COMPUTATIONAL ASTROPHYSICS REPORT

UNIVERSITY OF COPENHAGEN

DEPARTMENT OF PHYSICS

Final report

Code

```python
def HLLC00_2D(ql,qr): #Variant 0 of HLLC Riemann Solver
    n=len(ql.U)
    c_left  = (ql.gamma*ql.P / ql.D)**0.5
    c_right = (qr.gamma*qr.P / qr.D)**0.5
    c_max = np.maximum(c_left,c_right)
    SL = np.minimum(np.minimum(ql.U,qr.U)-c_max,0) # <= 0.
    SR = np.maximum(np.maximum(ql.U,qr.U)+c_max,0) # >= 0.
    nominator=qr.P-ql.P+ql.D*ql.U*(SL-ql.U)-qr.D*qr.U*(SR-qr.U)
    denominator=ql.D*(SL-ql.U)-qr.D*(SR-qr.U)
    S_star=nominator/denominator     #calculation of the S*
    Ul = primitive_to_conservative(ql)
    Ur = primitive_to_conservative(qr)
    Fl = Hydro_Flux(ql,Ul)
    Fr = Hydro_Flux(qr,Ur)
    F_star=np.zeros((4,n,n)) # We define 4 arrays which are correlated to the fluxes of ρ (0), u (1), v (2) and E(3), in 1D omit v
    mask1 = (SL > 0)
    F_star[0][mask1] += Fl.D[mask1]
    F_star[1][mask1] += Fl.mU[mask1]
    F_star[2][mask1] += Fl.mV[mask1]  # In 1D omit the V coordinate of velocity
    F_star[3][mask1] += Fl.Etot[mask1]
    mask2 = (S_star >= 0) & (SL <= 0)
    a=np.zeros((n,n))
    b=np.zeros((n,n))
    Energy_left=np.zeros((n,n))
    a[mask2]+=ql.D[mask2]*((SL[mask2]-ql.U[mask2])/(SL[mask2]-S_star[mask2]))
    F_star[0][mask2]+=Fl.D[mask2]+SL[mask2]*(a[mask2]*1-Ul.D[mask2])
    F_star[1][mask2]+=Fl.mU[mask2]+SL[mask2]*(a[mask2]*S_star[mask2]-Ul.mU[mask2])
    F_star[2][mask2]+=Fl.mV[mask2]+SL[mask2]*(a[mask2]*ql.V[mask2]-Ul.mV[mask2]) #In 1D  omit the V coordinate of velocity
    Energy_left[mask2]+=(ql.P[mask2]/(ql.gamma-1))+0.5*(ql.D[mask2]*(ql.U[mask2]**2+ql.V[mask2]**2))  # In 1D omit the V coordinate of velocity in the energy
    b[mask2]+=(Energy_left[mask2]/ql.D[mask2])+(S_star[mask2]-ql.U[mask2])*(S_star[mask2]+ql.P[mask2]/(ql.D[mask2]*(SL[mask2]-ql.U[mask2])))
    F_star[3][mask2]+=Fl.Etot[mask2]+SL[mask2]*(a[mask2]*b[mask2]-Ul.Etot[mask2])
    mask3 = (S_star < 0) & (SR > 0)
    a=np.zeros((n,n))
    b=np.zeros((n,n))
    Energy_right=np.zeros((n,n))
    a[mask3]+=qr.D[mask3]*((SR[mask3]-qr.U[mask3])/(SR[mask3]-S_star[mask3]))
    F_star[0][mask3]+=Fr.D[mask3]+SR[mask3]*(a[mask3]*1-Ur.D[mask3])
    F_star[1][mask3]+=Fr.mU[mask3]+SR[mask3]*(a[mask3]*S_star[mask3]-Ur.mU[mask3])
    F_star[2][mask3]+=Fr.mV[mask3]+SR[mask3]*(a[mask3]*qr.V[mask3]-Ur.mV[mask3]) # In 1D we omit the V coordinate of velocity
    Energy_right[mask3]+=(qr.P[mask3]/(ql.gamma-1))+0.5*(qr.D[mask3]*(qr.U[mask3]**2+qr.V[mask3]**2))
    b[mask3]+=(Energy_right[mask3]/qr.D[mask3])+(S_star[mask3]-qr.U[mask3])*(S_star[mask3]+qr.P[mask3]/(qr.D[mask3]*(SR[mask3]-qr.U[mask3])))
    F_star[3][mask3]+=Fr.Etot[mask3]+SR[mask3]*(a[mask3]*b[mask3]-Ur.Etot[mask3])
    mask4 = (SR <= 0)
    F_star[0][mask4] += Fr.D[mask4]
    F_star[1][mask4] += Fr.mU[mask4]
    F_star[2][mask4] += Fr.mV[mask4]  # In 1D omit the V coordinate of velocity
    F_star[3][mask4] += Fr.Etot[mask4]
    Flux=void()
    Flux.D=np.copy(F_star[0,:])
    Flux.mU=np.copy(F_star[1,:])
    Flux.mV=np.copy(F_star[2,:])  # In 1D omit the V coordinate of velocity
    Flux.Etot=np.copy(F_star[3,:])
    return Flux
```

Figure 1: The implementation of the code in 2D. There are comments for 1D

Finite Volume hydrodynamics and Riemann Solvers

The finite volume hydrodynamics is a method for solving the equations of fluid dynamics by dividing the fluid domain into a large number of small, discrete control volumes. This method can be used on all differential equations, which can be written in the divergence form. The basic principle is to integrate the equations of fluid dynamics over each control volume, which allows for the conservation of mass, momentum, and energy. The control volumes can be of any shape, but they are usually chosen to be small enough so that the flow variables within each control volume would be approximately constant. The equations of fluid dynamics are then solved at the interfaces between the control volumes, which allows for the determination of the flow variables within each control volume.

The accuracy of the solution can be improved by using high-order methods for the spatial and temporal discretization. For example, higher order finite volume methods such as WENO or ENO schemes can be used for spatial discretization, and high-order time-stepping methods such as Runge-Kutta can be used for temporal discretization. Additionally, to capture shocks, contact surfaces and other discontinuities in the flow, approximate Riemann solvers are used to calculate the flow variables at the interfaces between control volumes. These solvers rely on different mathematical approaches, such as linearization, upwind schemes, and slope limiting, to accurately calculate the flow variables at the interfaces.

An approximate Riemann solver is a method for solving the Riemann problem, which is a simplified representation of a fluid flow problem involving a discontinuity in the flow variables. The Riemann problem is used to calculate the flow variables on either side of the discontinuity, which are then used to update the flow variables in the control volumes. Thus, the Riemann solver can be thought of as a method for determining the exact solution of the equations of fluid dynamics at the interfaces of the control volumes, which can be used to calculate the exact flow variables within each control volume. Approximate Riemann solvers are used in finite volume hydrodynamics because they are computationally efficient and can be used to accurately simulate complex fluid flows. Many Riemann solvers exist, such as Lax-Friedrichs, Roe, and more recently, the more accurate ones like HLLC (Harten-Lax-van Leer-Contact).

Now, at this report we are going to concentrate on the compressible Euler equations. These equations are correlated to the conservation concepts of mass, momentum and energy. One of the difficulties of such equations is spotted on the non-linear term of the velocity, which is related to the appearance of shocks and turbulence on the flow. Moreover, a hydrodynamic problem facilitates the wave propagation and thus we expect a change on the density, velocity as well as the pressure of the system. Our approach to such a demanding problem would be the finite volume method. As already being mentioned, we will consider that the flux goes through the surface of each cell of the grid resulting to a vital equation:

$$u_i^{n+1} - u_i^n = -\frac{\Delta t}{\Delta x} \cdot (F_{i+1/2}^{n+1/2} - F_{i-1/2}^{n+1/2}),\ where\ F_{i+1/2}^{n+1/2} = \frac{1}{\Delta t} \int_t^{t+\Delta t} dt F_{i+1/2}(q(x_{i+1/2})) \tag{1}$$

Consequently, we are dealing with a Riemann problem a typical form of which is $\boldsymbol{U}_t + \boldsymbol{F}(\boldsymbol{U})_x = 0$.

In general, the key concept of Riemann solvers it the evaluation of the flux through the identification of the appropriate wave pattern in the Riemann problem solution. The main aim of this report would be the HLLC (Harten-Lax-van Leer-Contact) approximate Riemann solver, which is an extension of the simpler Riemann solver, named HLL (Harten-Lax-van Leer). In order to explain the idea behind HLLC, we are going to address the HLL at first. More specifically, a wave configuration with two waves separating three constant states is considered (Figure 2). Once the wave speeds are known, the HLL solver uses them to compute the fluxes across the discontinuity. However, this is true only for two-equation hyperbolic systems, such as the shallow water equations in one dimension. The two-wave assumption is false for more complex systems, such as the split two-dimensional shallow water equations or the Euler equations. As a result, physical features including contact surfaces, shear waves, and material interfaces may not be accurately resolved. The ensuing numerical smearing for the limiting scenario in which these features are stationary in relation to the mesh is unsatisfactory. In order to ease the problem of the intermediate wave, a new approaches of the HLL was created. One of those is the HLLC.

The basic difference between these two solvers is that the latter the wave configuration of the HLLC solver is based on the idea that there are three types of waves that can propagate through a fluid: fast waves, intermediate waves, and slow waves. The HLLC solver uses this wave configuration to determine the behavior of the fluid in different regions of the computational domain. All these quantities as well as the specific procedure that we are going to follow, are thoroughly described to the book ”Riemann Solvers and Numerical Methods for Fluid Dynamics”[2], one could be able to calculate all the terms that are needed so as to implement the HLLC solver and define the flux for each conserved quantity. (maybe add something more)

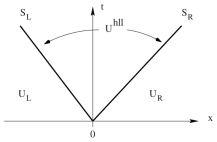

Figure 2: The scheme of HLL solver. The value U is correlated to the conserved quantities, e.g density. The S_L and S_R are the signal speeds.

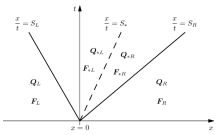

Figure 3: The scheme of HLLC solver. The value U is correlated to the conserved quantities, e.g density. The S_L and S_R are the signal speeds. The star quantities are calculated inside the intermediate state.

Implementation of HLLC in various tests

In a first step, we used the HLLC solver in order to simulate the procedure of a blast wave for an exponential density profile. Simultaneously, we solved the same system with the HLL solver and tried to compare the results. Our results are displayed on the figures 4, 5. The solvers composed almost exactly the same solution. Though with a closer look at the figure 5 it is apparent that there are 4 points, two in each axis that shows a small instability (axis-coordinated). A reason for that might be the local low Mach number that is created. Particularly, when the shock wave moves in x-direction the velocity components of the local transverse direction , has a vanishing magnitude. Consequently, the local directional Mach number will vanish as well during the computation of the fluxes in transverse directions of the shock wave propagation. All in all, the low local Mach number could be the driving mechanism of the numerical grid-aligned instability and it is worth mentioning that there has been extend research on the Riemann solvers at low Mach number regimes[1]. Thus, in order to cope with such phenomena, new versions of the HLLC Riemann solver have proposed, e.g HLLM-LM [1].

In summary, through the comparison of the results we got from the two solvers we implemented, the HLLC resolved efficiently the 2D problem of the blast wave on both cases of the different density profile. In this way we could verify the validity of this mechanism.

Secondly, even though the validity of the aforementioned solver was confirmed via the 2D problem test, we investigated its behaviour on one dimensional problems as an extra. Concretely, for that reason we chose 2 shock tube problems[2].

Problems	ρ_L	u_L	P_L	ρ_R	u_R	P_R
Test problem 1	1	0.75	1	0.125	0	0.1
Test problem 3	1	0	1000	1	0	0.01

The test problem 1 is a modification of the Sod's problem and it is characterized by shock wave that has a right direction as well as a rarefaction wave. The test problem 3 is focusing on the accuracy and the consistency of the solvers as it includes severe discontinuities and strong shocks. The main goal of this test bench is to show the validity of the HLLC in terms of producing somehow the same results as the one being display on Chapter 10 of the Riemann book[2] for these problems.

The figures 6, 7 and 8 show the result that we got by implementing the HLLC solver to the shock tube problem of Sod. At first, both HLL and HLLC methods produce almost identical results without any difference. If we compare the results we got with the original that are shown on the Figure 9, we could mention that the HLLC we used approaches efficiently enough the solution that is depicted in terms of the form.

Now, the figures 10, 11 and 12 manifest the outcome of the HLLC solver when the initial conditions of the test problem 3 were implemented. Our solver managed to simulate and solve this problem quite well. Our results are similar with one being exhibited to Figure 13. However, in the case of velocity we see that there are signs of diffusion as a kind of peak is present close to x=0.5. A possible reason for that could be that the flux is not efficiently slope limited in compare to the HLLC solver that has been used for the results on figure 13. Moreover, considering figure 14 and the density image on 13 , it is obvious that the HLLC solver yields a sharper representation of the contact discontinuity, whereas the HLL solver retains a slightly more diffusive character. Overall, this test could verify validity of the HLLC solver.

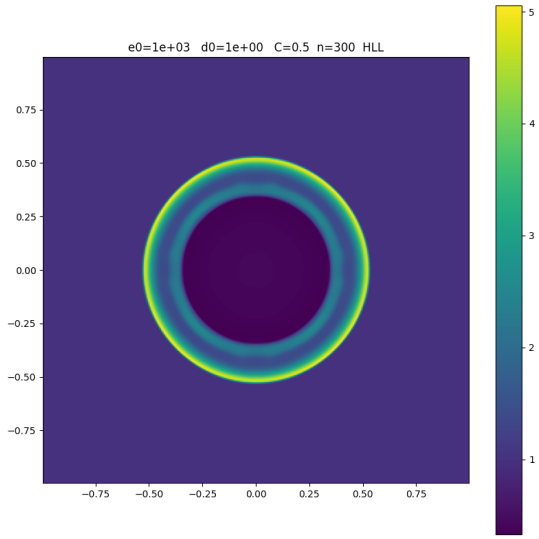

Figure 4: The result of the HLL solver for the 2-dimensional problem of the blast wave. The profile of the density is exponential.The number of points are 300 and the Courant time equals 0.5

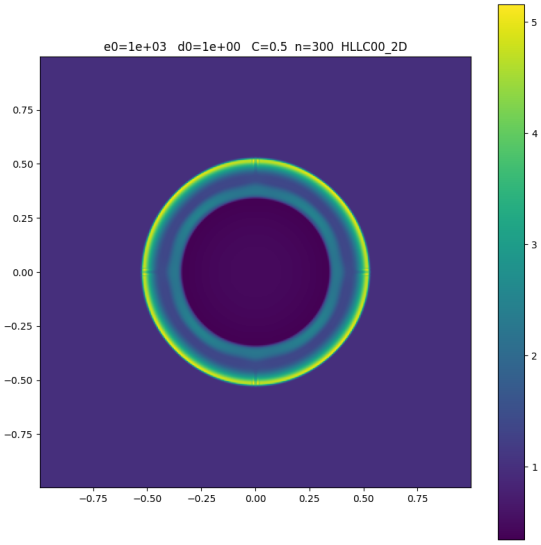

Figure 5: The result of the HLLC solver for the 2-dimensional problem of the blast wave. The profile of the density is exponential.The number of points are 300 and the Courant time equals 0.5.

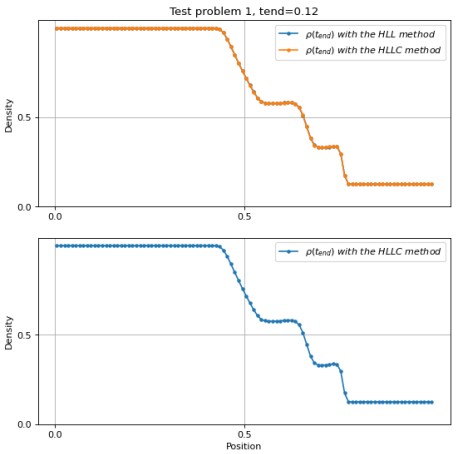

Figure 6: On the top image there is the approach of the test problem 1 concerning the density profile by the HLL and HLLC Riemann solver for t=0.12 units. The bottom image depicts the HLLC approach solely.

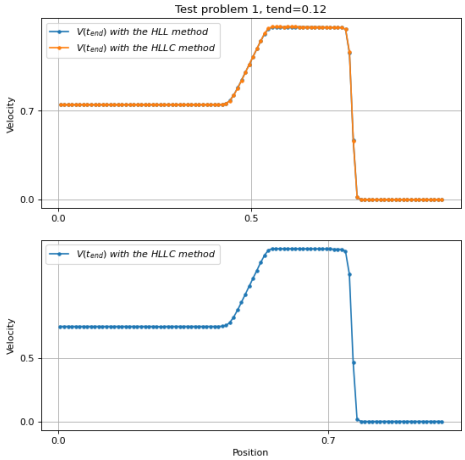

Figure 7: On the top image there is the approach of the test problem 1 concerning the velocity by the HLL and HLLC Riemann solver for t=0.12 units. The bottom image depicts the HLLC approach solely.

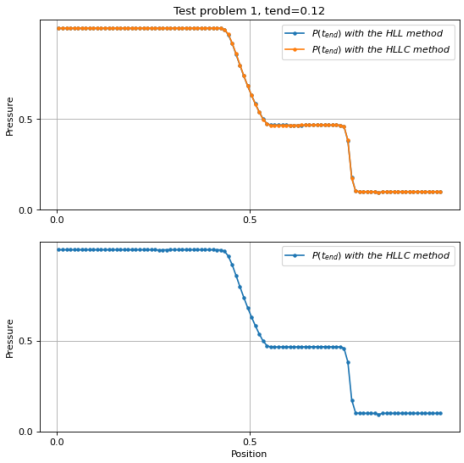

Figure 8: On the top image is the approach of the test problem 1 concerning the density profile, by the HLL and HLLC Riemann solver for t=0.12 units. The bottom image depicts the HLLC approach solely.

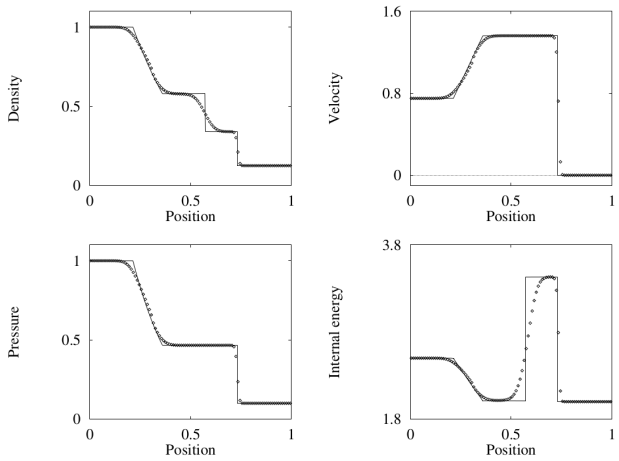

Figure 9: This set of images are the results are depicted the comparison of the HLLC solver with the exact solution of the test problem 1.

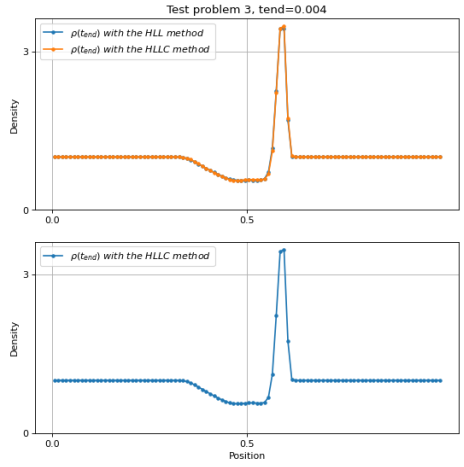

Figure 10: On the top image there is the approach of the test problem 3 concerning the density profile by the HLL and HLLC Riemann solver for t=0.004 units. The bottom image depicts the HLLC approach solely.

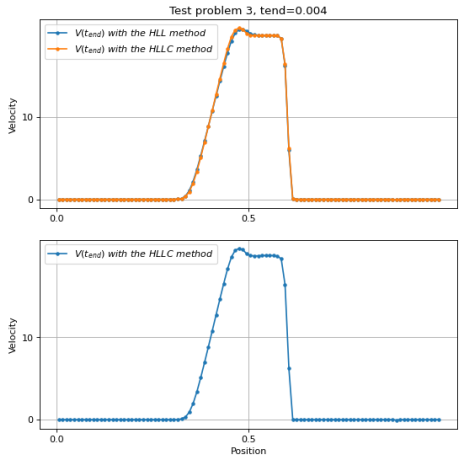

Figure 11: On the top image there is the approach of the test problem 3 concerning the velocity by the HLL and HLLC Riemann solver for t=0.004 units. The bottom image depicts the HLLC approach solely.

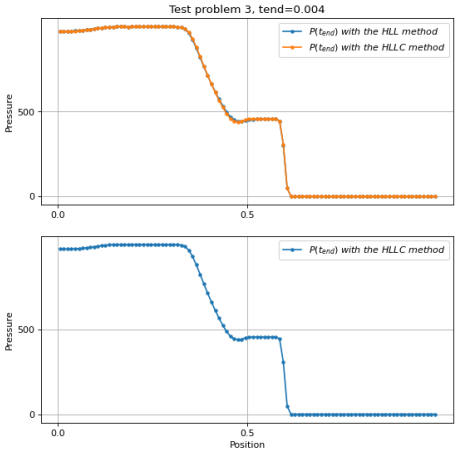

Figure 12: On the top image is the approach of the test problem 3 concerning the density profile, by the HLL and HLLC Riemann solver for t=0.004 units. The bottom image depicts the HLLC approach solely.

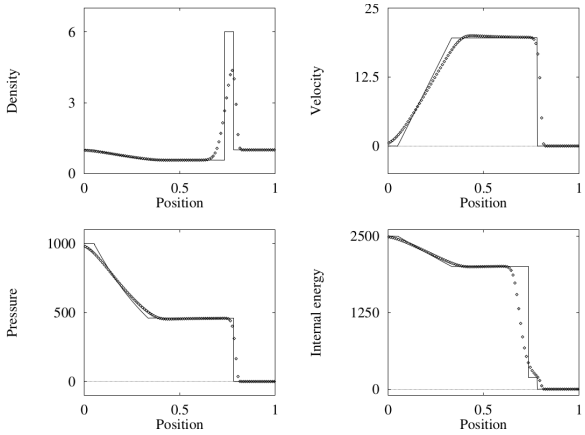

Figure 13: This set of images are the results are depicted the comparison of the HLLC solver with the exact solution of the test problem 3.

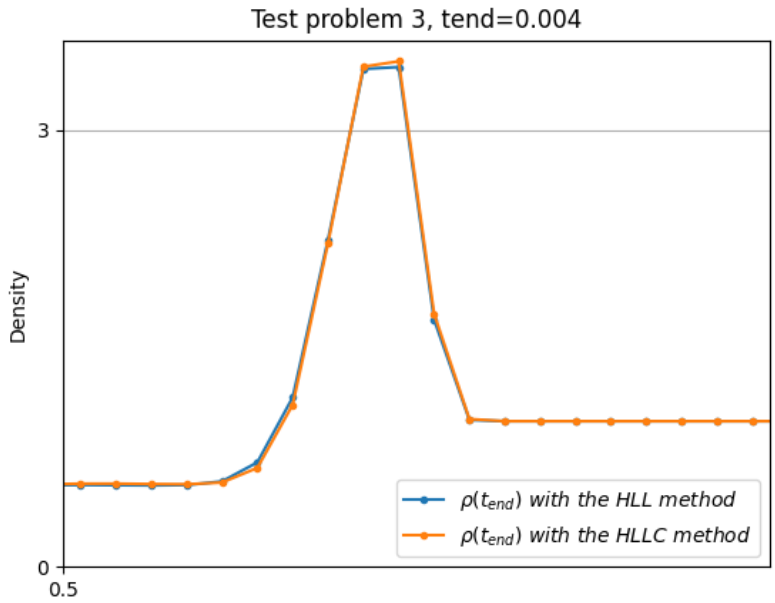

Figure 14: This figure is the zoomed version of figure 10.

References

[1] Nico Fleischmann, Stefan Adami, Nikolaus A. Adams, ”A shock-stable modification of the HLLC Riemann solver with reduced numerical dissipation”, Journal of Computational Physics 423 109762, 2020,21 pages
[2] Professor Eleuterio F. Toro, ”Riemann Solvers and Numerical Methods for Fluid Dynamics”, Springer, 2009, 724