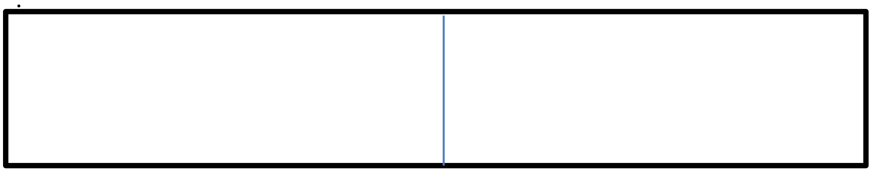
van Leer Flux Vector Splitting Scheme for Sod Shock Tube Problem

1. Problem



Initial condition:

Density_left = 1.0	Density_right = 0.125
Pressure_left = 1.0	Pressure_right = 0.1
Velocity_left = 0.0	Velocity_right = 0.0

2. Numerical scheme

1D Euler Equation in conservation law form is $\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = \mathbf{0}$

Where
$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix} \quad , \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{bmatrix}.$$

The Jacobian matrix **A** is
$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2}(\gamma-3)u^2 & (3-\gamma)u & \gamma-1 \\ \frac{1}{2}(\gamma-2)u^3 - \frac{a^2 u}{\gamma-1} & \frac{3-2\gamma}{2}u^2 + \frac{a^2}{\gamma-1} & \gamma u \end{bmatrix}.$$

Splitting Jacobian matrices as $\hat{\mathbf{A}}^+ = \dfrac{\partial \mathbf{F}^+}{\partial \mathbf{U}} \ , \quad \hat{\mathbf{A}}^- = \dfrac{\partial \mathbf{F}^-}{\partial \mathbf{U}}$, where **F** is a function of density, sound speed and Mach number

$$\mathbf{F} = \mathbf{F}(\rho, a, M) = \begin{bmatrix} \rho a M \\ \rho a^2 (M^2 + \frac{1}{\gamma}) \\ \rho a^3 M (\frac{1}{2}M^2 + \frac{1}{\gamma-1}) \end{bmatrix}$$

Then

$$\mathbf{F}^+ = \frac{1}{4}\rho a (1+M)^2 \begin{bmatrix} 1 \\ \frac{2a}{\gamma}(\frac{\gamma-1}{2}M+1) \\ \frac{2a^2}{\gamma^2-1}(\frac{\gamma-1}{2}M+1)^2 \end{bmatrix} \quad \mathbf{F}^- = -\frac{1}{4}\rho a (1-M)^2 \begin{bmatrix} 1 \\ \frac{2a}{\gamma}(\frac{\gamma-1}{2}M-1) \\ \frac{2a^2}{\gamma^2-1}(\frac{\gamma-1}{2}M-1)^2 \end{bmatrix}$$

3. Source codes:

```

!!!     This program solves Riemann problem for the Euler equations
!!!     using first-order upwind methods based on VanLeer flux vector splitting.
!!!     This work is licensed under the Creative Commons Attribution-NonCommercial 3.0 Unported License.
!!!     Ao Xu, Profiles: <http://www.linkedin.com/pub/ao-xu/30/a72/a29>

!!!
!!!                                  Shock Tube
!!!     -------------------------|-------------------------
!!!     |                        |                        |
!!!     |                        |                        |
!!!     |                        |                        |
!!!     -------------------------|-------------------------
!!!                                  Contact Discontinuity
!!!

!!!     x1,x1----------------Left/Right side
!!!     diaph----------------Initial discontinuity
!!!     pl,pr----------------Left/Right side pressure
!!!     rhol,rhor------------Left/Right side density
!!!     ul,ur----------------Left/Right side velocity
!!!     al,ar----------------Left/Right side local speed of sound

        program main
        implicit none
        integer, parameter :: N=1000
        integer :: i, itert, itc
        real(8), parameter :: gamma=1.4d0, R=287.14d0
        real(8) :: X(0:N+2), p(0:N+2), a(0:N+2), u(0:N+2), u1(0:N+2), u2(0:N+2), u3(0:N+2), f1(0:N+1), f2(0:N+1), f3(0:N+1)
        real(8) :: x1, x2, dx, t, dt, lambda, diaph
        real(8) :: pl, pr, ul, ur, al, ar, rul, rur, retl, retr,rhol, rhor

        !!! input initial data
        x1 = 0.0d0
        x2 = 1.0d0
        dx = (x2-x1)/float(N)
        t = 0.25d0
        dt = 1e-4
        itert = NINT(t/dt)
        lambda = dt/dx
```

```
    diaph = 0.5d0
    itc = 0
    pl = 1.0d0
    pr = 0.1d0
    rhol = 1.0d0
    rhor =0.125d0
    ul = 0.0d0
    ur = 0.0d0
    al = SQRT(gamma*pl/rhol)
    ar = SQRT(gamma*pr/rhor)

    !!! convert primitive variables to conservative variables
!!!   rul,rur--------------Left/Right side   rho*u
!!!   retl,retr------------Left/Right side   rho*E = rho*(e+0.5*u^2)= rho*(p/rho/(gamma-1)+0.5*u^2) = p/(gamma-1)+0.5*u^2*rho
!!!   E = e+0.5*u^2           e = p/rho/(gamma-1)
    rul = rhol*ul
    rur = rhor*ur
    retl = 0.5d0*rul*ul+pl/(gamma-1.0d0)
    retr = 0.5d0*rur*ur+pr/(gamma-1.0d0)

    !!! construct initial conditions
    call initial(N,X,dx,diaph,rhol,rhor,rul,rur,retl,retr,u1,u2,u3)

    t = 0.0d0
    do itc =1,itert
        !!! find conservative numerical fluxes
        t = t+dt
        do i =0,N+1
            call VanLeer(gamma,u1(i),u2(i),u3(i),u1(i+1),u2(i+1),u3(i+1),f1(i),f2(i),f3(i))
        enddo

        !!! update conserved variables
        do i=1,N+1
            u1(i) = u1(i)-lambda*(f1(i)-f1(i-1))
            u2(i) = u2(i)-lambda*(f2(i)-f2(i-1))
            u3(i) = u3(i)-lambda*(f3(i)-f3(i-1))
            u(i) = u2(i)/u1(i)
            p(i) = (gamma-1.0d0)*(u3(i)-0.5d0*u2(i)*u2(i)/u1(i))
            a(i) = SQRT(gamma*p(i)/u1(i))
        enddo
    enddo
```

```

    write(*,*) 'FVS Applied to the Euler Equations(The van Leer Splitting)'
    write(*,*) 'dt=',dt
    write(*,*) 'dx=',dx
    write(*,*) 'Final time = ',t

    open(unit=02,file='./shock_tube_vanLeer.dat',status='unknown')
    write(02,101)
    write(02,102)
    write(02,103) N
    do i = 1,N
            p(i) = (gamma-1.0d0)*(u3(i)-0.5d0*u2(i)*u2(i)/u1(i))
            a(i) = SQRT(gamma*p(i)/u1(i))
            u(i) = u2(i)/u1(i)
            write(02,100) X(i), u1(i), p(i) ,u(i)
    enddo

100 format(2x,10(e12.6,'      '))
101 format('Title="Sod Shock Tube"')
102 format('Variables=x,rho,p,u')
103 format('zone',1x,'i=',1x,i5,2x,'f=point')

    close(02)
    write(*,*) 'Data export to ./shock_tube_vanLeer.dat file!'

    stop
    end program main



    subroutine initial(N,X,dx,diaph,rhol,rhor,rul,rur,retl,retr,u1,u2,u3)
    implicit none
    integer :: i, N
    real(8) :: dx, diaph, rhol, rhor, rul, rur, retl, retr
    real(8) :: X(0:N+2),u1(0:N+2), u2(0:N+2), u3(0:N+2)

    do i = 0,N+2
        X(i) = i*dx
        if(X(i).LT.diaph) then
            u1(i) = rhol
            u2(i) = rul
            u3(i) = retl
        elseif(X(i).GE.diaph) then
```

```fortran
                        u1(i) = rhor
                        u2(i) = rur
                        u3(i) = retr
                endif
        enddo

        end subroutine initial



        subroutine VanLeer(gamma,rl,rul,retl,rr,rur,retr,f1,f2,f3)
        implicit none
        real(8) :: gamma, rl, rul, retl, rr, rur, retr, f1, f2, f3
        real(8) :: rhol, rhor, ul, ur, pl, pr, hl, hr, al, ar, Ml, Mr
        real(8) :: Mp, Mm, tp, tm, fp1, fp2, fp3, fm1, fm2, fm3

        !!! convert conservative variables to primitive variables
        !!!     rul,rur--------------Left/Right side   rho*u
        !!!     retl,retr------------Left/Right side   rho*E
        !!!     hl,hr----------------Left/Right side   H = E+p/rho = (rho*E+p)/rho
        rhol = rl
        rhor = rr
        ul = rul/rhol
        ur = rur/rhor
        pl = (gamma-1.0d0)*(retl-0.5d0*rul*rul/rhol)
        pr = (gamma-1.0d0)*(retr-0.5d0*rur*rur/rhor)
        hl = (retl+pl)/rhol
        hr = (retr+pr)/rhor
        al = SQRT(gamma*pl/rhol)
        ar = SQRT(gamma*pr/rhor)
        Ml = ul/al
        Mr = ur/ar

        !!! compute positive flux splitting
        if(Ml.LE.-1.0d0) then
                fp1 = 0.0d0
                fp2 = 0.0d0
                fp3 = 0.0d0
        elseif(Ml.LT.1.0d0) then
                Mp = 0.25d0*(Ml+1.0d0)*(Ml+1.0d0)
                tp = (gamma-1.0d0)*ul+2.0d0*al
                fp1 = rhol*al*Mp     !!! 0.25*rho*a*(1+M)^2
                fp2 = fp1*tp/gamma     !!! 0.25*rho*a*(1+M)^2*[2*a/gamma*((gamma-1)/2*M+1)]
```

```
            fp3 = 0.5d0*fp1*tp*tp/(gamma*gamma-1.0d0)    !!! 0.25*rho*a*(1+M)^2*[2*a^2/(gamma^2-1)*((gamma-1)/2*M+1)^2]
    else
            fp1 = rul    !!! rho*u
            fp2 = rul*ul+pl    !!! rho*u*u+p
            fp3 = rhol*hl*ul    !!! rho*u*H
    endif

    !!! compute negative flux splitting
    if(Mr.LE.-1.0d0) then
            fm1 = rur
            fm2 = rur*ur+pr
            fm3 = rhor*hr*ur
    elseif(Mr.LT.1.0d0) then
            Mm = -0.25d0*(Mr-1.0d0)*(Mr-1.0d0)
            tm = (gamma-1.0d0)*ur-2.0d0*ar
            fm1 = rhor*ar*Mm    !!! -0.25*rho*a*(1-M)^2
            fm2 = fm1*tm/gamma    !!! -0.25*rho*a*(1-M)^2*[2*a/gamma*((gamma-1)/2*M-1)]
            fm3 = 0.5d0*fm1*tm*tm/(gamma*gamma-1.0d0)    !!! -0.25*rho*a*(1-M)^2*[2*a^2/(gamma^2-1)*((gamma-1)/2*M-1)^2]
    else
            fm1 = 0.0d0
            fm2 = 0.0d0
            fm3 = 0.0d0
    endif

    !!! compute conserative numerical fluxes
    f1 = fp1+fm1
    f2 = fp2+fm2
    f3 = fp3+fm3

    return
    end subroutine VanLeer
```