Solving the Viscous Burgers' Equation with Mimetic Differences

Michael Sanders and Miguel A. Dumett

July 17, 2023

Publication Number: CSRCR2023-02

Computational Science &
Engineering Faculty and Students
Research Articles

Database Powered by the
Computational Science Research Center
Computing Group & Visualization Lab

COMPUTATIONAL SCIENCE & ENGINEERING

SAN DIEGO STATE UNIVERSITY

Computational Science Research Center
College of Sciences
5500 Campanile Drive
San Diego, CA 92182-1245
(619) 594-3430

Solving the Viscous Burgers' Equation with Mimetic Differences

Michael Sanders* and Miguel A. Dumett †‡

July 17, 2023

Abstract

In this paper, the Burgers' equation is solved using mimetic differences.

1 Introduction

The viscous Burgers' equation is a nonlinear PDE governing advective-diffusive flows. In two spatial dimensions with $u = u(x, y, t)$ and $v = v(x, y, t)$, the system is [2]

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{1}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{2}$$

In this equation, the only parameter is ν, which is the kinematic viscosity of the flow. The kinematic viscosity can loosely be thought of as the "stickiness" of the fluid, or its internal resistance to flow; for example, honey has a higher kinematic viscosity than water. The Reynolds number, which quantifies the ratio of a fluid's inertial forces to its viscous forces, is directly related to ν via the relation $Re = \frac{1}{\nu}$. The Reynolds number is used in the analytic solution to the equation.

In this paper, the MOLE library [1] of mimetic differences is used to estimate the solution to the viscous Burgers' equation. The numerical results are compared to the analytic results to quantify the accuracy of the mimetic method.

2 Methods

2.1 The Grid

For this system, we want to produce velocity fields as output. Specifically, we solve the sytem for $u(x, y, t)$ and $v(x, y, t)$, which together form the complete velocity fields $\boldsymbol{w} = \begin{pmatrix} u \\ v \end{pmatrix}$. Typically, vector data would be carried on the vector field discretization of the mimetic grid (points marked v and w in Figure 1, for reference). However, because we don't have scalar quantities in our system, we carry the velocity data on the scalar grid points instead, which are marked f in Figure 1.

The problem leverages the analytic solution to supply initial and Dirichlet boundary conditions [2]:

$$u(x, y, t) = \frac{3}{4} - \frac{1}{4\left[1 + \exp\left(\frac{(-4x+4y+t)Re}{32}\right)\right]} \tag{3}$$

*Computational Science Master's Program at San Diego State University (msanders7730@sdsu.edu).
†Editor: Jose E. Castillo
‡Computational Science Research Center at San Diego State University (mdumett@sdsu.edu).

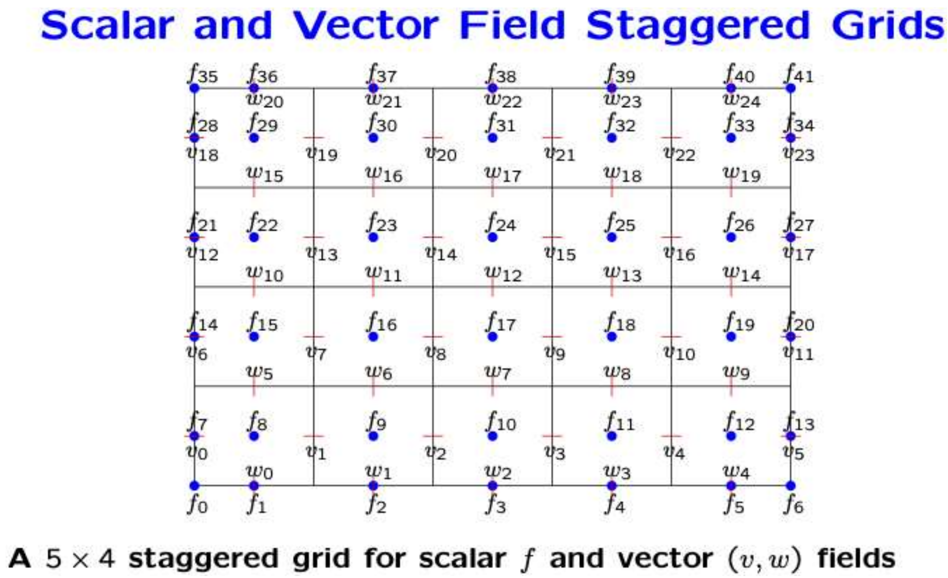


Figure 1: The mimetic staggered grid (with consent from M.A. Dumett)

$$v(x, y, t) = \frac{3}{4} + \frac{1}{4\left[1 + \exp\left(\frac{(-4x+4y+t)Re}{32}\right)\right]} \tag{4}$$

The domain is valid in 2D spatial Euclidean space. The problem domain spans $x = [-5, 5]$, $y = [-5, 5]$.

2.2 The Scheme

For the temporal discretization, a simple forward difference is used:

$$\frac{\partial u}{dt} = \frac{u^{t+1} - u^t}{\Delta t}, \qquad \frac{\partial v}{dt} = \frac{v^{t+1} - v^t}{\Delta t}$$

For the spatial discretization, we leverage the mimetic operators in the Matlab version of the MOLE library. To solve the first derivatives in our system, we use MOLE's gradient operator as generated using the `grad2D` function. The output of this function is a matrix G, which is composed of sub-operators for the partial derivatives G_x and G_y such that $G = \begin{pmatrix} G_x \\ G_y \end{pmatrix}$.

Because $\boldsymbol{G}$ calculates the gradient by differencing points on the grid, the result of G_u is stored at intermediate points between the values of u, which are labeled v_k and w_k in Figure 1. We shift our data back to our desired nodes using the interpolation operator I^G. Similar to G, I^G is constructed from sub-operators, so that $I_G = \left(I_x^G \mid I_y^G\right)$. This operator is created using `interpGMat2D()` in the MOLE library. The process of computing the gradient and interpolating back loses some boundary data (fewer points are present on the v and w grids than the f grid), so all boundary data is reapplied after these operations.

We use the 2D Laplace operator generated from MOLE using `lap2D()` to handle the RHS of the system.

When we compute u^{t+1}, the only data available is at the previous time step, along with the updated boundary conditions. When we compute v^{t+1}, though, we have newer u data available from the u^{t+1} calculation, so we use the new u^{t+1} data to find v^{t+1}.

In total, our equations in discrete form are (forward Euler in time and implicit nonlinearities)

$$\left[I + \Delta t\left(diag(u^t) I_x^G G_x + diag(v^t) I_y^G G_y\right)\right] u^{t+1} = \left[I + \nu \Delta t L\right] u^t$$
$$\Rightarrow T_1 u^{t+1} = S u^t$$

$$\left[I + \Delta t \left(diag(u^{t+1}) I_x^G G_x + diag(v^t) I_y^G G_y\right)\right] v^{t+1} = \left[I + \nu \Delta t L\right] v^t$$
$$\Rightarrow T_2 v^{t+1} = S v^t$$

We solve the system for the next time step using the MATLAB \ operator.

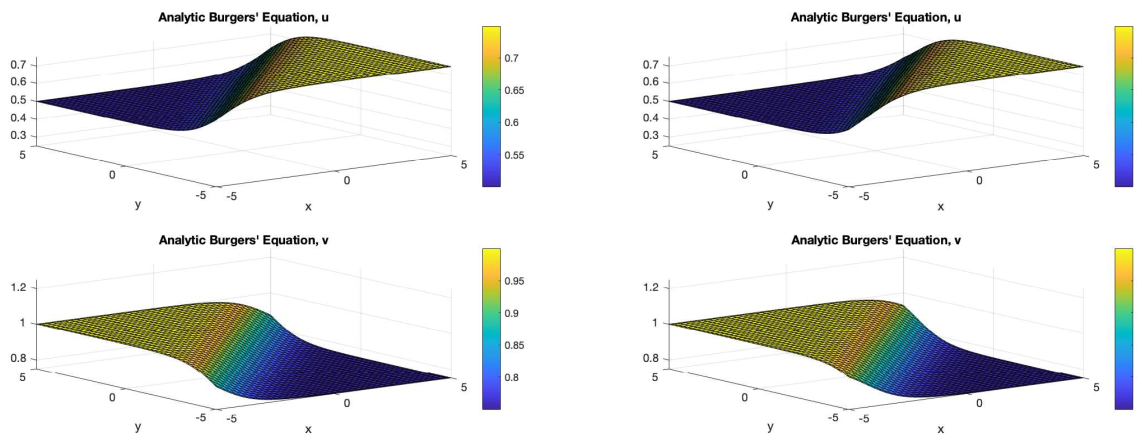
$$u^{t+1} = (T_1 \backslash S) u^t \tag{5}$$

$$v^{t+1} = (T_2 \backslash S) v^t \tag{6}$$

3 Results

All simulations used to generate figures ran for 3 simulated seconds, used $m = 40$ and $n = 40$, and applied second-order mimetic operators.

The primary test case is defined by the analytic solution to the viscous Burgers' equation presented in [2], which was previously stated in equations (3) and (4). The analytic solution is used to obtain the initial and boundary conditions for the mimetic method. The start and end states of the analytic solution, which define our target solution, are provided in Figure 2 below. The results of the mimetic method are provided in Figure 3.



(a) $t = 0$ (b) $t = 3$

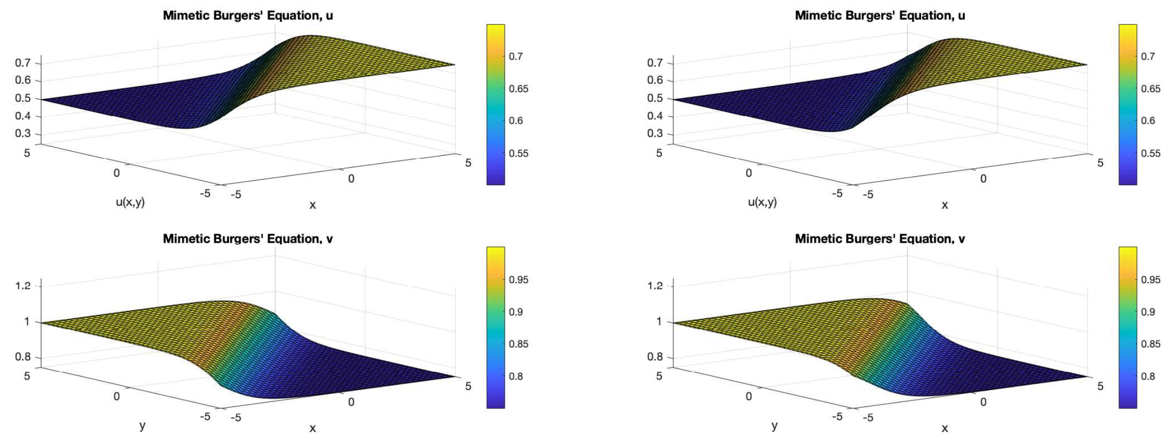
Figure 2: Evolution of the exact solution

Because the differences between the mimetic and analytic results are subtle, an error plot of the difference between the analytic and mimetic end states is provided in Figure 4.

A resolution study was conducted to determine how the mimetic solution converged to the exact solution as the grid was refined. Error data is collected in Table 1. A log plot of the L^2 data along with a line of best fit can be seen in Figure 5.

Δx	L_2 Error	Max Error	Wall Time (s)
2	7.251e-3	2.195e-2	3
1	2.575e-3	1.334e-2	5
0.5	6.361e-4	4.305e-3	11
0.25	1.796e-4	1.298e-3	67

Table 1: Simulation data for several grid refinements



(a) $t = 0$ (b) $t = 3$

Figure 3: Evolution of the mimetic solution

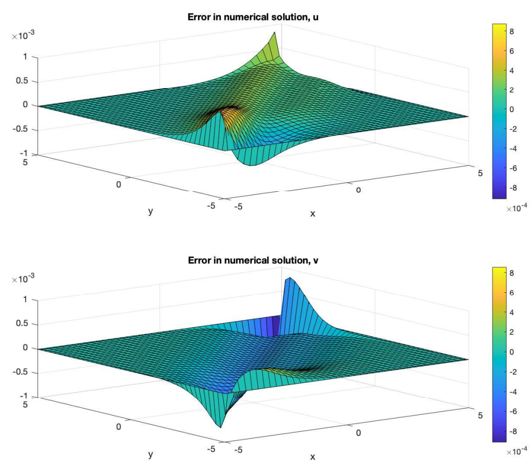


Figure 4: Error in the mimetic solution at $t = 3$

An extra case was also run using the Witch of Agnesi curve. A simple Witch curve in two dimensions is defined by

$$z = \frac{1}{x^2 + y^2 + 1}$$

For this simulation, Dirichlet boundaries were enforced using the operator returned by `robinBC2D()`. The boundary values of the initial distribution were used throughout the solution. The initial and end states are shown in Figure 6.

4 Discussion

From the convergence history, we see that we are getting nearly second-order accuracy from our method, which is what we expect based on the order of our mimetic operators. Additionally, from Figure 4 we see that the majority of our error is accrued near the boundary. The order of accuracy of the mimetic solution could be improved by using higher-order operators, of course. More accurate time-stepping schemes, such as Runge-Kutta, could also be used in place of the simple forward Euler used in this case to improve the solution accuracy.

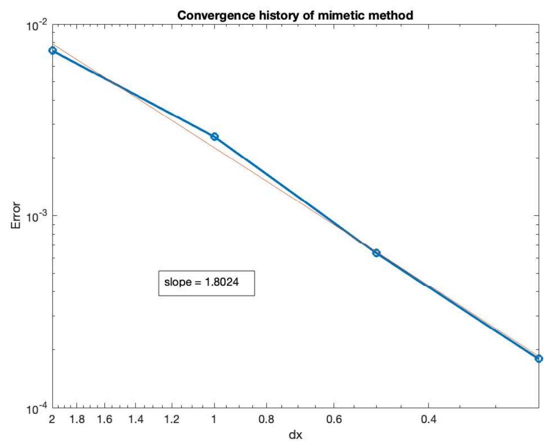
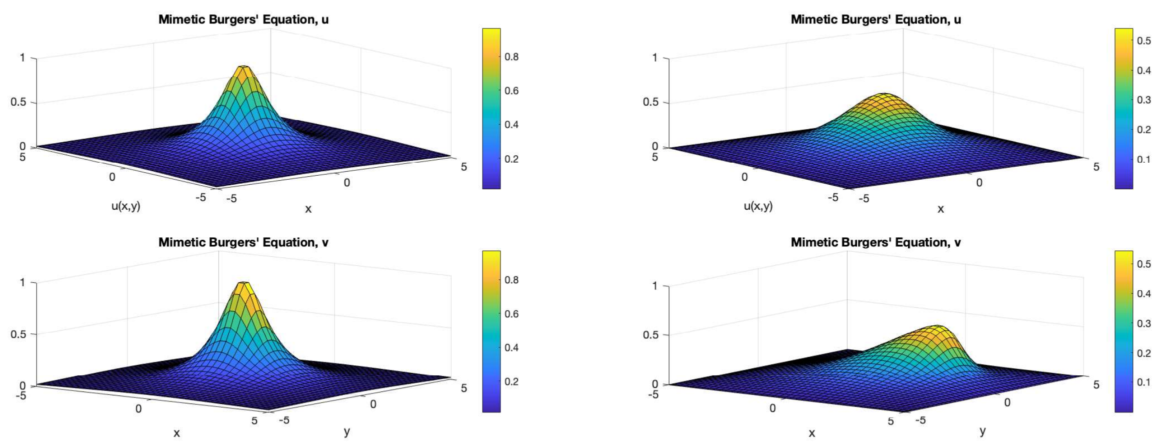


Figure 5: Log plot of the $L2$ error as the grid is refined



(a) $t = 0$

(b) $t = 3$

Figure 6: Witch of Agnesi simulation results

The Witch of Agnesi solution is provided to demonstrate a classic viscous Burgers' behavior, the "nonlinear diffusive hump". As the simulation progresses, the initial condition both spreads out and "bunches up" as the faster advection catches up to the slower flow upstream. Even though there is not an analytic solution with which to compare the result, the presence of well-documented behavior is additional evidence that the method is working.

5 Conclusion

Overall, the mimetic operators did a good job of approximating the solution to the viscous Burgers' equation. The maximum error after 3 simulated seconds was 1.298×10^{-3} for the finest grid refinement, and the L^2-norm of the combined u and v errors was 1.796×10^{-4}. The simulation took approximately 67 seconds of wall time to complete. Second-order convergence was expected based on the construction of the matrices, and the solution exhibited 1.8-order convergence as the grid was refined.

Ultimately, the mimetic operators proved to be a useful and accurate tool for the approximation of the viscous Burgers' equation.

References

[1] Corbino, J., and Castillo, J. E. High-order mimetic finite-difference operators satisfying the extended gauss divergence theorem. *Journal of Computational and Applied Mathematics 364* (2020), 112326.

[2] Shukla, H., Tamsir, M., Srivastava, V., and Kumar, J. Numerical solution of two dimensional coupled viscous burgers equation using the modified cubic b spline differential quadrature method. *AIP Advances 4* (09 2014), 117134–10.