Solution to Burger's Equation (Viscous)

Ian William Hoppock

May 30, 2017

1 General Remarks and Documentation

We solve the viscous Burger's equation

$$u_t + uu_x = \alpha u_{xx}$$

such that $x \in [-1, 1]$, and the initial condition is $u(x, 0) = -\sin \pi x$. First, we treat the system in time,

$$\int_{t^n}^{t^{n+1}} u_t dx + \int_{t^n}^{t^{n+1}} uu_x dx = \int_{t^n}^{t^{n+1}} \alpha u_{xx} dx,$$

using a one-step Adams-Moulton for the linear term (u_{xx}) and a two-step Adams-Bashforth for the non-linear term (uu_x):

$$u^{n+1} - u^n + \frac{dt}{2}\left(3(uu_x)^n - (uu_x)^{n-1}\right) = \alpha \frac{dt}{2}\left(u_{xx}^{n+1} + u_{xx}^n\right).$$

We treat the spatial integration with two specific spectral methods:

1. Fourier-Galerkin with periodic boundary conditions;
2. Collocation on the Tchebyshev-Gauß-Lobatto points based on the n^{th} Tchebyshev function with Dirichlet boundary conditions $u(-1, t) = u(1, t) = 0$.

The two programmes are written in Matlab, not C, as I unfortunately (I do say that sincerely) could not get the `fftw3.h` working in the timeframe that I had. For the Fourier-Galerkin, simply run the programme `Viscous_Burgers_FG.m` in Matlab in the same directory as the Matlab function `burger.m`, which is called repeatedly. For the collocation, run `Viscous_Burgers_TGL.m` in Matlab.

2 Fourier-Galerkin

We see results as expected. The larger the diffusion term α, the less steepening there is, with substantially more diffusion. We plot the graph at various values of α to establish that the code is working.

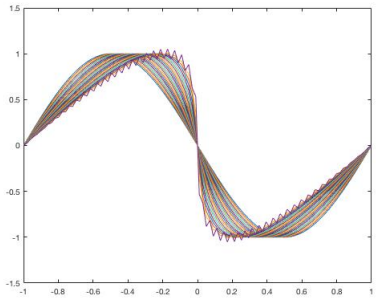

Figure 1: $\alpha = 0$. We expect no diffusion, i.e., we simply have the solution of the inviscid Burger's equation. Indeed, we have the exactly.

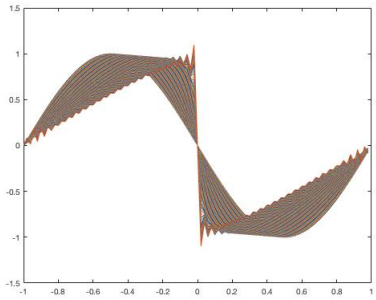

Figure 2: $\alpha = 0.00001$. With such a small diffusion coefficient, we expect very gradual diffusion with, with the bulk of movement to be in steepening. Note, in this picture, as in all relevant pictures, we see the Gibbs phenomenon acting precisely as it ought to.

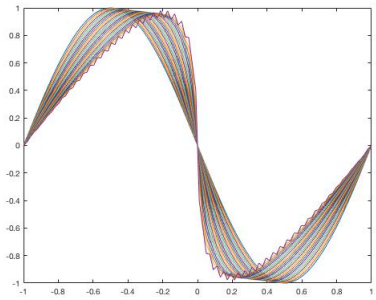

Figure 3: $\alpha = 0.00001$. We have the same results as the above figure; however, we have greater spatial resolution. Above we have 100 spatial points, here it is 500.

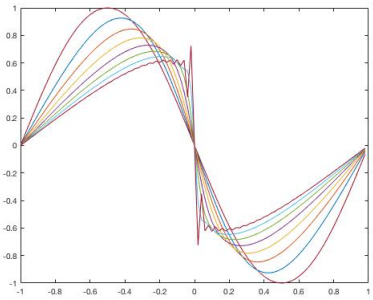

Figure 4: $\alpha = 0.001$. We see a clearer Gibbs phenomenon in the very last line plotted. This is not shown in the figure below, because the greater spatial resolution puts off this for a few more time steps.

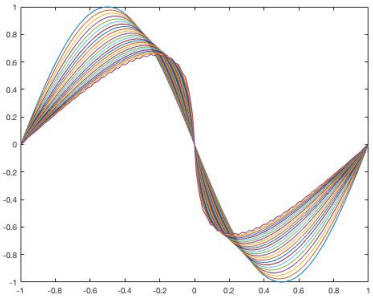

Figure 5: $\alpha = 0.001$. We have the same results as the figure above; however, we have a greater spatial resolution. Again, above we have 100 spatial points, here it is 500. Indeed, here we do not see an unresolvable situation, as it was stopped just short

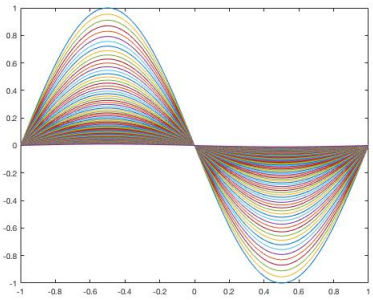

Figure 6: $\alpha = 1$. Having such a high diffusion coefficient, the non-linear term does not have a chance to steepen the wave, and we see pure diffusion.

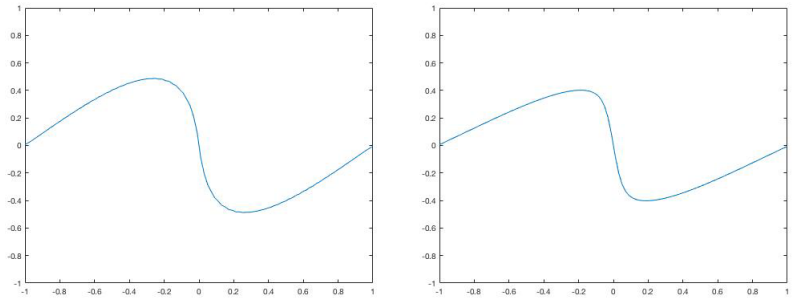

Figure 7: Left: $\alpha = 0.01$. Right: $\alpha = 0.1$. The final wave position after the simulation with two different diffusion coefficient values. The left side is steeper than the right, as expected, since it is diffusing less in time.

3 Collocation on Tchebyshev-Gauß-Lobatto

In lieu of using the standard Tchebyshev-Gauß, we use the Tchebyshev-Gauß-Lobatto points. The former are the zeros of an oscillatory function, which goes on to result in the exclusion of the boundary points. This is truly problematic for orthogonality, thus the latter consist of the extrema and the end points of the same oscillatory function, as seen in Figure 8. Thus, we have

$$u = \sum_{k=0}^{N} u_k T_k = \sum_{k=0}^{N} u_k \Psi_k,$$

where

$$\Psi_k = (-1)^{k+1} \frac{(1-x^2)T_N'}{c_k N^2 (x - x_k)}$$

and

$$c_k = \begin{cases} 1: & 0 < k < N \\ 2: & k = 0,\ N \end{cases}.$$

Thus, our system is now closed and then functions are cardinal such that

$$u(x_j) = u_k$$

and

$$x_j = \cos \frac{j\pi}{N}.$$

Because cardinal functions are either zero or one on grid points, the coefficients are also grid point values of the function.

Thusly, we have a system that uses cardinal functions on the Tchebyshev-Gauß-Lobatto points. Next, the derivatives follow readily

$$\frac{\mathrm{d}u}{\mathrm{d}x} = \frac{\mathrm{d}}{\mathrm{d}x} \sum_{k=0}^{N} u_k T_k = \sum_{k=0}^{N} u_k \frac{\mathrm{d}}{\mathrm{d}x} \Psi_k.$$

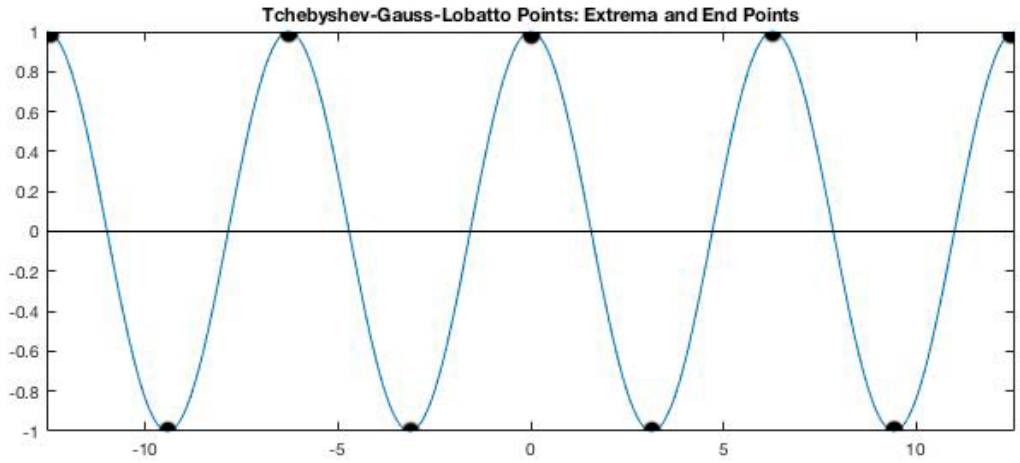

Figure 8: The Tchebyshev-Gauß-Lobatto points are located at the extrema and the end points. Above there are nine shown. This is different from the Tchebyshev-Gauß points, which are the zeros of the curve. Were we to use those points, there would be eight in the above.

Thence, from our equation of Ψ and from the Tchebyshev derivative recurrence relation, we have

$$\frac{\mathrm{d}\Psi_k}{\mathrm{d}x} = \frac{(-1)^{k+1}}{c_j N^2}\left(\frac{x(x-x_k)+(1-x^2)}{(x-x_k)^2}T_N' + \frac{N^2}{x-x_k}T_N\right).$$

Evaluating the above at each grid point x_j forms the following matrix

$$d_1 = \begin{cases} \frac{1+2N^2}{6} & j=k=0 \\ -\frac{1+2N^2}{6} & j=k=N \\ -\frac{x_k}{2(1-x_k^2)} & j=k, 0<k<N \\ \frac{(-1)^{j+k}c_j}{c_k(x_j-x_k)} & j\neq k \end{cases}.$$

At last, we have the following results, again by running `Viscous_Burgers_TGL.m` in Matlab. Please note the CFL conditions when working with the code. In all of the following figures, with one exception at the end, we present a triptych of the situation. The first panel shows the standard progression of the wave, superimposed, with its appropriate diffusion and steepening. The second panel shows the entire progression, superimposed, until it blows up. This is is visible at the end points. It likely shows a few time steps after the initial blow up to make it noticeable on the busy image. The last panel shows just the blown up wave.

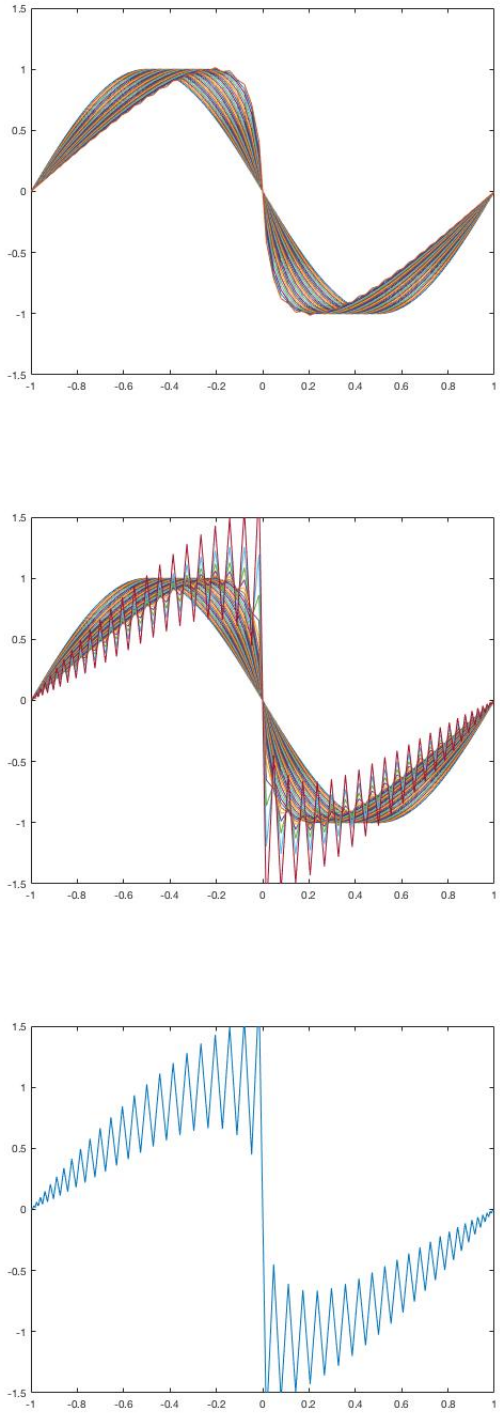

Figure 9: $\alpha = 0$. The standard inviscid Burger's equation. This acts exactly as we expect with a rather standard blowing up as a result its non-diffusive nature.

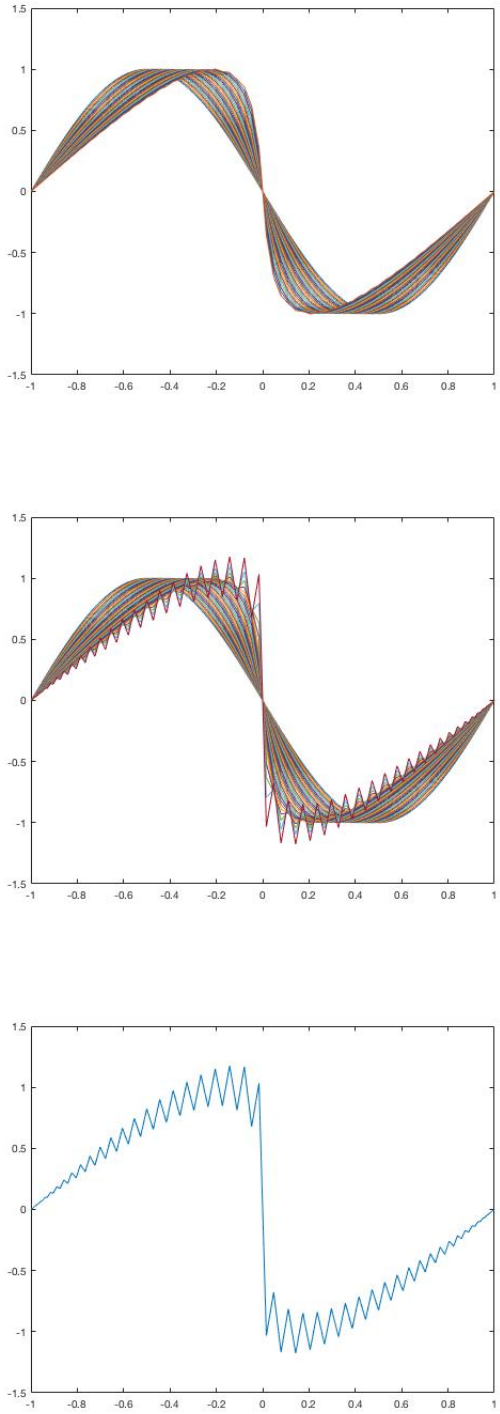

Figure 10: $\alpha = 0.001$. Indeed, this is such a small diffusion coefficient that we barely can tell that it has diffused. Use a straight edge to convince oneself.

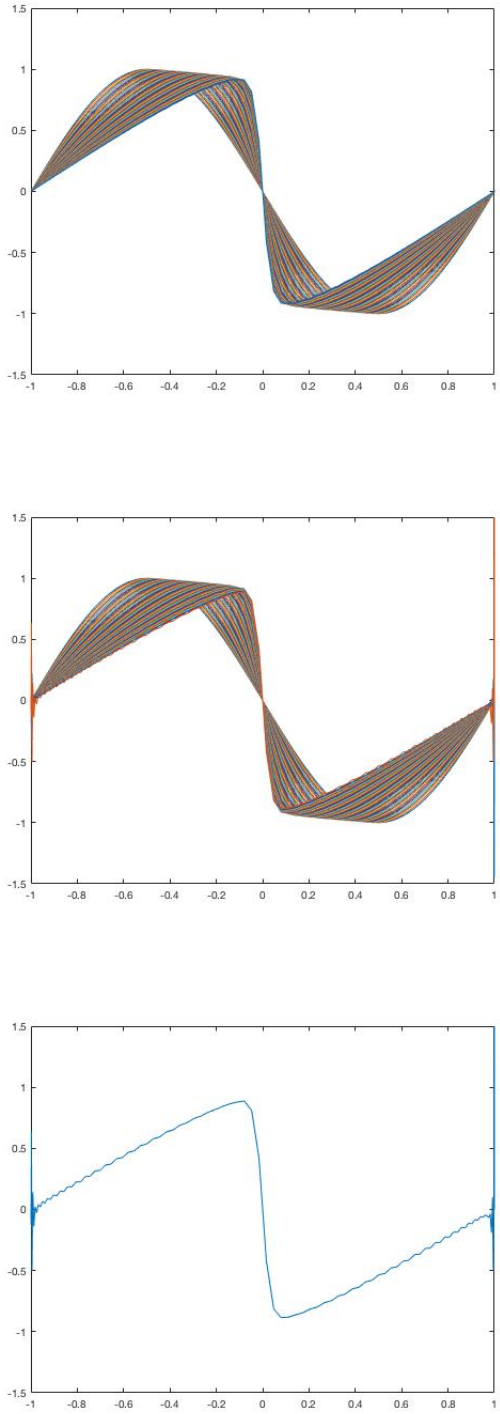

Figure 11: $\alpha = 0.01$. We see more acute steepening of the wave coupled with diffusion. As a result of this, the Tchebyshev-Gauß-Lobatto method is blowing up at the endpoints, as we expect.

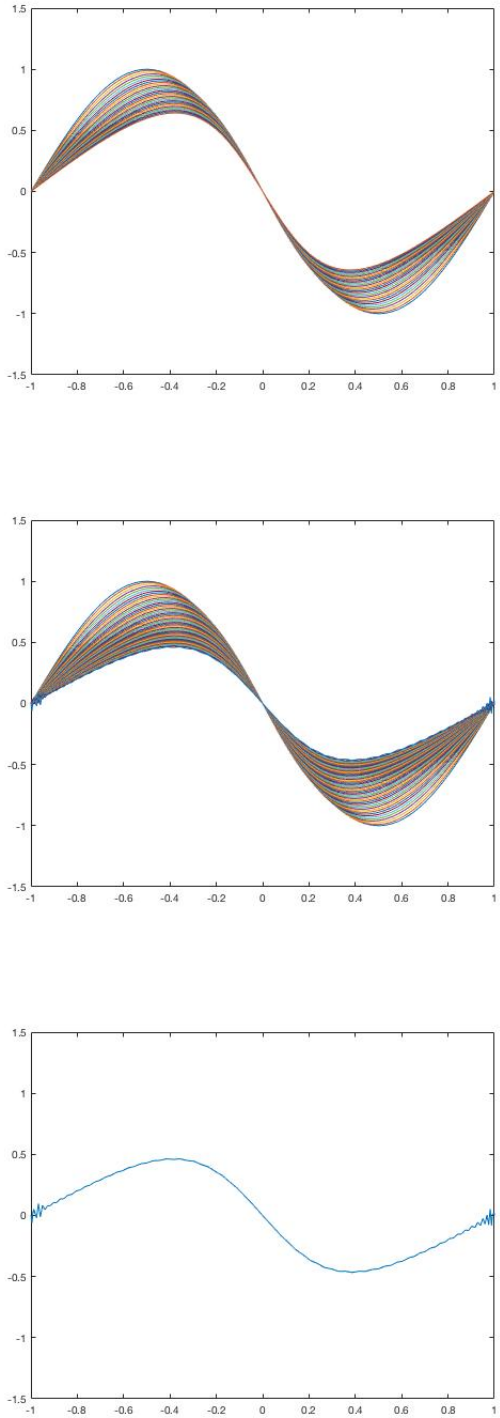

Figure 12: $\alpha = 0.1$. Indeed, we see a similar result as the figure prior, with more exaggerated diffusion.

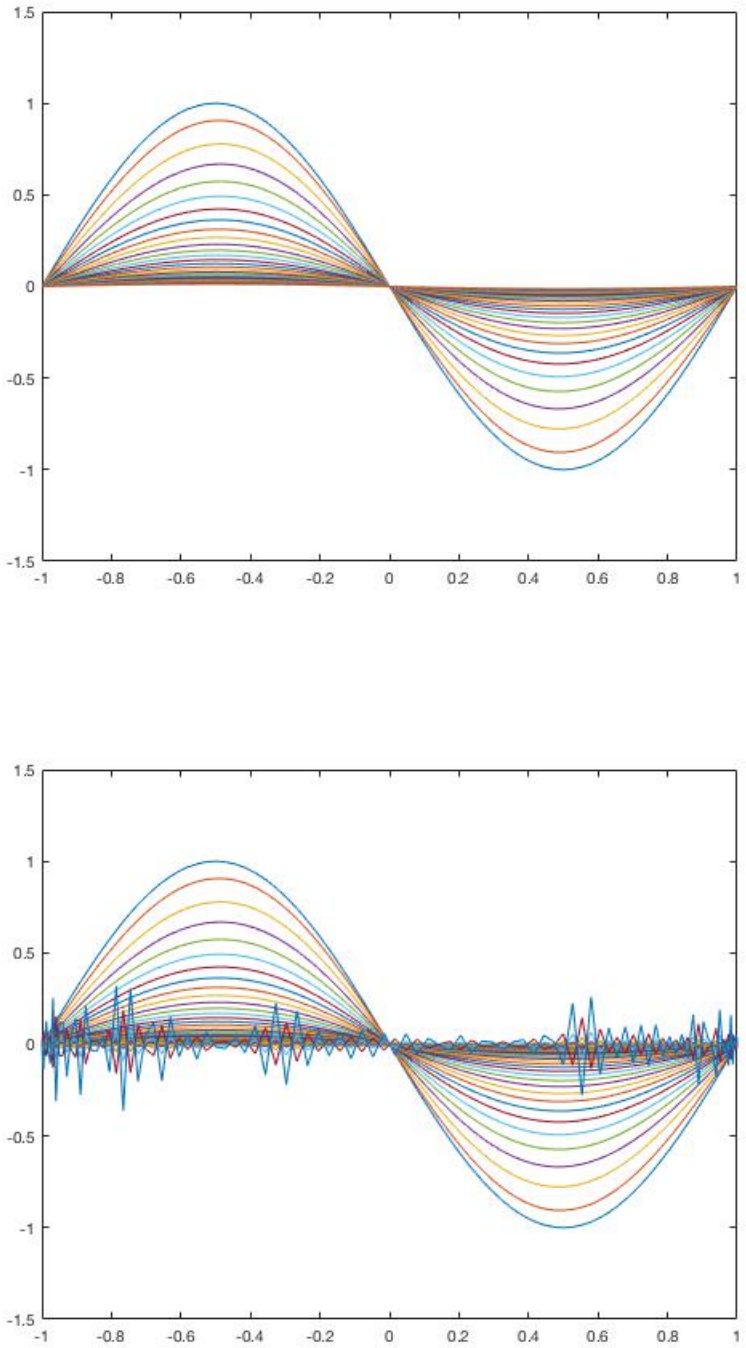

Figure 13: $\alpha = 1$. Lastly, we see such strong diffusion that there is virtually no steepening. The third panel is omitted, as there is little to infer from it. However, it is clear that the Tchebyshev-Gauß-Lobatto grid indeed blows up on a flat line.

4 Comparison

These two methods are both quite difficult to work with. Whilst there is the spectral accuracy that comes with these methods, it is this author's opinion that finite difference methods are more tuned to being user friendly and more readily programmed in lower level languages, i.e., C. I would not mind, if I had the time, programming these in Julia to see if I found them easier with which to work.

Now, for the specific methods used, Galerkin type methods are substantially easier to code. This is because Galerkin and collocation methods are morally different. In the former we seek to represent the function in a discretised space using a set of basis functions and periodic boundary conditions. In the latter we choose a polynomial of a certain degree N and sample a number of points in the domain. Then we find a solution which satisfies the equation at the collocation points.

In comparing the two methods against each other, for a one-dimensional PDE, it is reasonable to suggest that the Fourier-Galerkin is better than the collocation on the TGL points.