutions we use Sympy (Certik et al., 2023), Python's symbolic processor. It is an open source library still evolving and being debugged to become a full fledged Computer Algebra System (CAS). It includes functions for linear algebra, polynomials, ODEs and PDEs, roots of equations, etc. It also has utilities to convert symbolic variables and functions to implement and evaluate them as numerical ones. It is sometimes somewhat cumbersome to use and some bugs occur in simplification, nonetheless it is a useful tool.

2.4 SciPy

Sometimes NumPy is insufficient to perform more complex calculations such as differential equations, eigenvalue problems, work with sparse matrices, perform numerical integration or optimization. For these tasks we use SciPy (Gommers et al., 2023). It uses Numpy as a basis, it has the same high performance features and it is also an open source project with BSD license. It also includes a module with physical and numerical constants. It has the particularity that subpackages must be imported separately.

2.5 Matplotlib

To plot results we use the Matplotlib library (Hunter et al., 2023). The Pyplot subpackage has an interface and commands somewhat similar to those in Octave or Matlab and allows to create both static and dynamic figures as well as animations. It permits exporting the images and videos in different formats easily.

3 NOTEBOOKS FOR CLASSES

In this section we comment briefly the notebooks we created and how they are integrated with the development of the course. They are uploaded to a Github repository of the author with Creative Commons License, so they can be freely downloaded and/or modified by any interested user.

3.1 Governing Equations and Introduction to Partial Differential Equations

There are two sections in the first notebook. In the first one we review the basic conservation laws for a differential control volume. We introduce the conservation of mass, momentum and energy as well as the material derivative and the conservation law for an arbitrary scalar variable ϕ. In this section no practical exercises or calculations were made. We show a snapshot of how

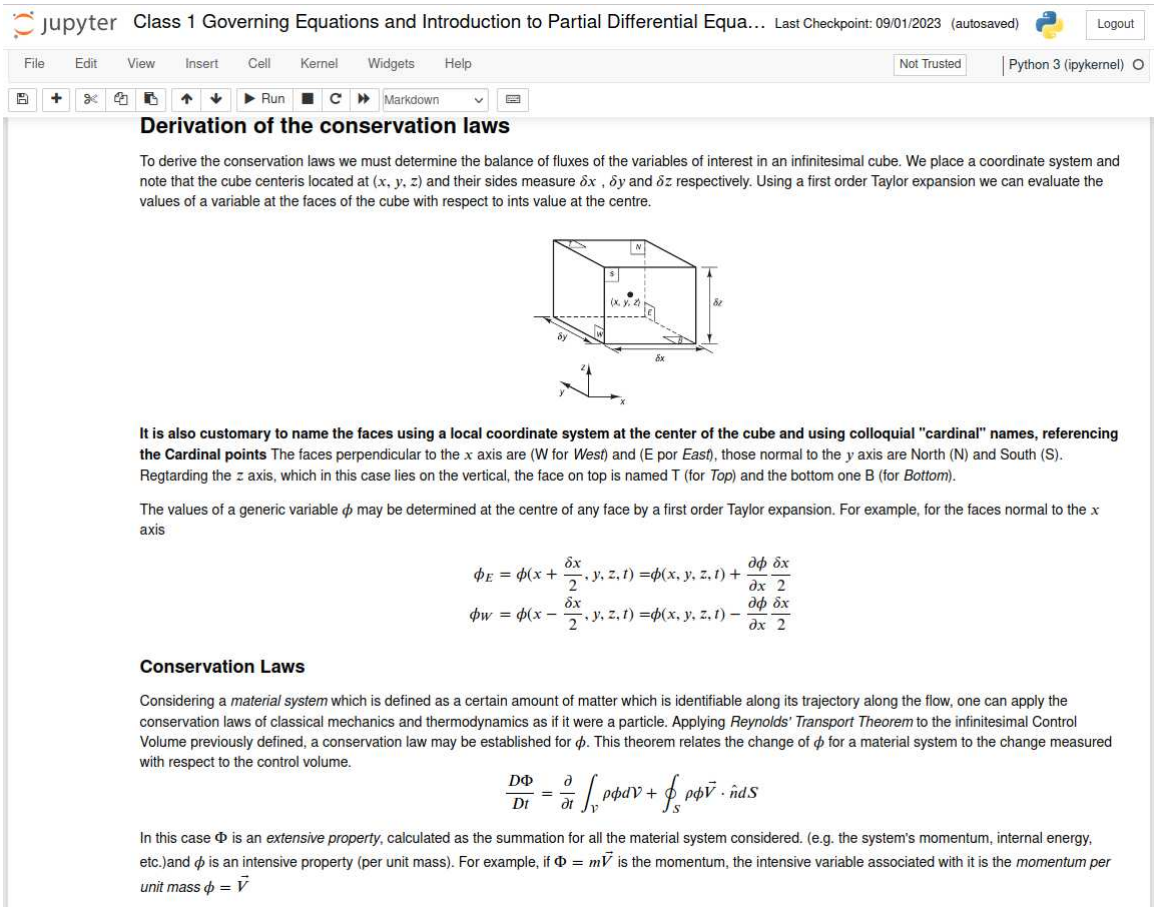


Figure 1: Snapshot of the governing equations in the first notebook.

the equations and figures look like in Fig.1.

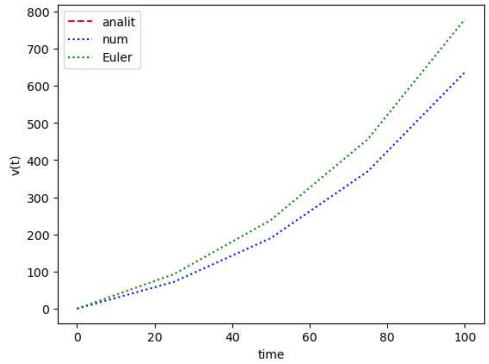
In the second section we make an introduction to partial differential equations (PDEs). We start by making a review of of Ordinary Differential Equations (ODEs) with examples to discuss analytical and numerical solutions for ODEs for different Initial and Boundary Conditions (ICs and BCs). As an example of an Initial Value problem (IVP), we solve the motion of a rocket with a purely vertical motion starting from the ground.

We also solve the temperature distribution along a bar of uniform cross section A and length L as an example of a Boundary Value problem (BVP). We obtained analytical solutions with SymPy and evaluated them with the *Lambdify* utility. We also solve this equation with Scipy's Odeint function and we introduce the solution with Euler's ODE integrator to review it and to compare intuitively truncation errors, as shown in Fig. 2a. We did the same for the Temperature along the bar using SciPy's `solve_bvp` function, introduce the solution using the FDM, and plot the results in Fig. 2b to compare them.

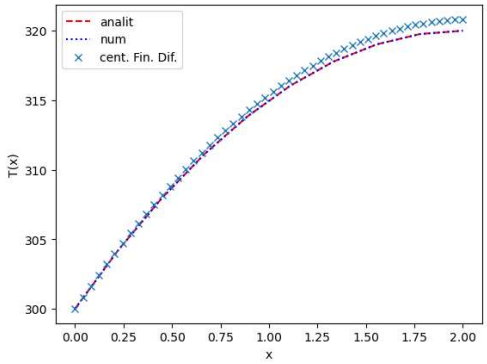
3.1.1 Physical Classification of PDEs

Afterwards we introduce the physical classification of PDEs with examples: we divide them in equilibrium, eigenvalue and advance problems.

Equilibrium Problems Equilibrium problems are those where generally two or more spatial variables are involved (there is no time dependence) and the spatial domain is constrained by



(a) Motion of a rocket moving vertically.



(b) Temperature distribution in a bar.

Figure 2: Solutions of ODEs representing initial and boundary value problems.

boundary conditions. In this case the boundary conditions act as "referees" that define which possible solution in the domain are acceptable. Since the problem is stationary, any change in boundary conditions or inside the domain is instantanously propagated to the entire domain. As an example we present the temperature distribution in a plate of length L and height H, which reduces to Poisson's Equation. We introduce the analytical solution by separation of variables and Fourier Series using Sympy.

$$\kappa \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) = q \tag{1}$$

with boundary conditions

$$T(x = 0, y) = T_0 \qquad T(x, y = 0) = 0 \tag{2}$$

$$T(x, y = H) = 0 \qquad T(x = L, y) = 0 \tag{3}$$

We solve for a single Nonzero Dirichlet BC to use a single Fourier Series. Proposing a separation of variables as

$$T(x, y) = \phi(x)\psi(y)$$

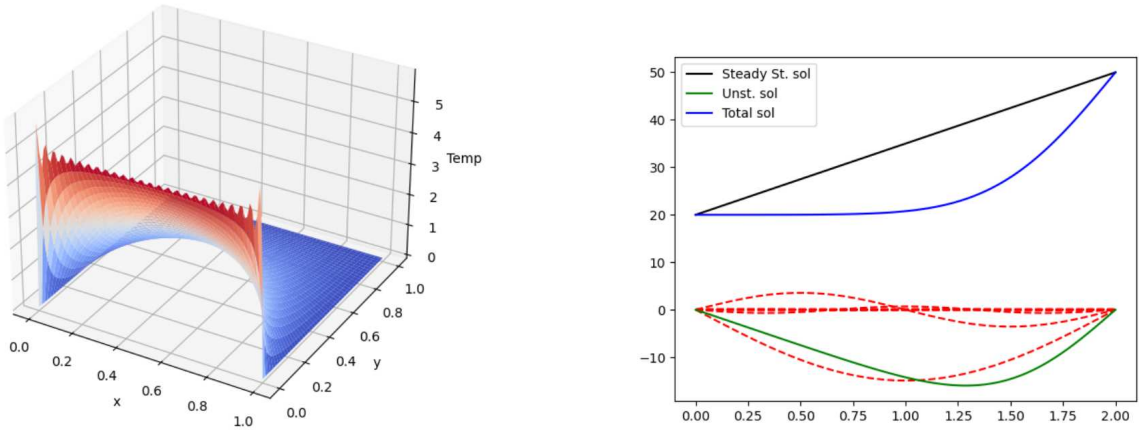
we obtain the solution as Fourier Series like

$$T(x, y) = C_n \sinh \frac{n\pi(x - L)}{H} \sin \frac{n\pi(y)}{H}$$

where the coefficients C_n are obtained using SymPy and evaluating the Fourier coefficients of the considered BC. This solution is useful to compare with numerical solutions in the following notebooks, and can be shown in Fig. 3a.

Eigenvalue Problems In this section we use as an example the bucking of a beam with constant cross section, even though it is an ODE. We do this to present the simplest example possible, and we solve for the eigenvalues with SymPy and SciPy, and plot the first modal form.

Advance Problems These problems usually include time as a variable, so the domain in this dimension is generally infinite, (i.e. there is no upper bound for time). In this case we have boundary conditions in the spatial domain (which may vary in time) and a solution which



(a) Temperature distribution in a plate.

(b) Transient temperature distribution in a bar.

Figure 3: Analytical solutions for the diffusion equation.

evolves in time from its initial state (IC). This solution may reach a Steady state, a periodic or cuasi periodic solution or it may evolve forever. In this case we set an example of the temperature distribution along a bar of length L, density ρ and thermal conductivity κ

$$\rho c_p \frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2} \tag{4}$$

with an arbitrary initial condition

$$T(x, t = 0) = f(x)$$

and boundary conditions

$$T(x = 0, t) = T_0 \tag{5}$$
$$T(x = L, t) = T_L \tag{6}$$

Once again we obtained the solutions with SymPy and evaluate them with Lambidfy,and the solution can be shown in Fig. 3b

3.1.2 Mathematical Classification of PDEs

In this section we introduce the calculation of characteristics for partial differential equations and systems of partial differential equations. Second order PDEs exhibit different mathematical and numerical properties depending on the value of the discriminant of their characteristic equation. Considering a generic PDE for a scalar variable ϕ

$$a\frac{\partial^2 \phi}{\partial t^2} + b\frac{\partial^2 \phi}{\partial t \partial x} + c\frac{\partial^2 \phi}{\partial x^2} = -d\frac{\partial \phi}{\partial t} - e\frac{\partial \phi}{\partial x} - f\phi + g(x,t) = H(\frac{\partial \phi}{\partial t}, \frac{\partial \phi}{\partial x}, \phi, x, t)$$

it may have wave-like continuous or discontinuous solutions, or it may have solutions that are propagated instantaneously to the entire domain. To find out if discontinuous solutions are possible we ask if the partial derivatives of the solution are discontinuous along any curve $\mathcal{C}(\theta)$ in the domain plane, where θ is an arbitrary parameter.

Some authors, such as Hoffmann and Chiang (2000) and Ferziger and Peric (2002) introduce these concepts early on their textbooks to clarify the mathematical properties of different flow models (e.g. boundary layer equations, Navier Stokes, Euler, etc.). We apply this methodology to the 2D Poisson's equation (1), to the unsteady Heat equation along a bar (4), yielding that the former is elliptic and the latter is parabolic. This was intuitively discussed when the analytical and numerical solutions were analyzed.

We also apply it to the transonic potential flow equations for small disturbances (Toro, 2009), assuming $M_\infty < 1$

$$\frac{\partial^2 \phi}{\partial x^2} - \frac{1}{1 - M_\infty^2} \frac{\partial^2 \phi}{\partial y^2} = 0 \tag{7}$$

yielding an hyperbolic system with wave-like solutions in space, even though time is not a variable.

The determination of characteristics may be extended to systems of PDEs in two different ways: one proposed by Hirsch (2007) and another proposed by Hoffmann and Chiang (2000). We apply both methods to Cauchy-Riemann's equations (Hoffmann and Chiang, 2000) to Euler's equations for inviscid incompressible flow (Toro, 2009) as well as Prandtl's boundary layer equations (Hirsch, 2007)

3.2 Introduction to Discretization and Numerical Methods

In the second notebook we introduce the finite difference method, starting with uniform structured discretization of a regular domain in one or two dimensions. We make use of Taylor's series to define truncation error in a discrete grid

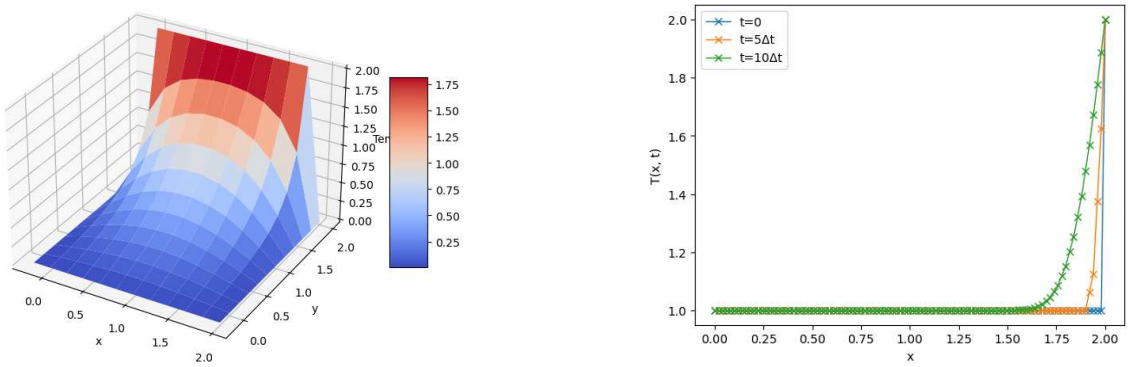
$$\begin{aligned} \phi(x) = \phi(x_i) + (x - x_i) \left.\frac{\partial \phi}{\partial x}\right|_{x_i} + \frac{(x - x_i)^2}{2!} \left.\frac{\partial^2 \phi}{\partial x^2}\right|_{x_i} + \frac{(x - x_i)^3}{3!} \left.\frac{\partial^3 \phi}{\partial x^3}\right|_{x_i} \\ + \cdots + \frac{(x - x_i)^n}{n!} \left.\frac{\partial^n \phi}{\partial x^n}\right|_{x_i} + H, \end{aligned} \tag{8}$$

to derive finite difference formulas for first and second derivatives, introducing the concept of truncation error as the remainder of the series (Chapra and Canale, 2015). We also discuss schemes of different orders of truncation errors (i.e. centered and decentered schemes) and their properties.

This schemes are extended to 2D domains and applied to the numerical solution of Poisson's equation (1), previously discussing the data structure of the algorithm. The solution is plotted in Fig. 4a. Afterwards, we discuss numerical time integration, introducing the difference between explicit and implicit schemes, and their stencils. We solve the unsteady temperature distribution along a bar using forward and backward Euler schemes and Crank-Nicolson, the solution is plotted in Fig. 4b. These examples give rise to the concepts of **Consistency**, **Stability** and **Convergence** (Versteeg and Malalasekera, 2007), which are briefly discussed. We also introduce Von Neumann's stability analysis.

3.3 Convection-Diffusion-Reaction Equation

In this section we apply the numerical methods and stability analysis of the previous notebook to the unsteady convection-diffusion-reaction equation, starting from simplified versions (e.g. steady state 1D convection diffusion) to more complex ones (unsteady 2D convection diffusion). This equation is simple enough to break the ice but it features all the phenomena



(a) Temperature distribution in a plate.

(b) Transient temperature distribution in a bar.

Figure 4: FDM solutions for equilibrium and transient diffusion problems.

present in the conservation law for a variable ϕ in a flow. We introduce the 1D convection-reaction equation (Nigro and Storti, 2011)

$$\frac{\partial \phi}{\partial t} + a_0 \frac{\partial \phi}{\partial x} = \Gamma \frac{\partial^2 \phi}{\partial x^2} + S_\phi \tag{9}$$

with BCs given by

$$\phi(x = 0) = \phi_{in} \tag{10}$$

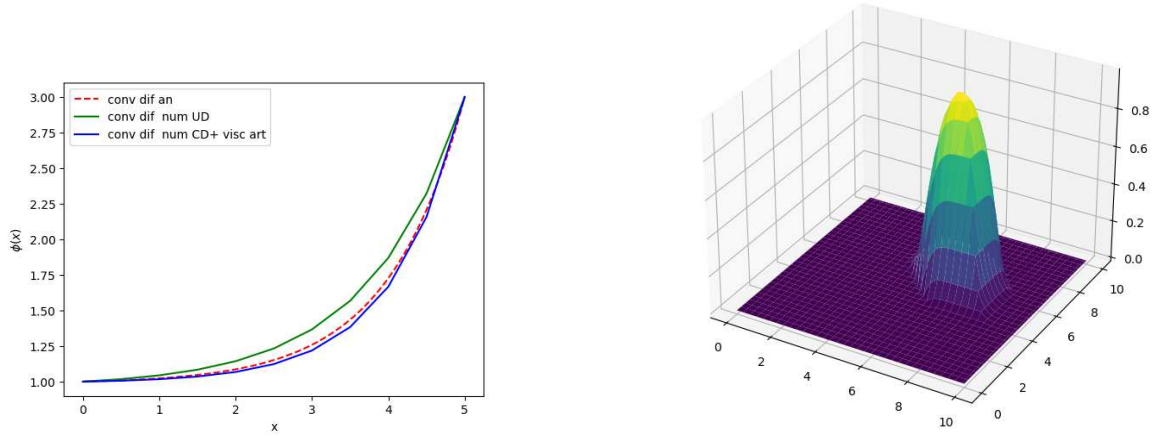
$$\phi(x = L) = \phi_{out} \tag{11}$$

$$\tag{12}$$

Once again we obtain the analytical solution with SymPy as a function of Pèclet's number $\text{Pe}_x = a_0\, x/\Gamma$ based on the local coordinate (Pe_x) and on the entire domain (Pe_L). We then introduce the numerical solution of the 1D steady state convection-diffusion-reaction equation with centered and forward finite differences and different discretizations steps Δx, as can be seen in Fig. 5a. The smaller steps introduce instabilities in the solution with the centered differences, which is afterwards analyzed applying Von Neunmann's criterion (Nigro and Storti, 2011) and Scarborough's criterion (Versteeg and Malalasekera, 2007). We also introduce the concept of **numerical diffusion** and use a scheme that adds it to the physical diffusion, so as to obtain the correct solution

$$-\frac{a_0}{\Delta_x}\left(\phi_i - \phi_{i-1}\right) = -\frac{a_0}{2\Delta_x}\left(\phi_{i+1} - \phi_{i-1}\right) + \frac{\alpha_{num}}{\Delta_x^2}\left(\phi_{i+1} - 2\phi_i + \phi_{i-1}\right) \tag{13}$$

Finally, we repeat these analyses with the unsteady version of the 1D convection-diffusion equation using forward Euler, backward Euler and Crank-Nicolson and upwind and centered differences in space. We produce animations of a transient rectangular wave to illustrate the instabilities. Performing a stability analysis and defining **Courant's number** $\text{CFL} = a_0\Delta_t/\Delta_x$ and **Fourier's number** $\text{Fo} = \Gamma\Delta_t/\Delta_x^2$ we get the well known stability conditions for the explicit scheme $\text{CFL} \leq 1$, $\text{Fo} \leq 0.5$. Then, we repeat this with the 2D version of the unsteady convection diffusion equation, a snapshot of the solution can be seen in Fig. 5b.



(a) 1D convection-diffusion equation.

(b) 2D convection-diffusion equation.

Figure 5: FDM solutions for convection-diffusion problems.

3.4 Introduction to the Finite Volume Method and the Pressure Velocity Coupling Problem

In this section we introduce the integral 1D conservation law for a scalar variable and derive the Finite Volume Method (FVM) from it, and how BCs are implemented. To do this we define the **convective and diffusive Fluxes**

$$F = \rho A u, \qquad D = \frac{\Gamma}{\delta x} \tag{14}$$

We prove that the FVM formulation coincides with the solution of the FDM if the coefficients of the conservation law are constant and the problem is one dimensional. We relate the FVM schemes to how the fluxes in the interfaces of the control volumes are evaluated, and we implement the numerical solution. We show that the results are the same as with the FDM method.

Afterwards we formulate the unsteady form of the FVM using both explicit integration (forward Euler) and implicit (backward Euler) (3.4), obtaining the solutions shown in Fig. 6:

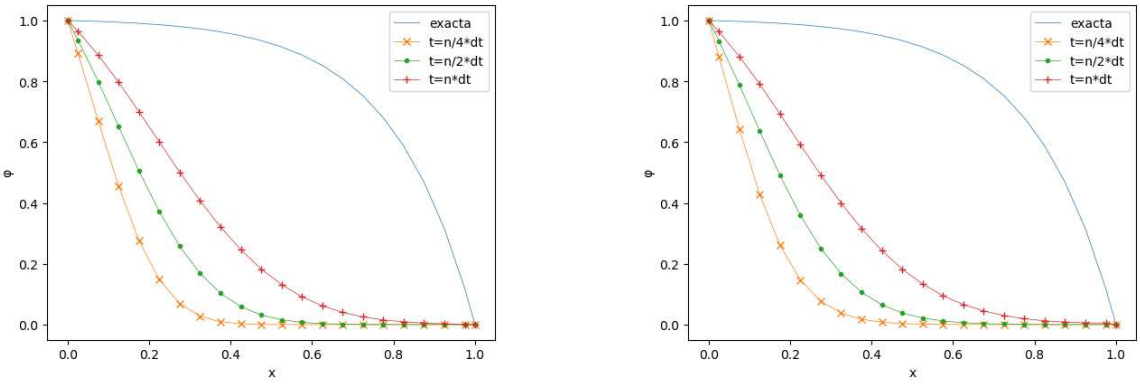
$$\rho\phi_i^{n+1} - \left(F_{i-1/2}^{n+1}\phi_{i-1}^{n+1} - F_{i+1/2}^{n+1}\phi_i^{n+1}\right)\frac{\Delta t}{\Delta x} - \left[D_{i+1/2}^{n+1}(\phi_{i+1}^{n+1} - \phi_i^{n+1}) - D_{i-1/2}^{n+1}(\phi_i^{n+1} - \phi_{i-1}^{n+1})\right]\frac{\Delta t}{\Delta x} = \rho\phi_i^n \tag{15}$$

Then, we introduce the FVM for the Navier Stokes Equations for Incompressible Flow (Versteeg and Malalasekera, 2007)

$$\frac{\partial(\rho uu)}{\partial x} + \frac{\partial(\rho uv)}{\partial y} = \frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu\frac{\partial u}{\partial y}\right) - \frac{\partial p}{\partial x} \tag{16}$$

$$\frac{\partial(\rho uv)}{\partial x} + \frac{\partial(\rho vv)}{\partial y} = \frac{\partial}{\partial x}\left(\mu\frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial y}\left(\mu\frac{\partial v}{\partial y}\right) - \frac{\partial p}{\partial y} \tag{17}$$

When we use the discrete form of these equations based on the FVM, we notice that if the pressure field has the form $p_i = 5$ for even i, $p_i = 10$ for odd i, the pressure field is oscillating



(a) Explicit solution.

(b) Implicit solution.

Figure 6: FVM solutions for the convection-diffusion equation.

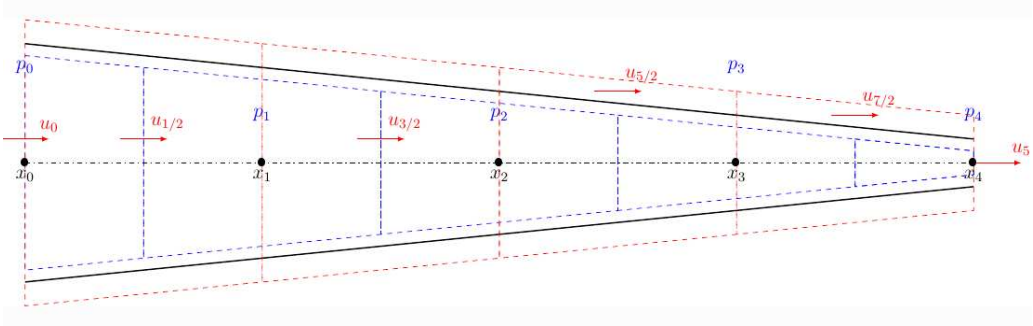


Figure 7: CVs for pressure and velocity for the SIMPLE scheme in a 1D nozzle.

but the derivative of the pressures at interfaces is zero. This gives rise to spurious solutions, so a staggered grid arrangement is proposed (Versteeg and Malalasekera, 2007). Taking this into account we present the **Semi-Implicit Method for Pressure Linked Equations** (SIMPLE) (Patankar and Spalding, 1971) and apply it to an example taken from Versteeg and Malalasekera (2007): a 1D nozzle with inviscid flow with some particularities.

$$\frac{\mathrm{d}\rho A u}{\mathrm{d}x} = 0 \tag{18}$$

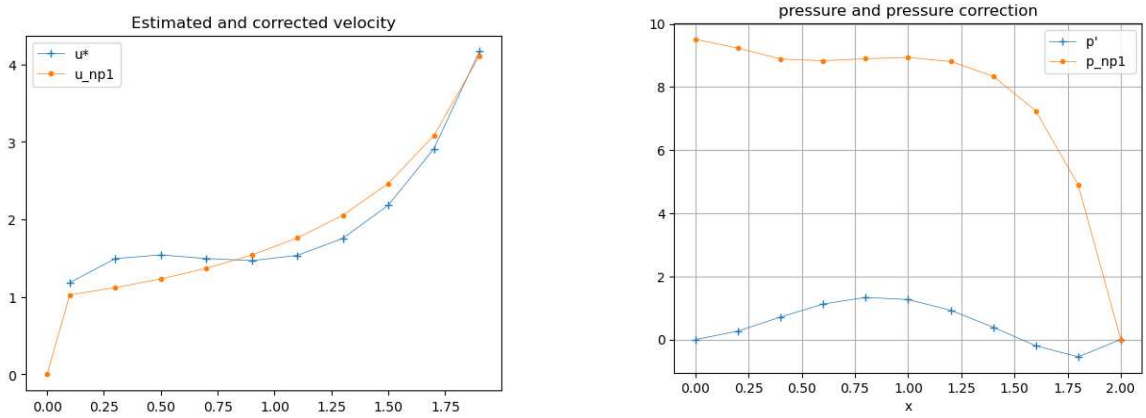
$$\rho u A \frac{\mathrm{d}u}{\mathrm{d}x} = -A \frac{\mathrm{d}p}{\mathrm{d}x} \tag{19}$$

The problem rendered some instabilities, which were corrected with approximated relaxation factors. The solutions are shown in Fig. 8 and the discretization in Fig. 7.

Finally we explain the interpolation method of Rhie and Chow (1983) for collocated grids with examples from Moukalled et al. (2016). We apply it to the 1D Nozzle, obtaining a solution as accurate and more robust as the traditional staggered grid scheme.

3.5 Advanced and Complementary Topics

In this notebook we introduce gradient computation (Green Gauss cell centered, Green Gauss vertex centered and Least Squares) and non orthogonal corrections for unstructured grids as presented in Moukalled et al. (2016). We do this to develop a code for unstructured grids for teaching, a task currently underway. We also present some more advanced methods like the PISO algorithm (Issa, 1985).



(a) Estimated and corrected velocities.

(b) Estimated and corrected pressures.

Figure 8: Numerical results for a 1D nozzle using the SIMPLE algorithm.

4 CONCLUSIONS

The development of these notebooks was long but it helped create a tool to teach a lab of Computational Fluid Mechanics without using more advanced software. The fact that it could be run in almost any computer and that the codes developed are simple and straightforward simplified the assignment of homework: most of the times the student has to extend or modify the code to include more features or a different algorithm. Furthermore, they are a viable tool to introduce the students to LaTeX and Python if they are not familiarized with them. This article is also a way to publicize them so that they hopefully may be used or improved by others.

REFERENCES

Barba L. et al. 12 steps to navier stokes. lecture notes in computational fluid mechanics. 2014.

Certik O. et al. The scipy library. https://www.sympy.org/en/index.html, 2023. Accessed: 2023-08-25.

Chapra S.C. and Canale R. *Numerical Methods for Engineers and Scientists*. McGraw Hill, 2015.

Cimino A.M. and Krause G.J. Github repo for notebooks for introduction to computational fluid mechanics. https://github.com/andrescimino/jupyter_cfd_fcefyn-unc, 2023.

Coleman K.A. *An Introduction to Partial Differential Equations with MATLAB*. CRC Press, 2013.

Ferziger J.H. and Peric M. *Computational Methods for Fluid Dynamics*. Springer, 2002.

Gommers R. et al. The scipy library. https://scipy.org/, 2023. Accessed: 2023-08-25.

Gruber J., Swartz A., et al. https://www.markdownguide.org/cheat-sheet/, 2023. Accessed: 2023-08-25.

Hirsch C. *Numerical Computation of Internal and External Flows*, volume I. Butterworth Heinemann, 2007.

Hoffmann K.A. and Chiang S. *Computational Fluid Dynamics*, volume I. Engineering Education System, 2000.

Hunter J. et al. https://matplotlib.org/, 2023. Accessed: 2023-08-25.

Issa R.I. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 1985.

Moukalled F., Mangani L., and Darwish M. *The Finite Volume Method in Computational Fluid*

Dynamics. An Advanced Introduction with OpenFOAM and Matlab. Springer, 2016.
Nigro N. and Storti M. *Métodos Numéricos en Fenómenos de Transporte*. CIMEC, 2011.
Olyphant T. et al. The numpy library. https://numpy.org/, 2023. Accessed: 2023-08-25.
Patankar S. and Spalding E. A calculation procedure for heat, mass and momentum transfer in three dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 1971.
Pletcher R.H., Tannehill J.C., and Anderson D.A. *Computational Fluid Mechanics and Heat Transfer*. CRC Press, 2013.
Rhie C.M. and Chow W.L. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA/ASME Third Joint Thermophysics, Fluids, Plasma and Heat Transfer Conference*, 1983.
Toro E. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer, 2009.
Van Rossum G. et al. The python language. 2023.
Versteeg H.K. and Malalasekera W. *An Introduction to Computational Fluid Dynamics. THE FINITE VOLUME METHOD*. Prentice Hall, 2007.
Wang P., Oliphant T., et al. https://www.anaconda.com/about-us, 2023. Accessed: 2023-08-25.