

Solving the Inviscid Burgers' Equation Numerically

Andre Gormann
agormann@sfu.ca

Ethan MacDonald
jem21@sfu.ca

Contents

1	Introduction	1
2	Theory	2
2.1	Finite Volume Methods for Conservation Laws	3
2.2	The REA Algorithm and Godunov's Method	4
2.3	The Riemann Problem and Managing Discontinuous Solutions	4
2.4	Convergence Notions	5
3	Experiments	6
3.1	'Easy' Problems	6
3.1.1	Riemann problem: shockwave	6
3.1.2	Riemann problem: rarefaction wave	8
3.2	'Hard' Problems	10
3.2.1	Square wave	10
3.2.2	Sine wave	10
3.2.3	Sine-squared wave	10
4	Physics Informed Neural Network Scheme	10
5	Conclusion	10
A	Appendix	11
A.1	Figures	11
A.1.1	'Hard' Problems	11
A.2	Code	13
A.3	Analytical Solution via the Method of Characteristics	16

1 Introduction

The 1-D inviscid Burgers' equation is a first-order hyperbolic partial differential equation (PDE) of the form

$$\partial_t u(x, t) + u(x, t) \partial_x u(x, t) = 0 \quad (1)$$

where $u \in C^1(\Omega)$, and $x, t \in \Omega \subset \mathbb{R} \times \mathbb{R}^+$. Note that a more compact form of (1) is $u_t + uu_x = 0$. The latter formulation is what is commonly seen in literature.

This equation has a brother named the *viscous* Burgers' equation (or simply referred to as Burgers' equation), which takes the form

$$u_t + uu_x = \epsilon u_{xx} \quad (2)$$

where $\epsilon > 0$ is the diffusion coefficient. We mention this because the inviscid Burgers' equation can be interpreted as resulting from letting $\epsilon \rightarrow 0$ in (2). This is important because it informs us what the 'correct' behavior of (1) should be.

The quasilinear equation (1) is not the only formulation of the inviscid Burgers' equation, and in a certain sense it is actually the 'wrong' one to study. This is because under a few reasonable assumptions, a completely smooth initial profile modeled by (1) will devolve into a discontinuous one in finite time. This is unsettling because then (1) fails to hold; the partial derivative of a discontinuous function does not exist!

Instead, we rewrite (1) as

$$u_t + f(u)_x = 0 \quad (3)$$

where

$$f(u) = \frac{1}{2}u^2 \quad (4)$$

is known as the flux function^[1]. This is known as the conservation form of the inviscid Burgers' equation. If we integrate (3) over $[a, b]$, where $[a, b] \subset \Omega$, then we get

$$\frac{d}{dt} \int_a^b u(x, t) dx = f(u(a, t)) - f(u(b, t)) \quad (5)$$

where we have exchanged differentiation and integration. The form (5) is known as the integral form of (3), and it is where the 'conservative' notion comes from^[2].

Importantly, this integral form has no problems admitting profiles u with spatial discontinuities (we assume it is not also discontinuous in time). It is this formulation (along with (3)) that we will be studying and developing our numerical schemes for, not the quasilinear form.

2 Theory

We will begin by introducing some notation that will be used throughout this section^[3]. Supposing that $\Omega = [a, b] \times \mathbb{R}^+$, we discretize the interval $[a, b]$ into a vector of N points x_j by defining a fixed mesh-width^[4] $\Delta x = (b - a)/N$ so that

$$x_j = a + j\Delta x. \quad (6)$$

Note that we are assuming periodic boundary conditions so that $x_0 = x_N$, and so we have exactly N points. For reasons that will soon become clear, we are also interested in the half-steps $x_{j\pm 1/2}$ defined by

$$x_{j\pm 1/2} = x_j \pm \frac{\Delta x}{2}. \quad (7)$$

Time will simply be denoted by t_n , with no explicit formula. This is because for stability purposes (see Section 2.4), we require a variable time-step.

^[1]Because the inviscid Burgers' equation can be put into this form, we are able to leverage a whole host of theory concerning advection conservation laws.

^[2]A more explicit derivation of (5) can be found in Section 2.1 of LeVeque, 2002.

^[3]The content of this entire section has been primarily adapted from LeVeque, 1992 and LeVeque, 2002.

^[4]It is possible to use a variable mesh-width, but we have elected not to.

We denote the *pointwise values* of the true solution u which solves (3) exactly at the mesh point (x_j, t_n) by

$$u_j^n = u(x_j, t_n). \quad (8)$$

The *cell averages* about the mesh point (x_j, t_n) are then defined by

$$\bar{u}_j^n = \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} u(x, t_n) dx. \quad (9)$$

2.1 Finite Volume Methods for Conservation Laws

For finite difference methods, at each time t_n , we are computing a vector $U^n \in \mathbb{R}^N$ where the j -th component U_j^n approximates the pointwise true solution u_j^n . In light of (5) though, it is perhaps more natural to instead view U_j^n as approximating the *cell average* \bar{u}_j^n . This gives rise to what is known as a *finite volume method*.

If we integrate (5) from time t_n to t_{n+1} , we get

$$\begin{aligned} \int_{x_{j-1/2}}^{x_{j+1/2}} u(x, t_{n+1}) dx &= \int_{x_{j-1/2}}^{x_{j+1/2}} u(x, t_n) dx \\ &\quad - \left[\int_{t_n}^{t_{n+1}} f(u(x_{j+1/2}, t)) dt - \int_{t_n}^{t_{n+1}} f(u(x_{j-1/2}, t)) dt \right]. \end{aligned} \quad (10)$$

Dividing by Δx and applying (9) yields

$$\bar{u}_j^{n+1} = \bar{u}_j^n - \frac{1}{\Delta x} \left[\int_{t_n}^{t_{n+1}} f(u(x_{j+1/2}, t)) dt - \int_{t_n}^{t_{n+1}} f(u(x_{j-1/2}, t)) dt \right]. \quad (11)$$

The goal of a successful finite volume method then is to accurately model the flux through the boundaries of each cell. Explicitly, we want to find some numerical flux function \mathcal{F} so that

$$\mathcal{F}(U_j^n, U_{j+1}^n) \approx \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} f(u(x_{j+1/2}, t)) dt \quad (12)$$

$$\mathcal{F}(U_{j-1}^n, U_j^n) \approx \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} f(u(x_{j-1/2}, t)) dt. \quad (13)$$

To this end, we say that a numerical method is in *conservation form* if

$$U_j^{n+1} = U_j^n - \frac{\Delta t}{\Delta x} [\mathcal{F}(U_j^n, U_{j+1}^n) - \mathcal{F}(U_{j-1}^n, U_j^n)]. \quad (14)$$

Note that while this derivation comes about through the introduction of control volumes, we can also understand it through the lens of finite differences. Specifically, from (3) we get the relation

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + \frac{F_{j+1/2}^n - F_{j-1/2}^n}{\Delta x} = 0 \quad (15)$$

where $F_{j\pm 1/2}^n \sim \mathcal{F}$ as before.

2.2 The REA Algorithm and Godunov's Method

The reconstruct-evolve-average (REA) algorithm is characterized by the following three-step process:

1. First we *reconstruct* a piecewise polynomial function $p(x, t_n)$ from the approximate cell averages U_j^n . Though we will be only considering piecewise linear polynomials, there is no explicit limit on the degree of these p .
2. Using the $p(x, t_n)$ as initial data, we then *evolve* the PDE until time t_{n+1} into the future.
3. Finally, we *average* the updated polynomial over each grid cell, obtaining new approximate cell averages

$$U_j^{n+1} = \frac{1}{\Delta x} \int_{x_{j-1/2}}^{x_{j+1/2}} p(x, t_{n+1}) dx. \quad (16)$$

2.3 The Riemann Problem and Managing Discontinuous Solutions

A Riemann problem is an initial value problem (IVP) for a conservation equation where the supplied initial profile consists of piecewise constant data with a single discontinuity. Typically, the discontinuity is located at $x = 0$, so that the initial data is of the form

$$u(x, 0) = \begin{cases} u_L & x < 0 \\ u_R & x \geq 0 \end{cases} \quad (17)$$

This problem is very important for the inviscid Burgers' equation specifically because for those points sufficiently away from the discontinuity, we can solve it exactly via the method of characteristics (see Appendix A.2). How we resolve those points near the discontinuity is a bit more complicated. One can appeal to the Lax entropy condition^[5] which states that, for the inviscid Burgers' equation, a discontinuity propagating with speed s is valid if $u_L > u_R$ (see Figure 1a). The exact speed of this discontinuity, referred to as a shock, is determined by the Rankine-Hugoniot condition^[6], which for the inviscid Burgers' equation is simply $s = (u_L + u_R)/2$. The solution to this problem is then

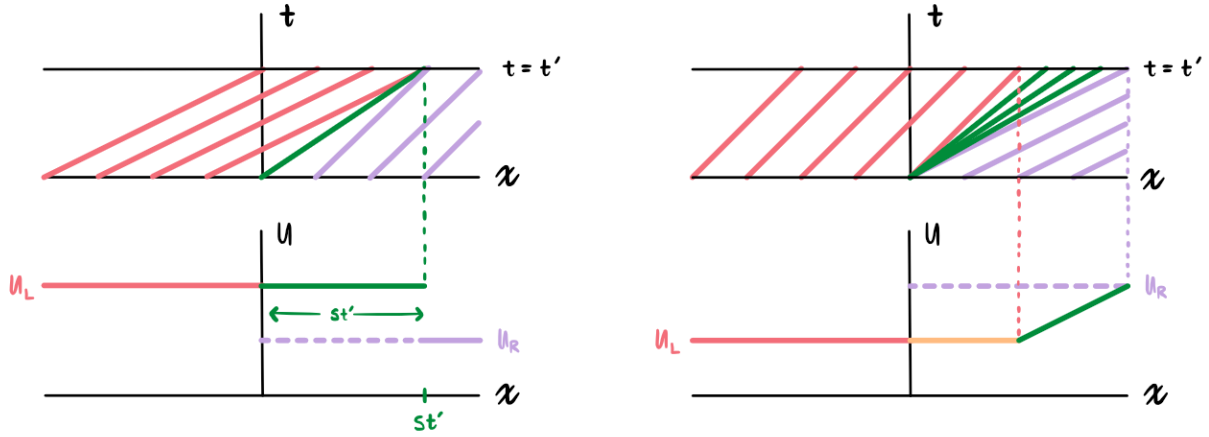
$$u(x, t) = \begin{cases} u_L & x < st \\ u_R & x \geq st \end{cases} \quad (18)$$

Alternatively, one may recall that (1) can be interpreted as the inviscid limit of Burgers' equation. By admitting a small, but nonzero, $\epsilon > 0$, one can study the behavior near a discontinuity. This is formally known as the vanishing-viscosity approach. By doing so experimentally, the Lax entropy condition is corroborated and the profile (18) is seen. Moreover, the vanishing-viscosity approach reveals the correct behavior when $u_L < u_R$; a rarefaction wave (see Figure 1b).

$$u(x, t) = \begin{cases} u_L & x < u_L t \\ x/t & u_L t \leq x \leq u_R t \\ u_R & x > u_R t \end{cases} \quad (19)$$

^[5]See LeVeque, 2002, Section 11.2 for more information.

^[6]See LeVeque, 2002, Section 11.8 for a derivation.



(a) Resolution of a Riemann problem when $u_L > u_R > 0$. A shockwave propagates to the right with speed $(u_L + u_R)/2$.

(b) Resolution of a Riemann problem when $0 < u_L < u_R$. A rarefaction wave fills the void left by the characteristics.

Figure 1: Resolving the discontinuity in a Riemann problem.

2.4 Convergence Notions

We wish to develop what it means for some approximate cell averages U_j^n to converge to a weak solution u of (5). To this end, we define a piecewise-constant function $U^{(\Delta t)}(x, t)^{[7]}$ by

$$U^{(\Delta t)}(x, t) = U_j^n \quad (x, t) \in [x_{j-1/2}, x_{j+1/2}) \times [t_n, t_{n+1}) \quad (20)$$

Now since weak solutions are not unique, we define the set

$$\mathcal{W} = \{u : u(x, t) \text{ is a weak solution to (5)}\} \quad (21)$$

Then the global error^[8] is given by

$$\text{dist} \left(U^{(\Delta t)}, \mathcal{W} \right) = \inf_{w \in \mathcal{W}} \|U^{(\Delta t)} - w\|_{1,T} \quad (22)$$

where

$$\|v\|_{1,T} = \int_0^T \|v(\cdot, t)\|_1 dt = \int_0^T \int_{\mathbb{R}} |v(x, t)| dx dt \quad (23)$$

Lastly, we define the total variation of a collection of cell averages U^n by

$$\text{TV}(U^n) = \sum_{j=1}^N |U_j^n - U_{j-1}^n| \quad (24)$$

We are now ready to state the convergence result.

^[7]It is unclear what is gained by this definition as by construction $U^{(\Delta t)}$ agrees with U_j^n .

^[8]The $w \in \mathcal{W}$ feels like a typo. Due to our unfamiliarity with the material though, we will leave it as printed in LeVeque, 2002, p. 246.

Theorem 1 Suppose $U^{(\Delta t)}$ is generated by a numerical method in conservation form with a Lipschitz continuous numerical flux, consistent with some scalar conservation law. If the total variation $TV(U^n)$ is uniformly bounded for all n , Δt with $\Delta t < \Delta t_0$, $n\Delta t \leq T$, then the method is convergent in the sense that the global error $\text{dist}(U^{(\Delta t)}, \mathcal{W}) \rightarrow 0$ as $\Delta t \rightarrow 0$.

In the paper LeVeque and Temple, 1985, the necessary conditions are proven to hold for the Godunov method, and so we have formal verification that our method is convergent^[9]. One of the assumptions made though is that either of the following CFL conditions are satisfied

$$\max_j |U_j^n| \frac{\Delta t_n}{\Delta x} \leq \frac{1}{2} \quad \text{or} \quad \max_j |U_j^n| \frac{\Delta t_n}{\Delta x} \leq 1 \quad (25)$$

The latter has the property that solutions to neighboring Riemann problems may interact. LeVeque and Temple, 1985, argues that this is not a problem though, and experimentally we have verified this (see Section 3).

3 Experiments

We will be verifying our theory experimentally on five different problems. The first two are Riemann problems which are ‘easy’ in the sense that a MATH student familiar with the relevant material should be able to characterize the solution completely. The consequence of this is that we can hard-code the solutions to compare against with respect to accuracy, both for our methods and for the Physics Informed Neural Network (PINN).

In contrast, we are unable to resolve the ‘hard’ problems analytically and so cannot directly compute the accuracy of our methods. Instead, we will have to employ numerical convergence studies and rely on the soundness of the underlying theory.

3.1 ‘Easy’ Problems

3.1.1 Riemann problem: shockwave

The initial value problem (IVP) of interest is

$$\begin{cases} u_t + uu_x = 0 & x \in (-\pi, \pi), t > 0 \\ u(-\pi, t) = 1 \\ u(\pi, t) = 0 \\ u(x, 0) = u_0(x) \end{cases} \quad (26)$$

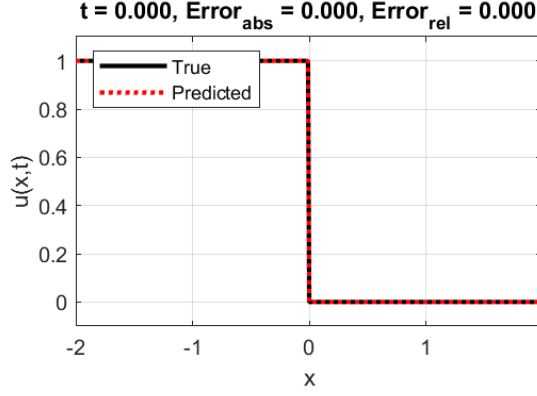
where

$$u_0(x) = \begin{cases} 1 & x < 0 \\ 0 & x \geq 0 \end{cases} \quad (27)$$

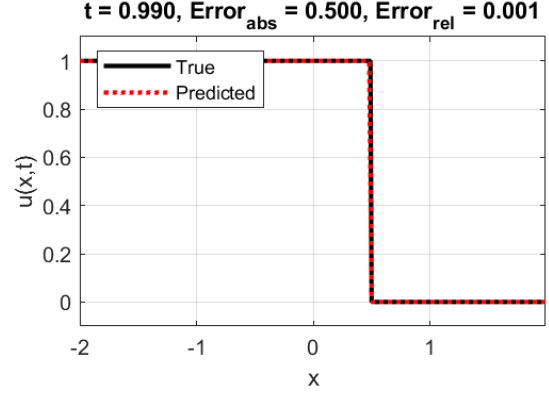
Following our prior discussion, the analytical solution is characterized by a shockwave, and is given by

$$u(x, t) = \begin{cases} 1 & x < t/2 \\ 0 & x \geq t/2 \end{cases} \quad (28)$$

^[9]While we would have liked to prove some of this ourselves (or at the very least explain the result more thoroughly), the material is simply too advanced for us.

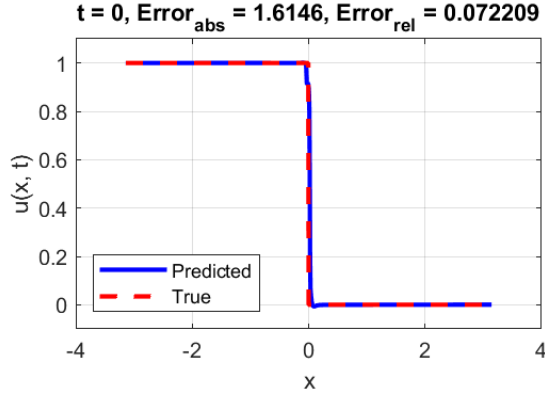


(a) Initial state for the Riemann Problem.

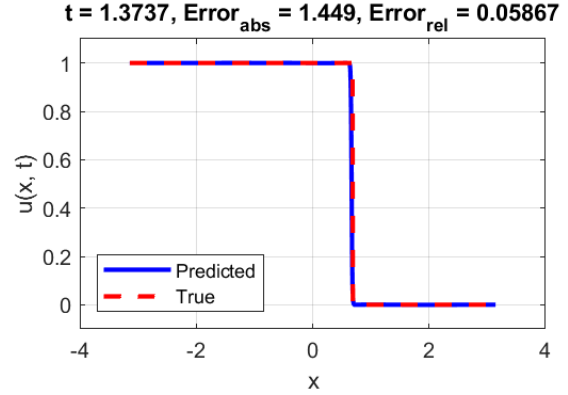


(b) As t increases, the error is small and consistent.

Figure 2: The numerical solution of the shockwave Riemann problem.



(a) Initial state. Notice that the neural network cannot achieve perfect accuracy at $t = 0$.

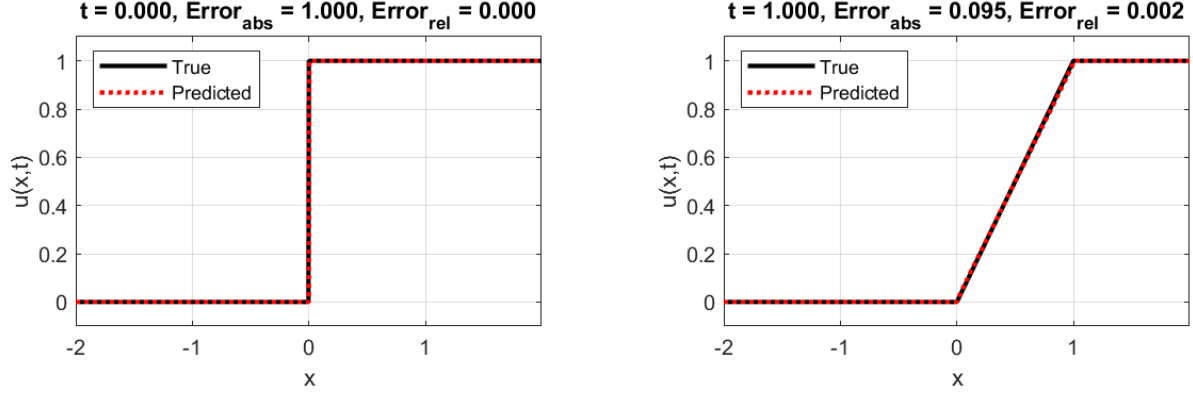


(b) As t increases, the error is larger than in the numerical solution, but still consistent.

Figure 3: The PINN of the shockwave Riemann problem.

For this problem, we attempted both a numerical approach and a PINN approach. The numerical solution was highly accurate and required little time to compute even when using 1000 grid points. Some plots are shown below.

When using the PINN approach, we achieved decent results. However, these results are far worse than the numerical solution and took much longer to compute as they required training a neural network. Our neural network used 9 layers with 20 neurons at each layer, as well as 10 000 internal collocation points during training. The code used for the PINN approach in this problem and the next were adapted from the MathWorks example “Solve PDE Using Physics-Informed Neural Network.” Some plots are shown below.



(a) Initial state of the Riemann problem.

(b) idk lol

Figure 4: The numerical solution of the rarefaction wave Riemann problem.

3.1.2 Riemann problem: rarefaction wave

We modify the (29) slightly so that its solution exhibits a rarefaction fan

$$\begin{cases} u_t + uu_x = 0 & x \in (-\pi, \pi), t > 0 \\ u(-\pi, t) = 0 \\ u(\pi, t) = 1 \\ u(x, 0) = u_0(x) \end{cases} \quad (29)$$

where

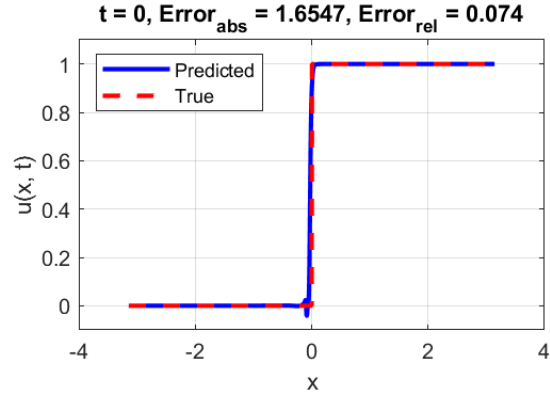
$$u_0(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \quad (30)$$

Instead of a shockwave, the analytical solution is characterized by a rarefaction wave, and is given by

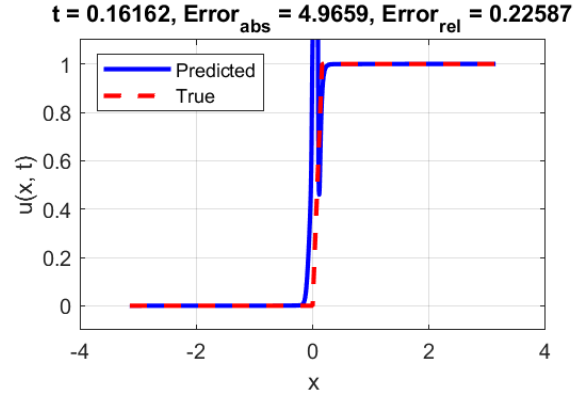
$$u(x, t) = \begin{cases} 0 & x < 0 \\ x/t & 0 \leq x \leq t \\ 1 & x > t \end{cases} \quad (31)$$

Once again, we attempted both a numerical approach and a PINN approach. Once again, the numerical approach was fast, accurate, and consistent. Some plots are shown below.

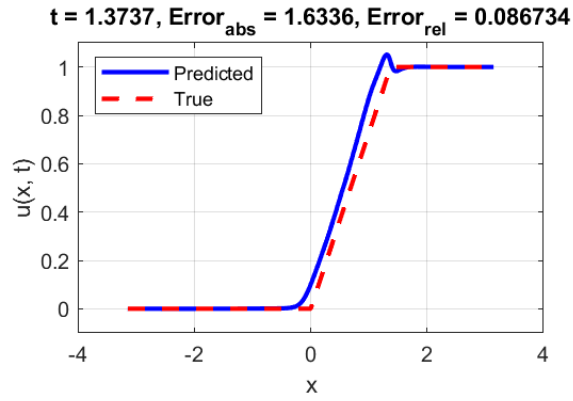
When using a PINN approach, the solution is somewhat oscillatory at the steep gradient around $t = 0$. However, as time progresses, this oscillation seems to dissipate. We presume that the neural network struggles with the steeper gradients near $t = 0$ because the gradient of u is computed as a part of the loss during training. Naturally, this becomes less of a problem as t increases because the gradient becomes more shallow and changes at a slower rate. The PINN solution can also be seen lagging behind the true solution. Presumably, this is a result of the changing gradient and would no longer be an issue for $t \rightarrow \infty$. Some plots are shown below.



(a) Initial state. Notice that a small oscillation is already being exhibited in u_L .



(b) The oscillation becomes extreme as the gradient changes rapidly.



(c) The oscillation subsides and the PINN solution lags behind the true solution.

Figure 5: The PINN solution of the rarefaction wave Riemann problem.

3.2 ‘Hard’ Problems

3.2.1 Square wave

Our first IVP has the form

$$\begin{cases} u_t + uu_x = 0, & x \in [0, 2\pi], t > 0 \\ u(x, 0) = u_0(x) \\ u(0, t) = u(2\pi, t), & t > 0 \end{cases} \quad (32)$$

where

$$u_0(x) = \begin{cases} 1 & x \in [\pi/2, 3\pi/2] \\ 0 & \text{otherwise} \end{cases} \quad (33)$$

3.2.2 Sine wave

The IVP is given by

$$\begin{cases} u_t + uu_x = 0, & x \in [0, 2\pi], t > 0 \\ u(x, 0) = \sin(x) \\ u(0, t) = u(2\pi, t), & t > 0 \end{cases} \quad (34)$$

3.2.3 Sine-squared wave

We modify the prior IVP slightly

$$\begin{cases} u_t + uu_x = 0, & x \in [0, 2\pi], t > 0 \\ u(x, 0) = \sin^2(x) \\ u(0, t) = u(2\pi, t), & t > 0 \end{cases} \quad (35)$$

4 Physics Informed Neural Network Scheme

While a PINN approach worked decently for

5 Conclusion

this shit was way harder than i thought it would be

A Appendix

A.1 Figures

A.1.1 'Hard' Problems

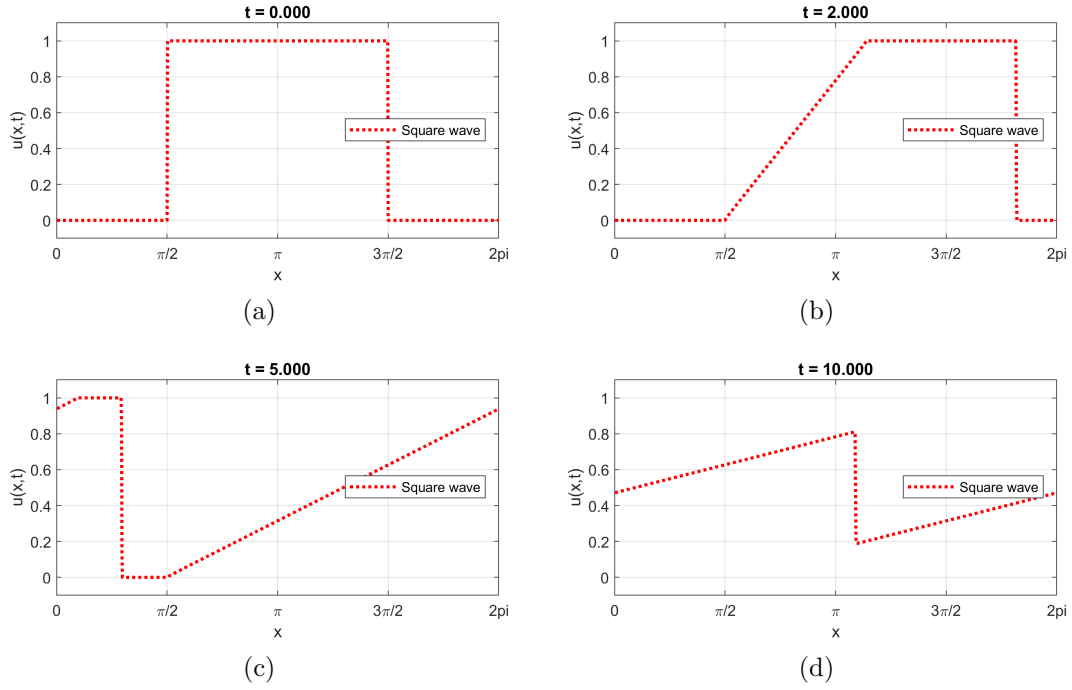


Figure 6: Numerical solutions to problem (32). Snapshots are taken at times $t = 0, 2, 5, 10$.

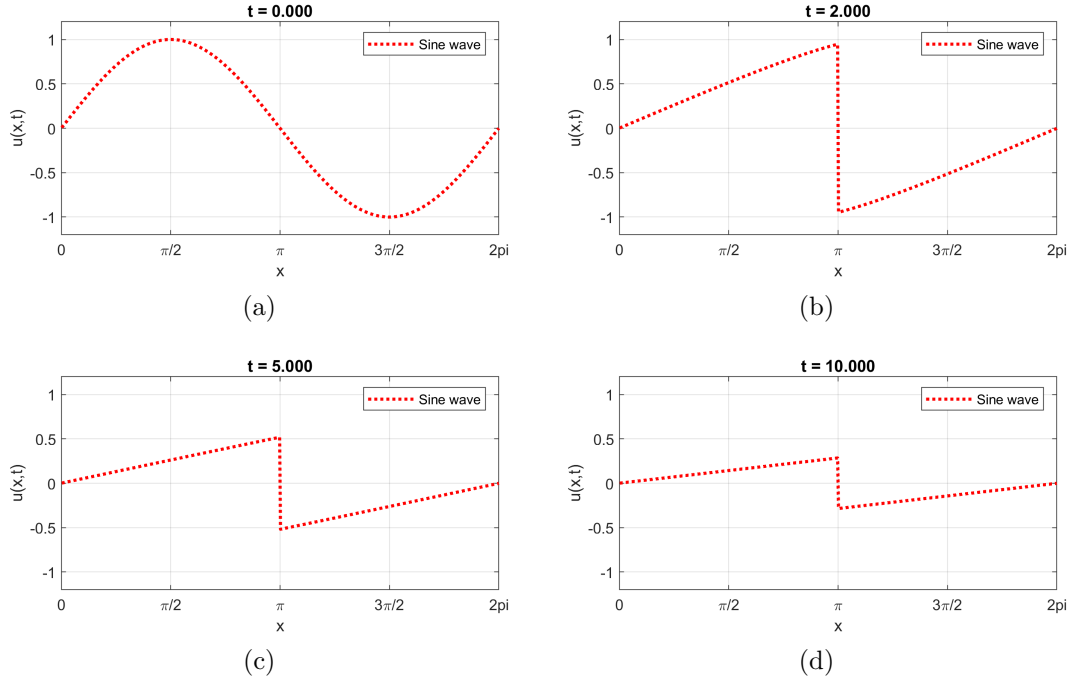


Figure 7: Numerical solutions to problem (34). Snapshots are taken at times $t = 0, 2, 5, 10$.

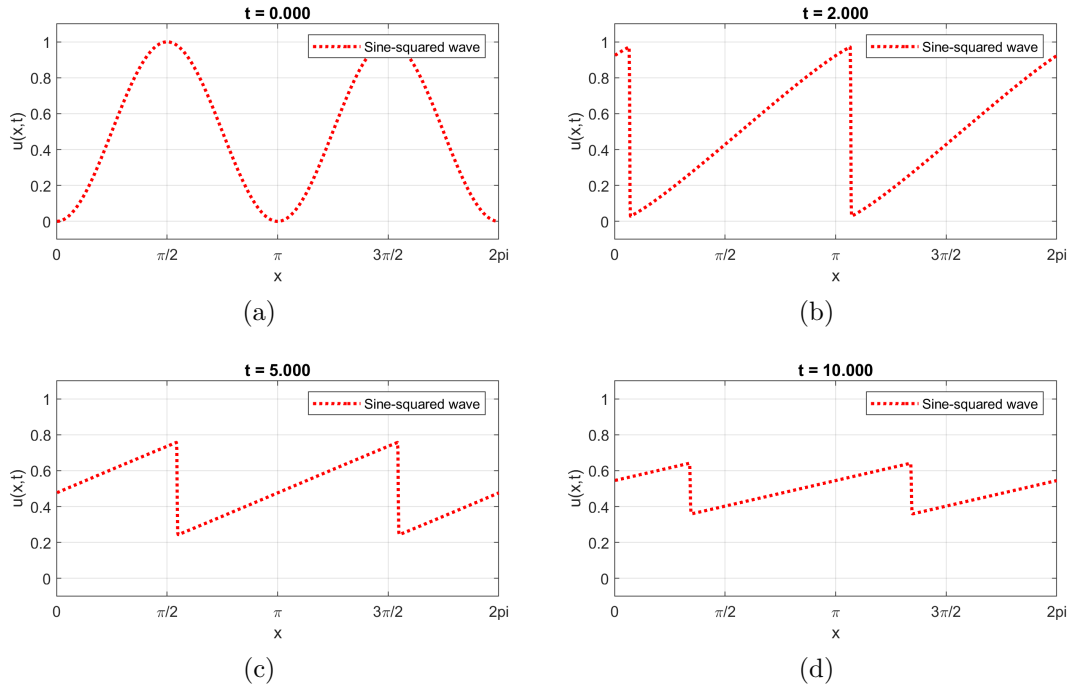


Figure 8: Numerical solutions to problem (35). Snapshots are taken at times $t = 0, 2, 5, 10$.

A.2 Code

All project code can be found on our GitHub page: <https://github.com/agormann/MACM416-project>.

```
1 function [T,U] = godunov(X,u0,tf)
2 %GODUNOV Solves the inviscid burgers equation via the finite volume
   method.
3 % Input
4 %   X := grid of x values
5 %   u0 := initial profile
6 %   tf := final time
7 % Output
8 %   T := time-steps that the problem was solved at
9 %   U := profiles at times t in T
10
11 % Setup
12 f = @(u) 0.5*u.^2; % flux function
13 nx = length(X);
14 dx = abs(X(1)-X(2));
15 u = u0;
16 t = 0;
17 U = u;
18 T = t;
19
20 % Solution
21 % dt = dx/max(abs(u)); % fix dt for mesh refinement
22 while t < tf
23     dt = dx/max(abs(u)); % compute dt (dynamic)
24     flux = f(u); % initial flux
25     F = zeros(size(u)); % flux update
26     % solving riemannn problem exactly
27     for i = 1:nx
28         uL = u(i);
29         fuL = flux(i);
30         uR = u(mod(i,nx)+1);
31         fuR = flux(mod(i,nx)+1);
32         if uL <= uR
33             F(i) = min(fuL,fuR);
34         elseif uL > uR
35             F(i) = max(fuL,fuR);
36         else
37             error('OOPSIE WOOPSIE!!')
38         end
39     end
40     u = u - dt/dx*(F - circshift(F,1)); % conservative update
41     t = t + dt; U = [U; u]; T = [T; t];
42 end
43 end
```

```

1 function [T,U] = godunov_dirichlet(X,u0,tf)
2 %GODUNOV_DIRICHLET Solves the inviscid burgers equation via the
   finite volume method.
3 % Dirichlet boundary conditions are assumed.
4 % Input
5 %   X := grid of x values
6 %   u0 := initial profile
7 %   tf := final time
8 % Output
9 %   T := time-steps that the problem was solved at
10 %   U := profiles at times t in T
11
12 % Setup
13 f = @(u) 0.5*u.^2; % flux function
14 nx = length(X);
15 dx = abs(X(1)-X(2));
16 u = u0;
17 t = 0;
18 U = u;
19 T = t;
20
21 % Solution
22 while t < tf
23     dt = dx/max(abs(u)); % compute dt (dynamic)
24     flux = f(u); % initial flux
25     F = zeros(size(u)); % flux update
26     % solving riemann problem exactly
27     for i = 1:nx-1
28         uL = u(i);
29         fuL = flux(i);
30         uR = u(mod(i,nx)+1);
31         fuR = flux(mod(i,nx)+1);
32         if uL <= uR
33             F(i) = min(fuL,fuR);
34         elseif uL > uR
35             F(i) = max(fuL,fuR);
36         else
37             error('OOPSIE WOOPSIE!!')
38         end
39     end
40     for i = 2:nx-1
41         u(i) = u(i) - dt/dx*(F(i)-F(i-1)); % conservative update
42     end
43     t = t + dt; U = [U; u]; T = [T; t];
44 end
45 end

```

```

1 function U = riemann(uL,uR,X,T)
2 %RIEMANN Solves a riemann problem exactly.
3 % Input
4 %   uL := value of profile to the left of 0
5 %   uR := value of profile to the right of 0
6 %   X := grid of x values
7 %   T := specific times to solve the problem at
8 % Output
9 %   U := matrix whose rows are the solution to the riemann problem
   at times specified in T.
10
11 nx = length(X);
12 u0 = uL*(X<0) + uR*(X>=0);
13
14 U = [u0; zeros(length(T)-1,nx)];
15
16 if uL > uR
17     % shockwave
18     s = (uL+uR)/2; % rankine-hugoniot condition
19     for i = 2:length(T)
20         t = T(i);
21         U(i,:) = uL*(X<s*t) + uR*(X>=s*t);
22     end
23 elseif uL < uR
24     % rarefaction wave
25     for i = 2:length(T)
26         t = T(i);
27         U(i,:) = uL*(X<uL*t) + (X/t).*((uL*t<=X)&(X<=uR*t)) + uR*(X>
           uR*t);
28     end
29 end
30
31 end

```

A.3 Analytical Solution via the Method of Characteristics

We wish to solve the 1-D inviscid Burgers' equation analytically.

Given data on some curve $\Gamma \subset \overline{\Omega}$, we are looking specific parametric curves $(x(t), t)$ which connect points $(x, t) \in \Omega$ to Γ . We want these curves to be precisely those which are parallel to the vector $(u, 1)$, that is

$$\frac{dx}{dt} = \frac{u(x(t), t)}{1} = u(x(t), t)$$

Now supposing that u solves (2), let $z(t)$ denote the value of u along a characteristic, i.e.

$$z(t) = u(x(t), t)$$

Then by the chain rule

$$\frac{dz}{dt} = \partial_x u(x(t), t) \frac{dx}{dt} u(x(t), t) + \partial_t u(x(t), t)$$

but $x'(t) = u(x, t)$, so

$$\frac{dz}{dt} = \partial_t u(x(t), t) + u(x, t) \partial_x u(x(t), t)$$

which is precisely 0 by (2). Hence, we have the following coupled system of ODEs

$$\begin{cases} x'(t) = z(t) = u(x(t), t) \\ z'(t) = 0 \end{cases} \quad (36)$$

Integrating the second term, we get that

$$z(t) = z_0$$

for some $z_0 \in \mathbb{R}$. But $z(t) = u(x(t), t)$, so then $u(x(t), t) = z_0$. This corroborates our findings with (3). Now by integrating the first term, we get

$$x(t) = z_0 t + x_0 \quad (37)$$

where $x_0 \in \mathbb{R}$. Evaluating at $t = 0$, we have that $x(0) = x_0$. Now assuming we are prescribed some initial condition $u(x, 0) = g(x)$, we have that (5) becomes

$$x(t) = g(x_0)t + x_0 \quad (38)$$

which are exactly those characteristic curves we initially sought.

References

- Choksi, R. (2022). *Partial differential equations: A first course*. American Mathematical Society.
- Iserles, A. (2009). *A first course in the numerical analysis of differential equations*. Cambridge University Press.
- LeVeque, R. J. (1992). *Numerical methods for conservation laws*. Birkhauser.
- LeVeque, R. J. (2002). *Finite volume methods for hyperbolic problems*. Cambridge University Press.
- LeVeque, R. J., & Temple, B. (1985). Stability of godunov's method for a class of 2x2 systems of conservation laws. *Transactions of the American Mathematical Society*, 288(1), 115–123.