ARXDE: Identifying Gradients in Fluid Physics with Autoregressive Modeling

Hanfeng Zhai[*] and Timonthy Sands[†]

Sibley School of Mechanical and Aerospace Engineering, Cornell University

October 12, 2021

Abstract

Fluid mechanics are widely studied and applied to various engineering disciplines including aerospace engineering, mechanical systems, environmental management, etc. Here, we present Auto-Regressive modeling for identifying terms in the ordinary or partial Differential Equations method (ARXDE) and applied it for fluid dynamics modeling. Our model employs the basis of auto-regressive moving average (ARMA) to predict the gradients of fluid flow fields. The flow field computed by computational fluid dynamics (CFD) is input into ARXDE as a numerical matrix and the gradients are computed from discretization to fit a linear model. The ARXDE predicts the flow field by employing ARMA to fit such a linear model. The ARXDE framework was applied to predict the gradients of the velocity of the turbulence fluid field in this paper. The results indicate with an increasing order with higher derivatives, ARXDE captures the physical trends of gradients more accurately, but generally displays higher mean square errors (MSE) since the numerical value of higher-order discretizations is bigger. Also, the CPU computation time is longer for higher orders higher derivatives. Our model brings in a novel approach for model higher-order nonlinear systems with linear models.

1 Introduction

Fluid dynamics (or fluid mechanics), begins in approximately 300 B.C. when Aristotle & Archimedes first elicit the continuum nature of fluids, the concept of resistance, pressure, and body force [1, 2]. Scientist first identifies the governing equation of fluid mechanics

[*]Email: hz253@cornell.edu
[†]Email: tas297@cornell.edu

in 19th centuries [3], which are known as the classic Navier-Stokes equation and widely applied in various industries including aerospace engineering [4], automobile industries [5], climate modeling [6], etc. Notably, the computation modeling complex climate system, which mostly involves fluid dynamics, was awarded Nobel Prize in Physics this year [7]. Moreover, computation modeling of the fluid systems and wind tunnel experiments are the key steps for developing new industry products in aerospace engineering [8, 9]. Hence, fluid dynamics modeling is a key issue that requires further studies in a nowadays fast-changing world.

However, due to the nonlinear nature of the Navier-Stokes equation, it is usually impossible to solve the equation analytically due to the highly non-linearity nature of the equation. Hence, various numerical method was developed for solving this equation including finite difference method [10], finite element method [11], finite volume method [12], smooth particle hydrodynamics [13], lattice Boltzmann method [14], etc. These numerical methods employ some basic characteristics: both aim to discretize the gradients (or derivatives) involved in the equation and turned them into linear multiplication of discretized numerical matrix for the solution of the fluid systems. Different methods apply different strategies yet both share some common nature: with the higher-order discretization, the physics are computed more accurately as the discretization approximates the real gradients, yet cost consumable higher computational resources.

Stepping into the 21st century, as data exploded at an unprecedented speed and computation resources increases drastically, artificial intelligence (AI) technologies including machine learning, deep learning, deterministic AI grows intensely. Especially, different data-driven methods that encoded physics information, i.e. governing equations, boundary conditions, initial conditions, are becoming a mainstream machine learning method for modeling fluid dynamics in recent years [16, 17, 15]. Zhai *et al.* find out that even not fully encoded physics information could still lead to generally good prediction results compared with pure deep neural networks, indicating the power of inner-encoded physics constraints [18]. However, with the extra focus on data-driven methods, control modeling methods and linear models are not as pursued heatedly by fluid mechanicians as deep learning since they didn't receive much attention. To emphasize, autoregressive modeling has rarely been adopted by physicists for fluid dynamics problems over the years. Some applications indicate autoregressive moving average (ARMA) displays good modeling accuracy with experimental data of turbulent [19]. And the damping between fluid-structure interactions and flight trajectories can also be modeled by ARMA [20, 21]. But applying ARMA to model computational fluid dynamics (CFD) is rarely touched and elaborated in the previous works.

Here, the ARMA modeling method is applied for modeling gradients calculated from CFD. The authors present Auto-Regressive modeling for identifying terms in the ordinary or partial Differential Equations method (ARXDE). To specify, the velocity field is computed in a $k-\epsilon$ turbulent model by CFD implemented in COMSOL Multiphysics®. The velocity field is thence discretized for gradients calculation using different order of finite difference

(FDM) schemes [22]. We mainly focus on the first and second derivatives of velocity fields with first and second-order schemes. The discretized data are modeled by ARXDE for and the benchmark by discretization of a different order of FDM schemes are compared with the ARXDE predictions.

In this manuscript, the basic mathematical model and the fluid mechanics problem are first formulated and defined in Section 2. Later on, the basic model of ARMA and the construction of the ARXDE framework are presented in Section 3. The prediction results, the mean square errors (MSE), and the CPU computation time comparison is presented as in Section 4. Eventually, the manuscript is briefly concluded in Section 5.

2 Problem Formulation

In fluid dynamics, the fluid flow behavior is governed by the Navier-Stokes (N-S) Equation, which can be written in the conservative form as:

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho\left(\mathbf{u}\cdot\nabla\right)\mathbf{u} = \nabla\cdot\left[-p\mathbf{I}+\mathbf{K}\right]+\mathbf{F} \tag{1}$$

where $\mathbf{u}$ is the velocity of the fluid, ρ is the density, $\mathbf{F}$ is the external force and p is the hydrostatic pressure.

Also, fluid obeys the conservation of mass, or can also be described as the continuum condition, taking the form:

$$\rho\nabla\cdot\mathbf{u} = 0 \tag{2}$$

Where in Equation 1 the $\mathbf{K}$ term takes the form:

$$\mathbf{K} = \left(\mu+\mu_T\right)\left(\nabla\mathbf{u}+\left(\nabla\mathbf{u}\right)^T\right) \tag{3}$$

For real-world fluid mechanics applications, when the Reynolds number $Re = \frac{\rho v L}{\mu}$, where L is the characteristic length and v is the characteristic velocity, exceeds a specific value about 4000 [23], the fluid flow is considered as turbulence. Here, in the fluid simulation, the $k-\epsilon$ turbulence model is employed, which further expands the N-S equation into the form:

$$\begin{aligned}
&\rho\frac{\partial k}{\partial t} + \rho\left(\mathbf{u}\cdot\nabla\right)k = \nabla\cdot\left[\left(\mu+\frac{\mu_T}{\sigma_k}\right)\nabla k\right]+p_k-\rho\epsilon \\
&\rho\frac{\partial \epsilon}{\partial t} + \rho(\mathbf{u}\cdot\nabla)\epsilon = \nabla\cdot\left[\left(\mu+\frac{\mu_T}{\sigma_\epsilon}\right)\right]+C_{\epsilon 1}\frac{\epsilon}{k}p_k - C_{\epsilon 2}\rho\frac{\epsilon^2}{k},\ \text{where}\quad \epsilon = ep
\end{aligned} \tag{4}$$

where k is the first transported variable is the turbulent kinetic energy, and ϵ is the second transported variable is the rate of dissipation of turbulent kinetic energy. The equations also consist of some adjustable constants σ_k, σ_ε, $C_{1\varepsilon}$ and $C_{2\varepsilon}$. The values of these constants have been arrived at by numerous iterations of data fitting for a wide range of turbulent flows [24].

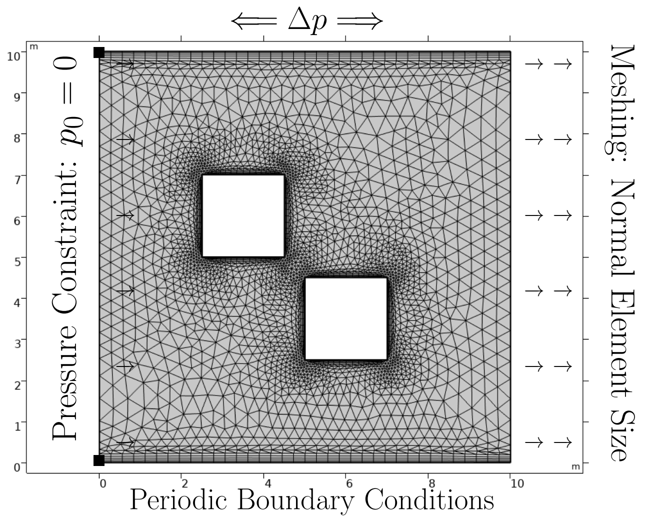
$\Longleftarrow \Delta p \Longrightarrow$
Pressure Constraint: $p_0 = 0$
Meshing: Normal Element Size
Periodic Boundary Conditions

Figure 1: The basic modeling and meshing for the computation domain of the fluid flow field. The model is implemented in COMSOL Multiphysics®, where the meshing are selected as *Normal*. The left side is set as the inflow and the right side is the outflow. A periodic boundary conditions (PBC) is implemented, where the pressure difference between the two sides $\Delta p = 100$Pa. Two pressure constraint point is set as the inflow where $p_0 = 0$. The $k - \epsilon$ turbulence computation model is adopted to compute the fluid field.

The schematic of the fluid mechanics model is shown in Fig1, where there are an inflow and an outflow passing through two solid obstacles that may generate vortexes. A periodic boundary condition (PBC) is implemented, where the pressure difference between the two sides $\Delta p = 100$Pa. Two pressure constraint point is set as the inflow where $p_0 = 0$. On the wall between the fluid and solid, we adopt the no-slip boundary conditions: $\mathbf{u} \cdot \mathbf{n} = 0$ and pressure constraint $p = p_0$. As for the mentioned PBC, the mathematical formulation takes the form [25]:

$$\begin{aligned} \mathbf{u}_{in} = \mathbf{u}_{out}, \quad p_{out} = p_{in} + \Delta p, \\ k_{in} = k_{out}, \quad \epsilon_{in} = \epsilon_{out} \end{aligned} \tag{5}$$

Above are the simplified governing equations controlling the whole fluid flow process. In the real modeling, we implement such a model in COMSOL Multiphysics1, where a normal element size meshing is adopted. The model also tells us that mesh becomes denser on the boundaries and walls since the calculation of the no-slip conditions requires higher accuracy. The simulation adopts a time-dependent CFD solver to compute the flow field for 300s. The flow field was output per 0.5s.

The numerical simulation of fluid flow fields is shown as in Figure 2. From the flow field, it can be observed that there are small vortices generated along the solid boundaries. At the beginning stage in $t = 5s$ in subfigure **A**, the velocity exhibits higher values along the corner of the solid boundaries. When it goes to $t = 25s$ in subfigure **B**, a higher velocity difference between the boundaries surrounding area and outer fluid zone can be observed. As for $t = 125s$ in subfigure **C**, the flow field tends to approximate a stable stage where the

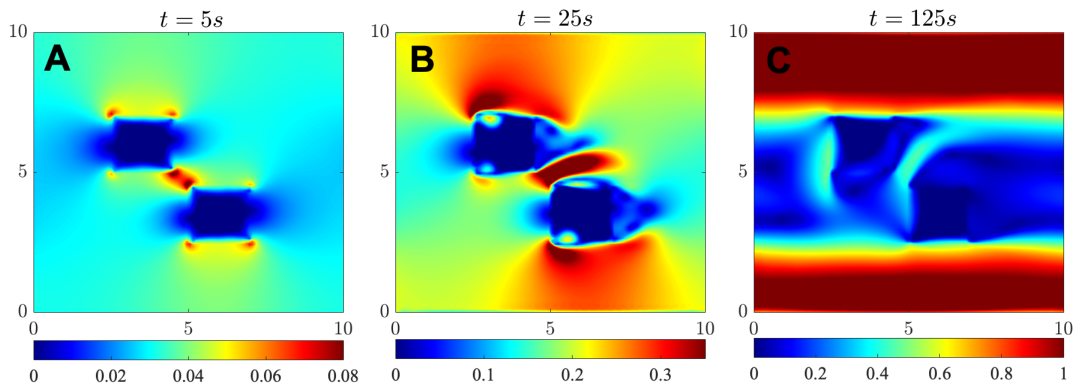

Figure 2: The velocity u field of the numerical computation results by CFD in COMSOL Multiphysics®, where the time are computed for 300s. The figure presented the flow field changes from $t = 5s$ to $t = 125s$ from **A** to **C**. Note that the fluid field become stable after $t \approx 125s$.

rest 100 more seconds the flow field stays in the same trend.

3 Data-driven modeling

To model and predict the gradients involved in the fluid field presented in Section 2, the ARMA method is adopted as introduced by Astrom [26]. The prediction of the gradient $\mathcal{G}$ by ARXDE is written as $\hat{\mathcal{G}}$, i.e. $u_{,x}$, $u_{,xx}$, $u_{,xy}$, *etc.*, is related to the fluid velocity in a linear relation parameterized by matrices A and B:

$$A(q)\hat{\mathcal{G}}(x) = B(q)u(x) \tag{6}$$

3.1 Autoregressive moving average

The ARMA model begin with a truth model to generate noisy outputs measurement time histories for constructing ARMA for gradients discretization:

$$\hat{\mathcal{G}}(x_i) = \frac{B(q)}{A(q)}u(x_i) \tag{7}$$

To construct of autoregressive moving average, we first expand matrices A and B in Equation 6:

$$\left[q^n + a_1 q^{n-1} + ... + a_n\right]\hat{\mathcal{G}}(x) = \left[b_1 q^{m-1} + b_2 q^{m-2} + ... + b_m\right]u(x) \tag{8}$$

Multiplying q^{-n} on the two sides of the Equation 8, the equation can be further expanded to

$$\left[1 + a_1 q^{-1} + a_2 q^{-2} ... + a_n q^{-n}\right]\hat{\mathcal{G}}(x) = q^{-n}\left[b_1 q^{m-n-1} + b_2 q^{m-n-2} + ... + b_m q^{-n}\right]u(x) \tag{9}$$

Multiplying out the equation and $\hat{\mathcal{G}}$ can be further solved:

$$\begin{aligned}\hat{\mathcal{G}}(x) = -a_1 q^{-1}\hat{\mathcal{G}}(x) - a_2 q^{-2}\hat{\mathcal{G}}(x) - ... - a_n q^{-n}\hat{\mathcal{G}}(x)\\ +b_1 q^{m-n-1}u(x) + b_2 q^{m-n-2}u(x) + ... + b_n q^{-n}u(x)\end{aligned} \tag{10}$$

Implement time shift operations: $q^{-1} \rightarrow (x-1)$, $q^{-2} \rightarrow (x-2)$, $q^{-n} \rightarrow (x-n)$, the equation can be further expanded into:

$$\begin{aligned}\hat{\mathcal{G}}(x) = -a_1\hat{\mathcal{G}}(x-1) - a_2\hat{\mathcal{G}}(x-2) - ... - a_n\hat{\mathcal{G}}(x-n)\\ +b_1 u(x+m-n-1) + b_2 u(x+m-n-2) + ... + b_m u(x-n)\end{aligned} \tag{11}$$

Now, for further modeling, the parameters $[\mathbf{a}, \mathbf{b}]$ are reconstructed into $[\alpha, \beta]$. Then Equation 11 can be further written into:

$$\begin{aligned}\hat{\mathcal{G}}(x) = -\alpha_1\hat{\mathcal{G}}(x-1) - ... - \alpha_n\hat{\mathcal{G}}(x-n)\\ +\beta_0 u(x+m-n) + \beta_1 u(x+m-n-1) + ... + \beta_m u(x-n)\end{aligned} \tag{12}$$

The final model is contructed in the form:

$$\hat{\mathcal{G}}(x) = \Phi(x-1)\hat{\theta} \tag{13}$$

where $\hat{\theta} = [\alpha_1,\ \alpha_2,\ ...\alpha_n,\ \beta_0,\ \beta_1,\ ...,\ \beta_m]^T$ and $\Phi = [-\mathcal{G}(x-1),\ -\mathcal{G}(x-2),\ ...,\ -\mathcal{G}(x-n),\ u(x+m-n),\ u(x+m-n+1),\ ...,\ u(x-n)]^T$. At this stage, Equation 13 is adopted to be the fitting equation for inferring the gradients of the fluid velocity field.

3.2 Gradients discretization

With the constructed ARMA model, we need to feed data and construct a discretized gradient calculation method for construct the final ARXDE framework. As proposed in Section 1, we adopt different orders of finite difference schemes for calculating the different derivatives of fluid velocity u [22]. Due to the presented ARMA modeling method depend only on the preceding steps, the left-sided finite difference schemes are thence adopted for discretizing the gradients of fluid fields.

First, the first derivative of velocity $\nabla_x u$, which can also be written as ∇u, can be discretized i by left-sided finite difference scheme in first order:

$$\left.\frac{\partial u}{\partial x}\right|_{x_i} = \frac{u_i - u_{i-1}}{\Delta x} + \mathfrak{O}(\Delta x) \tag{14}$$

Besides, the term ∇u can be discretized by left-sided finite difference scheme in second order for first derivative, which displays a higher accuracy for discretization:

$$\left.\frac{\partial u}{\partial x}\right|_{x_i} = \frac{3u_i - 4u_{i-1} + u_{i-2}}{2\Delta x} + \mathfrak{O}(\Delta x^2) \tag{15}$$

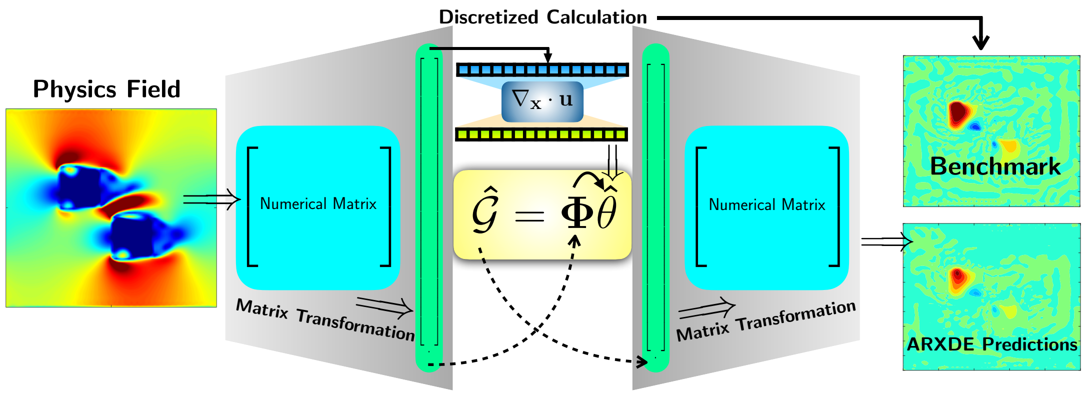

Figure 3: The schematic illustration for ARXDE. The fluid flow field computed by CFD is input into the framework as a numerical matrix (light blue square), whom further got reduced to a one-dimensional vector (light green rectangle); the vector are input to the matrix Φ as past values; the vector was also conducted a discretized calculation, where in our case is the gradients calculation with different orders, was input to the unknown regressors $\hat{\theta}$. The expected output predictions of the discretized gradient $\hat{\mathcal{G}}$ was output as a vector and then transfer back to a matrix that been shown as the ARXDE predictions.

The second derivative of the velocity u, written as $\nabla^2 u$ or Δu, can be discretized in left-sided finite difference scheme in first order:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} = \frac{u_i - 2u_{i-1} + u_{i-2}}{\Delta x^2} + \mathfrak{O}(\Delta x) \tag{16}$$

Similarly, Δu can also be discretized in left-sided finite difference scheme second order:

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} = \frac{2u_i - 5u_{i-1} + 4u_{i-2} - u_{i-3}}{\Delta x^2} + \mathfrak{O}(\Delta x^2) \tag{17}$$

Now, Equations 14-17 construct the discretized gradient calculation of our ARXDE model, which will be further discussed as elaborating the construction of ARXDE.

3.3 Model construction

The schematic illustration of the ARXDE model is shown as in Figure 3. Equations 14-17 are inserted as the gradient discretization as in the shaded blue square as on the mid-top side in the schematic. The fluid flow field computed by CFD is input into the framework as a numerical matrix and transformed into a one-dimensional vector. The ARMA model derived in Equation 13 is illustrated in the light yellow square in the schematic. The Φ matrix consists of values of the preceding steps and the gradients discretizations were input to the unknown regressors $\hat{\theta}$. The expected output predictions of the discretized gradient $\hat{\mathcal{G}}$ were output as a vector and then transfer back to a matrix that has been shown as the ARXDE predictions as illustrated in the contours.

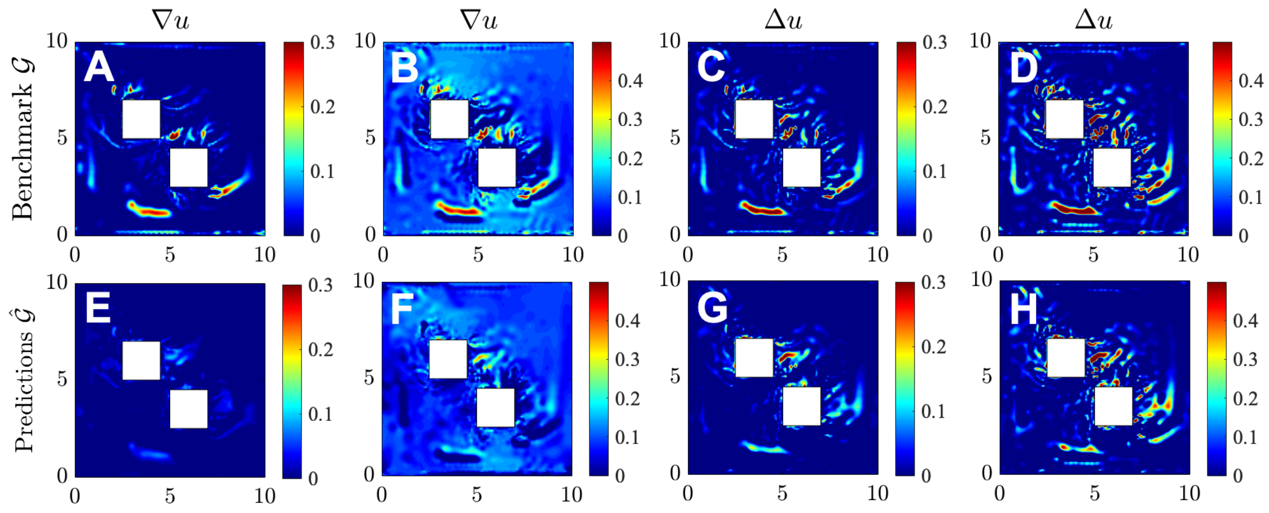

Figure 4: The prediction results compared with the discretization benchmark for the first and second derivatives of velocity u, with first and second orders. The sub figures **A, B, E, F** stands for benchmark and predictions with 1st and 2nd order discretization methods for the first derivative of velocity u. The sub figures **C, D, G, H** stands for benchmark and predictions with 1st and 2nd order discretization methods for the second derivative of velocity u.

4 Results and discussion

The ARXDE predictions compared with the benchmarks discretized by finite difference schemes on ∇u and Δu regarding different orders are shown in Figure 4. It can be observed from the predictions that for the same derivatives (comparing **A, B, E, F** or **C, D, G, H**), higher-order discretization generally displays higher accuracy. For the same orders discretization (comparing **A, C, E, G** or **B, D, F, H**), higher derivatives generally displays higher accuracy. With the increasing order, the discretized gradients benchmarks display higher numerical values.

The logarithmic plots of mean squared errors (MSE) regarding the increasing orders $\mathcal{O} = n + m$ as introduced in Equation 12 are shown in Figure 5. It can be deduced that with the increasing derivatives and higher-order discretizations the MSE increases. But MSE generally displays more evident decreasing trends with higher orders $\mathcal{O}$ for higher derivatives.

The CPU computation time for different gradient calculations with different orders implemented in Figure 6. Note that the computation was conducted on Intel® Core$^{\text{TM}}$ i5-8250U CPU @ 1.60GHz 1.80GHz. It can be deduced that with increasing derivatives and higher orders the computation time increases in a seemingly linear way. Such a phenomenon can be explained by an increasing multiplication operation implemented in algorithms that increase the CPU computation in a linear way since the increasing orders of gradients discretizations as shown in Equations 14-17 are combinations of linear multiplications.

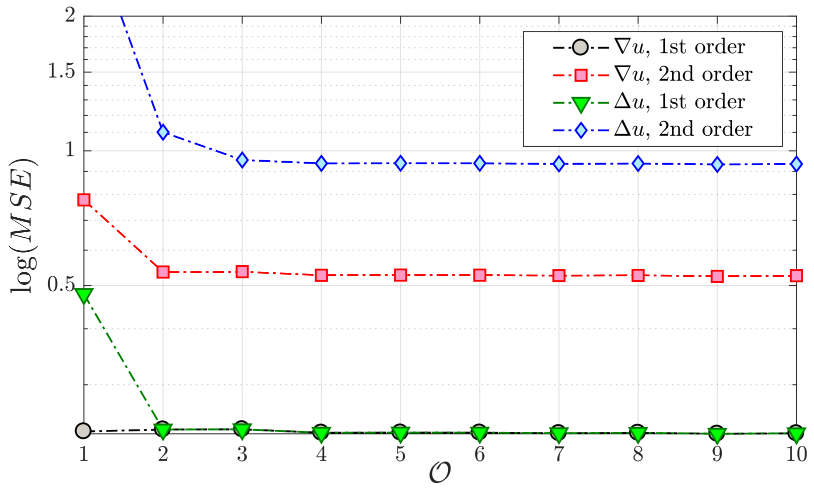

Figure 5: The logarithmic mean square errors (MSE) for four different ARXDE modeling methods regarding orders $\mathcal{O} = n + m$. The first derivatives (∇u) for 1st and 2nd order modeling are plotted in black and red dotted lines; and second derivatives ($\nabla^2 u = \Delta u$) for 1st and 2nd order modeling are plotted in green and blue dotted lines.

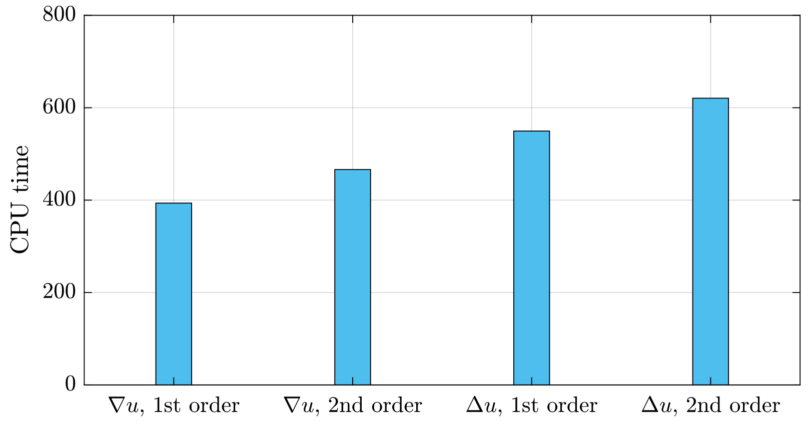

Figure 6: The CPU computation time for ARXDE modeling regarding different derivatives with different orders.

5 Conclusion

In this manuscript, the ARMA method was employed to solve a century-old fluid mechanics problem: nonlinearity. A novel linear-based learning model build on top of ARMA and gradient discretization called ARXDE is presented to infer different orders of different derivatives in fluid mechanics problems. The velocity of the fluid field was discretized by 1st and 2nd order left-sided finite different method for first and second derivatives as for the data feed for ARXDE. The fluid field data were numerated and transformed into a one-dimensional vector to feed the regression model. The prediction results indicate with higher derivatives and higher orders, the results of the prediction capture the physical trend more accurately. However, the MSE results indicate the opposite, where the numerical value of errors is actually higher. Such a trend can be accounted for by the higher values of the benchmark with higher derivatives higher-order models. Simultaneously, as the derivatives and orders go higher, the CPU computation time also increases in a seemingly linear trend. Our presented framework could bring insights for model nonlinear fluid systems and inferring complex fluid flow systems.

[redacted]

- [redacted]
- [redacted]
- [redacted]
- [redacted]

References

[1] M. Bahrami. Intro and Fluid Properties. *Fluid Mechanics (S 09)*. URL: http://www.sfu.ca/~mbahrami/ENSC%20283/Notes/Introduction.pdf.

[2] H. Zhai. Fluid Dynamics Basis & SPH Method. URL: https://hanfengzhai.net/note/SPH.pdf.

[3] J.D. Anderson, Jr. Governing Equations of Fluid Dynamics., *Computational Fluid Dynamics*, 3rd ed., Springer-Verlag Berlin Heidelberg 2009.

[4] Qun Zhang, Song Cen. 10 - Computational fluid dynamics in aerospace field and CFD-based multidisciplinary simulations. Multiphysics Modeling. Numerical Methods and Engineering Applications. *Elsevier and Tsinghua University Press Computational Mechanics Series.* `DOI:10.1016/B978-0-12-407709-6.00010-9`.

[5] Zhang, X. and Romzek, M., ”Computational Fluid Dynamics (CFD) Applications in Vehicle Exhaust System,” SAE Technical Paper 2008-01-0612, 2008.

[6] Pasha Piroozmand, Gianluca Mussetti, Jonas Allegrini, Mohammad Haji Mohammadi, Ehsan Akrami, Jan Carmeliet. Coupled CFD framework with mesoscale urban climate model: Application to microscale urban flows with weak synoptic forcing. *Journal of Wind Engineering and Industrial Aerodynamics.* Volume 197, February 2020, 104059.

[7] Davide Castelvecchi & Nisha Gaind. Climate modellers and theorist of complex systems share physics Nobel. *Nature News.* URL: https://www.nature.com/articles/d41586-021-02703-3.

[8] Radu Crahmaliuc. 5 Ready-to-Use CFD Simulations for Aircraft Design. *SimScale.* URL: https://www.simscale.com/blog/2017/01/5-cfd-simulations-aircraft-design/.

[9] Michael Braukus, Kathy Barnstorff, Lori Gunter. Industry Uses NASA Wind Tunnels to Design New Airplanes. *NASA News.* RELEASE : 06-060. URL:https://www.nasa.gov/home/hqnews/2006/feb/HQ_06060_LaRC_wind_tunnel.html#:~:text=The%20National%20Transonic%20Facility%20is,size%20of%20the%20actual%20aircraft..

[10] Christian Grossmann; Hans-G. Roos; Martin Stynes (2007). Numerical Treatment of Partial Differential Equations. Springer Science & Business Media. p. 23. ISBN 978-3-540-71584-9.

[11] Turner, M. J., Clough, R. W., Martin H. C. and Topp, L. J. (1956). Stiffness and Deflection Analysis of Complex Structures. Journal of the Aeronautical Sciences. `DOI:10.2514/8.3664`

[12] LeVeque, Randall (2002). Finite Volume Methods for Hyperbolic Problems. ISBN 9780511791253.

[13] Lind SJ, Rogers BD, StansbyPK. 2020 Review of smoothed particle hydrodynamics: towards converged Lagrangian flow modelling. *Proc. R. Soc. A* **476**:20190801.

[14] Chen, Shiyi; Doolen, Gary D. (1998). "Lattice Boltzmann Method for Fluid Flows". *Annual Review of Fluid Mechanics.* 30 (1): 329–364. Bibcode:1998AnRFM..30..329C. ISSN 0066-4189.

[15] von Rueden, L., Mayer, S., Beckh, K., et al. (2021) Informed Machine Learning - A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems. in *IEEE Transactions on Knowledge and Data Engineering.* 1 - 1.

[16] Raissi, M., Perdikaris, P., and Karniadakis, G.E. (2019) 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.

[17] Karniadakis, G.E., Kevrekidis, I.G., Lu, L., et al. Physics-informed machine learning. *Nat. Rev. Phys.* **3**, 422–440 (2021).

[18] Zhai, H., Zhou, Q., and Hu, G. BubbleNet: Inferring micro-bubble dynamics with semi-physics-informed deep learning. arXiv preprint. `arxiv:2105.07179`.

[19] Davide Faranda, Flavio Maria Emanuele Pons, Bérèngere Dubrulle, François Daviaud, Brice Saint-Michel, Éric Herbert, Pierre-Philippe Cortet. Modelling and analysis of turbulent datasets using ARMA processes. arXiv preprint. `arxiv: arXiv:1404.0547`.

[20] H.J. Zhang, Y. Zhou, R. M. C. So, M. P. Mignolet & Z.J. Wang. Fluid and Structural Damping Measurement Using an ARMA Technique. Conference: Proceedings of the International Conference on Advances in Structural Dynamics 2000 At: Hong Kong, HKSAR, PRCVolume: pp. 1109-1116.

[21] Yu Xing, Gang Wang, Yanan Zhu. Application of an Autoregressive Moving Average Approach in Flight Trajectory Simulation. AIAA AVIATION Forum. 13-17 June 2016, Washington, D.C. AIAA Atmospheric Flight Mechanics Conference.

[22] Ardeshir Hanifi. Finite Difference Schemes. *Computational Fluid Dynamics SG2212 (20100123).* URL:`https://www.mech.kth.se/~ardeshir/courses/literature/fd.pdf`.

[23] Wikipedia. Turbulence. URL:`https://en.wikipedia.org/wiki/Turbulence`.

[24] Wikipedia. K-epsilon turbulence model. URL:`https://en.wikipedia.org/wiki/K-epsilon_turbulence_model`.

[25] COMSOL Doc. Wall. URL:`https://doc.comsol.com/5.5/doc/com.comsol.help.comsol/comsol_ref_fluidflow.20.26.html`.

[26] Astrom KJ and Wittenmark B. Adaptive Control (2nd Edition). Dover Publications, Inc. **2008.**