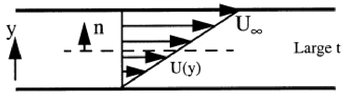
Solving compressible plane Couette flow with numerical methods and physics-informed neural networks

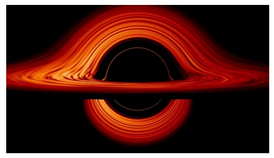
Jingrui (Ray) Hu
Mentor: Prof. Chang Liu, UConn School of Mechanical, Aerospace, and Mfg. Engineering: changliulab.engineering.uconn.edu

UCONN | COLLEGE OF ENGINEERING

Introduction



- Shear-driven flow
- Moving upper, stationary lower boundary
- Compressible flow applications
  - Aerodynamic heating simulation
  - Boundary layer analysis
  - High speed aerodynamics – rockets, military
  - Astrophysics – accretion disks



Steady state

- Describes the fluid state at equilibrium
- The no-slip boundary condition is obeyed

Momentum: $\tau_0 = \eta_0 \frac{dU_0}{dy}, \frac{d\tau_0}{dy} = 0$

Energy: $\frac{d}{dy}\left((\gamma-1){M_r}^2\tau_0 U_0 + \frac{\eta_0}{Pr}\frac{dT_0}{dy}\right) = 0$

Boundary conditions restrict possible solutions:

$U_0(0) = 0, U_0(1) = 1, T_0(0) = T_L, T_0(1) = 1$

- M_r = Mach number, $U_0(y)$ = velocity at y height
- Pr and γ are properties of the fluid (0.72, 1.4)
- η_0 depends on temperature (Sutherland's law)
- τ_0 is the shear stress profile (unknown)
- Flow does not depend on Reynolds number [1]

Startup

- Describes the fluid's behavior as it accelerates
- Velocity approaches steady-state as t increases
- Since the functions vary with both y and t, the equations become partial differential equations

Momentum: $\tau = \eta_0 \frac{\partial U}{\partial y}, \frac{\partial U}{\partial t} - \frac{\partial \tau}{\partial y} = 0$

Energy: $\frac{\partial T}{\partial t} - \frac{\partial}{\partial y}\left((\gamma-1){M_r}^2\tau_0 U_0 + \frac{\eta_0}{Pr}\frac{\partial T}{\partial y}\right) = 0$

Initial conditions: $U(y,0) = 0, T(y,0) = 1$

Boundary conditions are the same as above.

Generating training data

Steady state solution (shooting method, Fig. 1)

- Integrates the ODE system using an initial guess for unknown τ
- Improve the guess based on the boundary condition residuals (calculated - expected)
- Integrates derivatives using `scipy.integrate.solve_ivp` (RK45)
- Optimizes τ guess using `scipy.optimize.root_scalar` (Brentq)

Startup solution (Fig. 2)

- τ is defined based on velocity profile
- `solve_ivp` (BDF) numerically integrates equations over time from initial condition
- Derivatives are approximated using finite central differences with `numpy.gradient`

Solving with physics-informed neural networks (PINNs) (Fig. 3)

- Based on PINNs-TF2 library (TensorFlow 2) customized to run on DirectML
- Initial/boundary conditions are randomly sampled from the data (50 IC, 100 BC)
- Neural network weights are adjusted (it "learns") by minimizing mean square error loss calculated from data and PDEs [2]
- 6 layers: 1 input, 5 dense; 30,802 params
- Diminishing returns over 16k epochs (fig. 4)

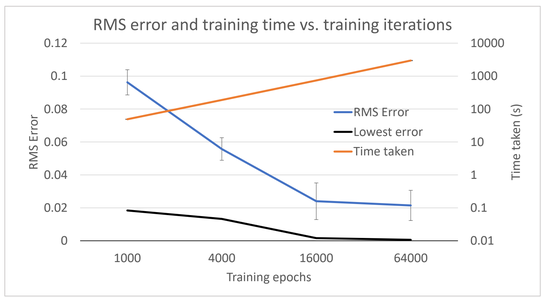


Fig. 4: NN output root mean square error (lower is better) and training time based on number of training epochs. Error bars represent 95% CI from 5 trials

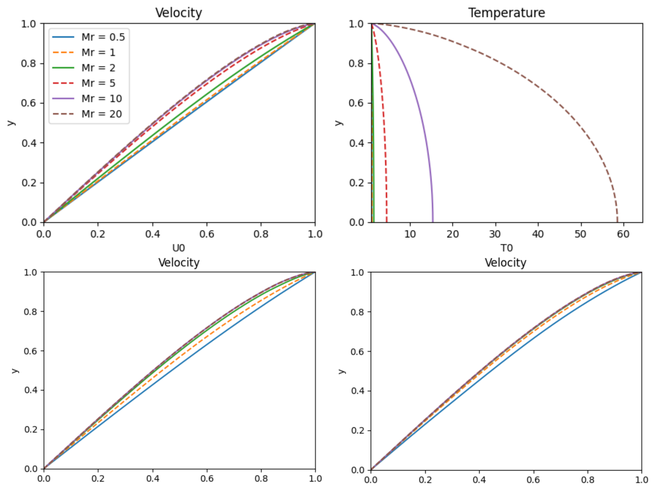


Fig. 1 Top: Steady state velocity and temperature profiles in air (Pr = 0.72, γ = 1.4)
Bottom: Steady state velocity profile with 10x air values for Pr (left) and γ (right)
Calculated for Mach 0.5 (subsonic), 1 (transonic), 2 (supersonic), and 5+ (hypersonic), showing convergence in the velocity profile at Mach 10+

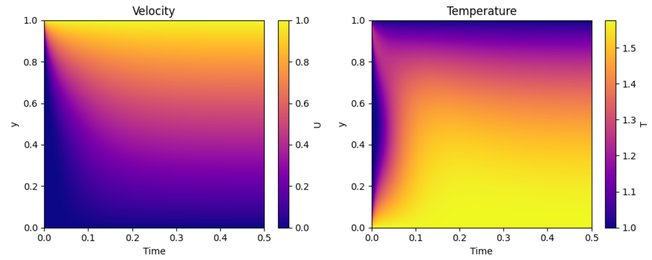


Fig. 2: Exact solutions for startup velocity and temperature profiles at Mach 2
The fluid is initially only moving at the boundary. As time increases, the rest of the fluid accelerates and approaches the steady state profile.

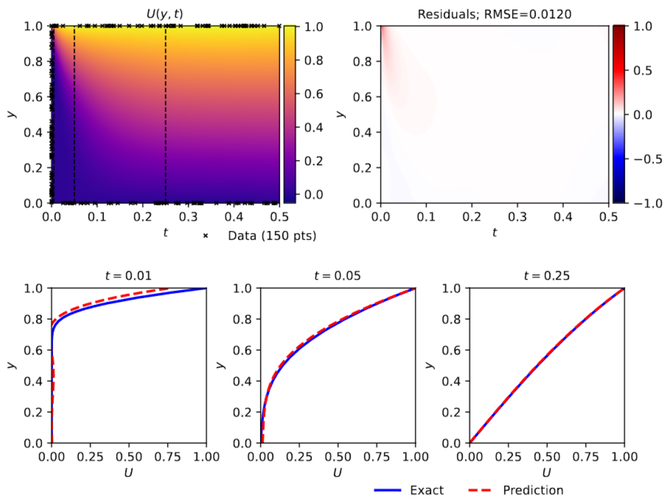


Fig. 3: PINN velocity output after 16000 training epochs for Mach 2
50 initial condition and 100 boundary condition samples were used along with 20000 mesh samples.

Conclusions and observations

Numerical simulation

- Higher Mach numbers cause the steady state to deviate more from *U* = *y* up to a point
  - At Mach 10+ temperature continues to increase while *U* does not
- Increasing Pr and γ produce graphs similar to increasing Mach number (Fig. 1 bottom), also significantly increase temperature
  - Also takes more time to reach steady state
  - High Mach and γ values cause oscillation and instability during integration

PINNs

- Excluding training, PINNs are much faster than exact simulation
- More accurate extrapolation beyond training data than regular NNs due to embedded PDEs
- Lower accuracy compared to training data especially with rapid change (Fig. 3 top right)
- Training is more computationally expensive than numerical simulation
- No significant improvement in accuracy from 64k epochs compared to 16k (Fig. 4)

Acknowledgements

I would like to express my gratitude to Prof. Chang Liu for his guidance, insightful feedback, and for sharing his extensive knowledge and experience in fluid dynamics.

References

[1] Bhattacharjee D., Mushtaq T., Seiler P., Hemati M. Structured Input-Output Analysis of Compressible Plane Couette Flow. AIAA SciTech 2023 Forum. 2023 Jan 23-27.
[2] Raissi M., Perdikaris P., Karniadakis G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics. 2019 Feb 1;378:686-707.

All code can be found on my GitHub repo:
https://github.com/huj31415/couette