CS230

Shock Parameterization for the Compressible Euler Equations using Deep Learning

Matthew Bonanni, Brett Bornhoft, and Ali Lasemi
Department of Mechanical Engineering
Stanford University
mbonanni@stanford.edu, bornhoft@stanford.edu, alasemi@stanford.edu

1 Introduction

The solution of the compressible Euler equations can be utilized in the design of high speed aerospace vehicles including supersonic combusting ramjets (scramjets) and low orbital re-entry capsules. The compressible Euler equations are partial differential equations that model continuum fluid flow without viscous effects. Assuming a calorically perfect gas, the equations are written as

$$\partial_t \begin{bmatrix} \rho \\ \rho \boldsymbol{u} \\ \rho E \end{bmatrix} + \nabla \cdot \begin{bmatrix} \rho \boldsymbol{u} \\ \rho \boldsymbol{u} \otimes \boldsymbol{u} + P\mathbb{I} \\ \boldsymbol{u}(\rho E + P) \end{bmatrix} = 0, \tag{1}$$

where ρ is the fluid density, $\boldsymbol{u}$ is the velocity vector, P is the pressure, $\mathbb{I}$ is the identity matrix, and $E = R_g/(\gamma - 1)T + \frac{1}{2}|\boldsymbol{u}|^2$ is the total energy per unit mass (with R_g denoting the specific gas constant, γ the specific heat ratio, and T the temperature). Pressure is computed from the ideal gas law, $P = \rho R_g T$.

The design of high-speed aerospace vehicles is enhanced through the use of computational fluid dynamics (CFD). CFD allows for the numerical simulation of the physical flow around the geometric representation of these systems. In our examples listed above, each has the common inclusion of shocks, or regions of compressed air that result in sharp gradients of the physical states on either side of the shock. Shocks can lead to large heat fluxes on vehicle bodies and have led to catastrophic failures in several situations.

Numerically "capturing" or resolving shocks is a current research field. Shocks within CFD solvers result in oscillatory behaviors that can quickly lead to numerical instabilites and failure. Common methods for addressing these issues include limiting the slopes across computational elements to remove numerical oscillations from the solution [1] or introducing artificial viscosity (or numerical diffusion) to stabilize the solution [2].

In this work, we propose to "capture" the shock (i.e. parameterize the shock) utilizing a deep neural network. The following proposal outlines two primary test problems that range in complexity. The first problem focuses on the analytical solution to a shock over a wedge that can be parameterized by its angle relative to the incoming flow vector. The second problem increases the complexity of the system by introducing single shocks around arbitrary bodies. These two problems are discussed at length with proposed methodology and datasets.

CS230: Deep Learning, Autumn 2021, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

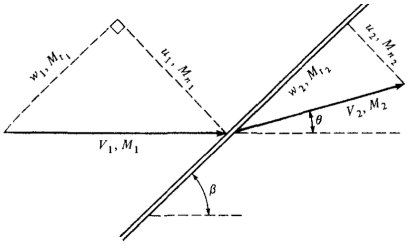
2 Data and Pre-Processing

2.1 Problem 1: Training a Neural Net to Capture Shocks on Wedges (An Analytical Exercise)

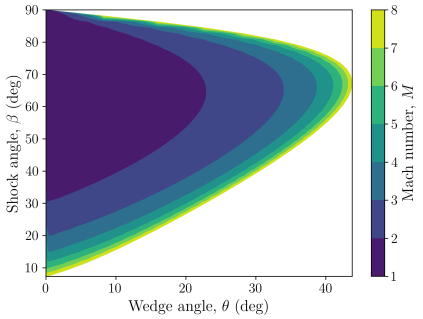
As an initial exercise for our ML model, we consider the supersonic flow over a wedge, as depicted in Figure 1a. This configuration results in a single, linear, oblique shock beginning at the leading edge of the wedge. The angle of this resulting shock, β, is a quantity of engineering significance. It is therefore of interest to predict this angle. In this specific configuration, an analytical relation known as the $\theta - \beta - M$ equation can be derived from the Rankine-Hugoniot jump conditions, yielding [3]:

$$\tan\theta = 2\cot\beta\left(\frac{M_1^2\sin^2\beta - 1}{M_1^2\left(\gamma + 2\cos 2\beta\right) + 2}\right) \tag{2}$$

The existence of this analytical solution is convenient for the generation of data in this first problem. The inputs to this problem are M_1, θ, and γ, and the output is β. A large table of this data is generated using Equation 2. A sample of this data for $\gamma = 1.4$ is presented in Figure 1b. Note that there are two solutions for β for each $\theta - M - \gamma$ combination. These represent the "strong" and "weak" shock solutions. In reality, for the external flows considered here, strong shocks rarely appear, so we only consider the weak solutions to make β single-valued. This results in a dataset with 70,921 cases.



(a) Diagram of supersonic flow over wedge with oblique shock, from [3].



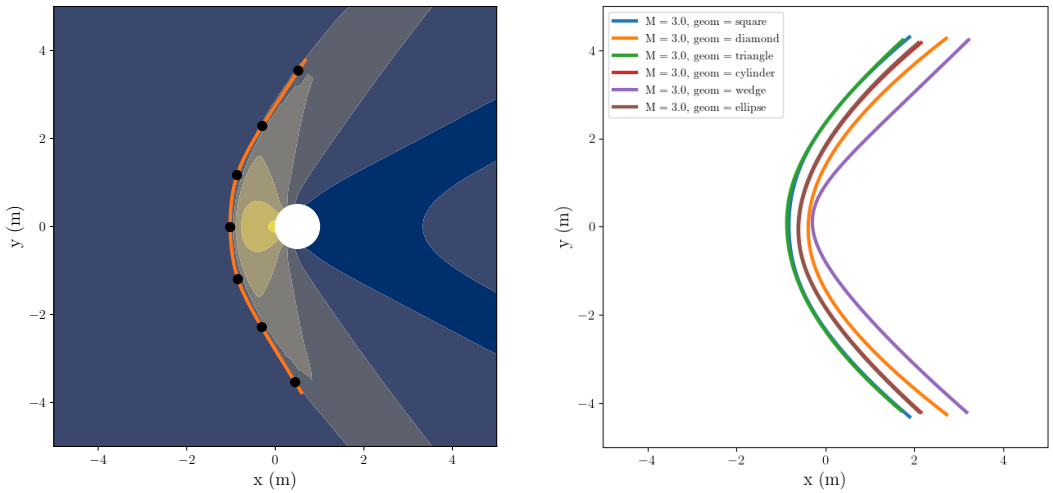
(b) Sample of $\theta - \beta - M$ data generated for $\gamma = 1.4$.

Figure 1: Description of the wedge configuration.

2.2 Problem 2: Arbitrary Shock Capturing for External High-Speed Flows

The second portion of this project builds upon Problem 1 by extending our neural network to more practical applications for which there are no analytical solutions. Furthermore, the network must be trained on data for which the shock is curved, and thus cannot be represented by just an angle. To generate the training/testing set, the high-speed flow over various blunt bodies is simulated using the well established/documented CFD solver SU2 [4]. In this problem, we are particularly interested in the model's ability to predict the shock for geometries which it has never seen before, lending it significant utility in application. Therefore, we want to include in our training set geometries which are significantly different from each other, such that they span the space of shock behavior. To this end, 5 different geometries are selected: a square, a diamond, a wedge (point leading), a triangle (point trailing), and a cylinder. An additional geometry, the ellipse, is selected for a test geometry which does not appear in the training data. The Mach number M_∞ is another input. The body is represented as a set of points, $\tilde{x}_{geom}$, which represent the geometry of the aerodynamic shape.

The training data itself requires some significant preprocessing. In Figure 2a, we illustrate an example of preprocessed data. First, the CFD solver is run for the given geometry and flow conditions. The results of the pressure field are then run through a simple script which identifies several points in the domain where a shock exists, by checking for a jump in the value of pressure moving from left to right. Finally, the points marked as containing a shock will be fit to a polynomial using Lagrange interpolation. It is important to note that even for the same M_∞, the different geometries have significantly different shock locations and shapes, as demonstrated by Figure 2b. The coefficients of the polynomial (which are the locations of the shock points) as well as $\tilde{x}_{geom}$ and M_∞ are used as the training data for each generated case. Due to the computational cost of the CFD simulations, the dataset for this problem is much smaller, with only 310 cases, not including the ellipse cases.



(a) Preprocessed simulation data, showing pressure contours and the polynomial fit to the shock location.

(b) Shocks for the 6 different blunt body geometries at $M_\infty = 3.0$.

Figure 2: Depictions of the data for problem 2.

3 Methods

3.1 Problem 1: Training a Neural Net to Capture Shocks on Wedges (An Analytical Exercise)

Using the Keras library [5], a deep neural network is used to predict β from the input data. Since this is a regression problem, a linear activation is used for the output layer. Two approaches are attempted: a simple linear regression, and a deep neural network. The first network (denoted Linear Model) utilizes a single neuron with a linear activation function. The second network (denoted Deep Model), uses 5 hidden layers each with 20 neurons. Each hidden layer in the Deep Model is led by a batch normalization and features a ReLU activation function. We use 90% of our data for our training and validation sets (81% training, 9% validation), and the remaining 10% for the test set.

3.2 Problem 2: Arbitrary Shock Capturing for External High-Speed Flows

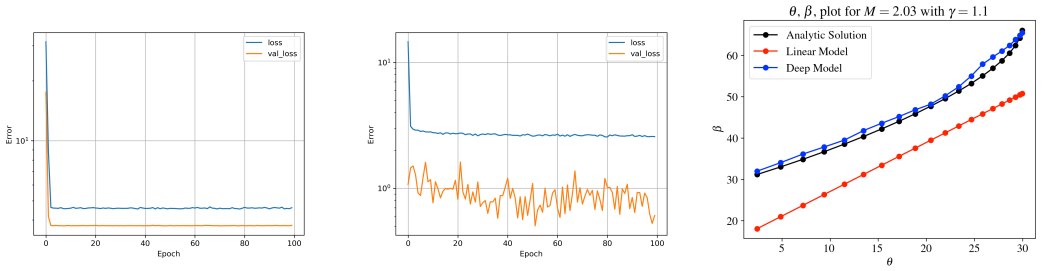
A deep neural network was used to locate the shocks. Since there are not previous projects on this application, we performed a hyperparameter search, and settled on a neural network with 7 hidden layers, each with 50 ReLU units. The boundaries of each of the blunt body geometries was described by 12 points, each with x and y coordinates. Additionally, we take the Mach number as an input. Therefore, the model has 25 total inputs. Since we are using polynomial fits of order 4, we have 5 points parameterizing each shock, each with x and y coordinates. Therefore, the model has 10 total outputs. (In the code, however, the number of inputs and outputs is detected from the data.) The network features batch normalization after each layer, including the input layer, and uses mean squared error as its loss function. The model was trained for 500 epochs with a learning rate of 0.005 and a batch size of 32. Like in problem 1, we use 90% of our data for our training and validation sets (81% training, 9% validation), and the remaining 10% for the test set.

4 Results

4.1 Problem 1: Training a Neural Net to Capture Shocks on Wedges (An Analytical Exercise)

Here, we detail our preliminary results of Problem 1. For Problem 1, we compare the results between two networks. In Figure 3, we plot the loss functions generated for subsequent training and test data. In addition, we plot a specific Mach number and ratio of specific heats to compare the linear and deep models with the analytical solution. A reasonable metric for success is to be within 3° of the analytical solution. Given this metric, the success rate for the linear model on the training/test data is

$52.96\%/53.16\%$ respectively. In contract, the deep network results in a training/testing data success rate of $97.8\%/97.9\%$ respectively. This clearly shows the advantage of the deep network.



(a) Loss function vs epochs for Linear Model.

(b) Loss function vs epochs for Deep Model.

(c) Diagram of supersonic flow over wedge with oblique shock, from [3].

Figure 3: Results for Problem 1.

4.2 Problem 2: Arbitrary Shock Capturing for External High-Speed Flows

For problem 2, the loss decay from the training methodology described above is presented in Figure 4. For this problem, our success metric for a given case is that the mean distance between the predicted and true shock interpolation points must be less than 0.25 of the characteristic length of the body. Based on this metric, our neural network achieves 90.7% correctness on the training data, and 90.3% correctness on the test data. In this case, the test data comes from the same set of geometries as the training data. To measure the model's ability to predict shocks from geometries it has never seen before, we additionally test it on the ellipse. For this new geometry, the model achieves 86% correctness. A sample of correctly and incorrectly predicted shocks for both in-training-set and out-of-training-set geometries are presented in Figure 5.

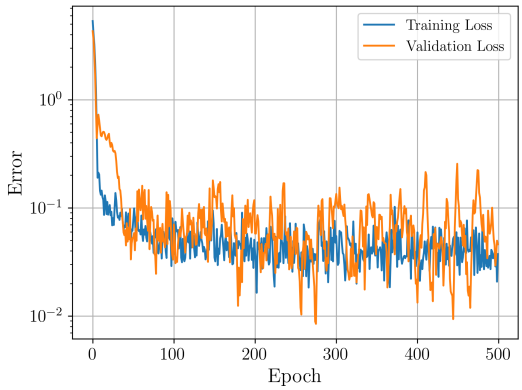
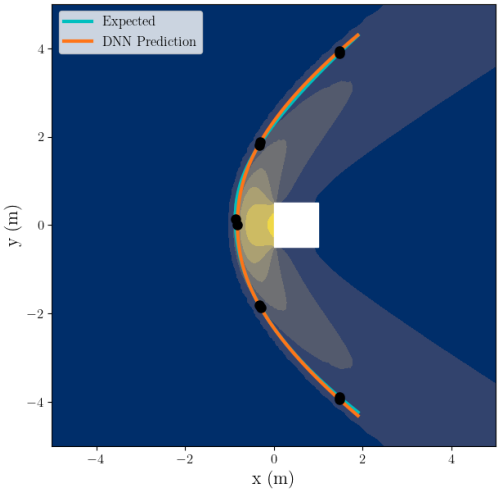


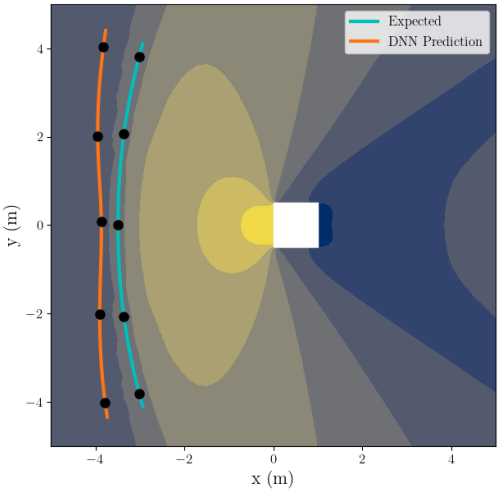
Figure 4: Loss function vs epochs for Problem 2.

5 Conclusions

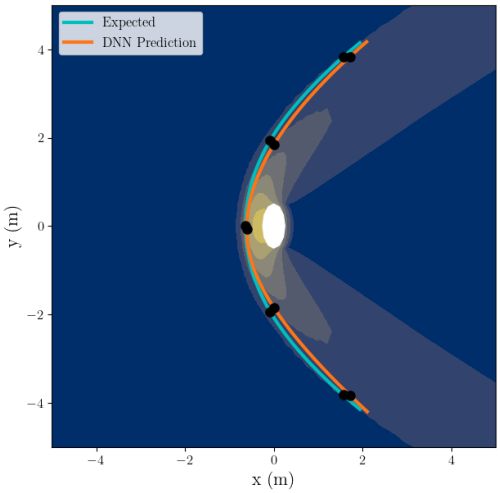
This project successfully demonstrated the ability of neural networks to predict shock locations in supersonic flows. In problem 1, the deep neural network successfully learned the $\theta - \beta - M$ equation. In problem 2, we extended to a configuration for which there is no analytical solution, with data generated by CFD simulations. Here, we demonstrated that not only can the deep neural network accurately predict the locations and shapes of shocks for known geometries, it can even do this for geometries it has never seen before. Notably, it appears that most of the model failures, for both in-training-set and out-of-training-set geometries, occur at cases with small M_∞. It is likely that this is a result of issues with the underlying data: for these cases, the shock is very diffuse in the CFD simulation, and the shock location extracted by the preprocessor is somewhat unreliable. Improved CFD simulations would address this issue.



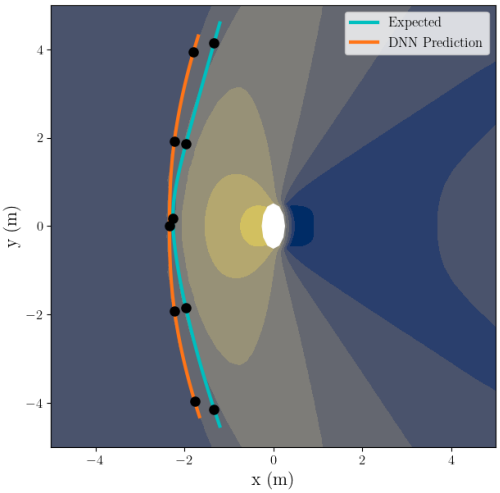
(a) Comparison for the square geometry at Mach 3, which was used to train the model. This prediction was deemed successful.



(b) Comparison for the square geometry at Mach 1.4, which was used to train the model. This prediction was deemed unsuccessful.



(c) Comparison for the ellipse geometry at Mach 3, which was not used to train the model. This prediction was deemed successful.



(d) Comparison for the ellipse geometry at Mach 1.5, which was not used to train the model. This prediction was deemed unsuccessful.

Figure 5: Comparison of expected results and neural network predictions.

6 Contributions

Matthew generated the data using the $\theta - \beta - M$ equation, and built the initial linear regression and neural networks for problems 1 and 2. Brett performed the CFD simulations for problem 2, and helped to tune and evaluate the neural network for problem 1. Ali developed the post-processing methodology for the problem 2 data, and helped to tune the neural network for problem 1 by refining the parameters of the dataset, and evaluated the model's performance on new geometries. All group members contributed to the final paper and presentation.

References

[1] A. Burbeau, P. Sagaut, and C.-H. Bruneau, "A problem-independent limiter for high-order runge–kutta discontinuous galerkin methods," *Journal of Computational Physics*, vol. 169, no. 1, pp. 111–150, 2001.

[2] E. J. Ching, Y. Lv, P. Gnoffo, M. Barnhardt, and M. Ihme, "Shock capturing for discontinuous Galerkin methods with application to predicting heat transfer in hypersonic flows," *Journal of Computational Physics*, vol. 376, pp. 54–75, 2019.

[3] J. D. Anderson, *Modern compressible flow: with historical perspective*, vol. 12. McGraw-Hill New York, 1990.

[4] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso, "Su2: An open-source suite for multiphysics simulation and design," *AIAA Journal*, vol. 54, no. 3, pp. 828–846, 2016.

[5] F. Chollet *et al.*, "Keras," 2015.