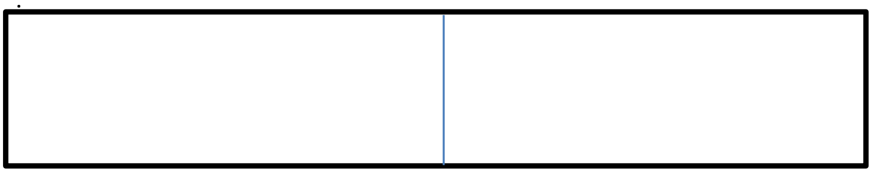
Roe Scheme for Sod Shock Tube Problem

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

Introducing the Jacobian matrix $\mathbf{A}(\mathbf{U}) = \dfrac{\partial \mathbf{F}}{\partial \mathbf{U}}$, get $\mathbf{U}_t + \mathbf{A}(\mathbf{U})\mathbf{U}_x = \mathbf{0}$.

Replace the Jacobian matrix by a constant Jacobian matrix $\tilde{\mathbf{A}} = \tilde{\mathbf{A}}(\mathbf{U}_L, \mathbf{U}_R)$, then the original Riemann problem is replaced by the approximate Riemann problem.

$$\left.\begin{array}{l} \mathbf{U}_t + \tilde{\mathbf{A}}\mathbf{U}_x = \mathbf{0} \\ \mathbf{U}(x,0) = \begin{cases} \mathbf{U}_L \ , \ x<0 \\ \mathbf{U}_R \ , \ x>0 \end{cases} \end{array}\right\}$$

The numerical flux $\mathbf{F}_{i+\frac{1}{2}} = \mathbf{F}_L + \sum\limits_{\tilde{\lambda}_i \le 0} \tilde{\alpha}_i \tilde{\lambda}_i \tilde{\mathbf{K}}^{(i)}$

an alternative choice $\mathbf{F}_{i+\frac{1}{2}} = \mathbf{F}_R - \sum\limits_{\tilde{\lambda}_i \ge 0} \tilde{\alpha}_i \tilde{\lambda}_i \tilde{\mathbf{K}}^{(i)}$.

another alternative choice $$\mathbf{F}_{i+\frac{1}{2}} = \frac{1}{2}(\mathbf{F}_L + \mathbf{F}_R) - \frac{1}{2}\sum_{i=1}^{m} \tilde{\alpha}_i |\tilde{\lambda}_i| \tilde{\mathbf{K}}^{(i)}$$

The eigenvalues of $\tilde{\mathbf{A}}$ are $\tilde{\lambda}_1 = \tilde{u} - \tilde{a}\ , \ \ \tilde{\lambda}_2 = \tilde{\lambda}_3 = \tilde{\lambda}_4 = \tilde{u}\ , \ \ \tilde{\lambda}_5 = \tilde{u} + \tilde{a}$

and the corresponding right eigenvectors are

$$\tilde{\mathbf{K}}^{(1)} = \begin{bmatrix} 1 \\ \tilde{u} - \tilde{a} \\ \tilde{v} \\ \tilde{w} \\ \tilde{H} - \tilde{u}\tilde{a} \end{bmatrix} ; \ \tilde{\mathbf{K}}^{(2)} = \begin{bmatrix} 1 \\ \tilde{u} \\ \tilde{v} \\ \tilde{w} \\ \frac{1}{2}\tilde{V}^2 \end{bmatrix} ; \ \tilde{\mathbf{K}}^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \tilde{v} \end{bmatrix}$$

$$\tilde{\mathbf{K}}^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \tilde{w} \end{bmatrix} ; \ \tilde{\mathbf{K}}^{(5)} = \begin{bmatrix} 1 \\ \tilde{u} + \tilde{a} \\ \tilde{v} \\ \tilde{w} \\ \tilde{H} + \tilde{u}\tilde{a} \end{bmatrix} .$$

where the Roe average values

$$\tilde{u} = \frac{\sqrt{\rho_L} u_L + \sqrt{\rho_R} u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} ,$$

$$\tilde{v} = \frac{\sqrt{\rho_L} v_L + \sqrt{\rho_R} v_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} ,$$

$$\tilde{w} = \frac{\sqrt{\rho_L} w_L + \sqrt{\rho_R} w_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} ,$$

$$\tilde{H} = \frac{\sqrt{\rho_L} H_L + \sqrt{\rho_R} H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} ,$$

$$\tilde{a} = \left((\gamma - 1)(\tilde{H} - \tfrac{1}{2}\tilde{\mathbf{V}}^2)\right)^{\frac{1}{2}}$$

Compute wave strengths $\tilde{\alpha}_3 = \Delta u_3 - \tilde{v}\Delta u_1\ ; \ \ \tilde{\alpha}_4 = \Delta u_4 - \tilde{w}\Delta u_1\ .$

$$\tilde{\alpha}_2 = \frac{\gamma - 1}{\tilde{a}^2}\left[\Delta u_1(\tilde{H} - \tilde{u}^2) + \tilde{u}\Delta u_2 - \overline{\Delta u}_5\right]$$

$$\tilde{\alpha}_1 = \frac{1}{2\tilde{a}}\left[\Delta u_1(\tilde{u} + \tilde{a}) - \Delta u_2 - \tilde{a}\tilde{\alpha}_2\right] ,$$

$$\tilde{\alpha}_5 = \Delta u_1 - (\tilde{\alpha}_1 + \tilde{\alpha}_2) ,$$

3. Source codes:

```
!!!     This program solves Riemann problem for the Euler equations using Roe Scheme
!!!     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
```

```fortran
        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
        do itc =1,itert
                !!! find conserative numerical fluxes
                t = t+dt
                do i =0,N+1
                        call Roe(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(*,*) "Roe's first-order upwind methods"
        write(*,*) 'dt=',dt
        write(*,*) 'dx=',dx
        write(*,*) 'Final time = ',t

        open(unit=02,file='./shock_tube_Roe.dat',status='unknown')
        write(02,101)
        write(02,102)
        write(02,103) N

        do i = 1,N+1
                p(i) = (gamma-1.0)*(u3(i)-0.5*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_Roe.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
                u1(i) = rhor
                u2(i) = rur
                u3(i) = retr
            endif
        enddo

        end subroutine initial



        subroutine Roe(gamma,r4,ru4,ret4,r1,ru1,ret1,f1,f2,f3)
        implicit none
        real(8) :: gamma, p1, p4, u1, u4, f1, f2, f3, ru1, ru4, ret1, ret4
        real(8) :: r1, r4, rho1, rho4, h1, h4, dv1, dv2, dv3
        real(8) :: lambda1, lambda2, lambda3, uavg, havg, aavg, rhoavg, rr


!!! Convert conservative variables to primitive variables.
        rho1 = r1
        rho4 = r4
        u1 = ru1/rho1
        u4 = ru4/rho4
        p1 = (gamma-1.0d0)*(ret1-0.5d0*ru1*ru1/rho1)
        p4 = (gamma-1.0d0)*(ret4-0.5d0*ru4*ru4/rho4)
        h1 = (ret1+p1)/rho1
        h4 = (ret4+p4)/rho4

!!! Step1: Compute Roe average values
        rr = SQRT(rho1/rho4)
        rhoavg = rr*rho4
        uavg = (u4+u1*rr)/(1.0d0+rr)
        havg = (h4+h1*rr)/(1.0d0+rr)
        aavg = SQRT((gamma-1.0d0)*(havg-0.5d0*uavg*uavg))

!!! Step2: Compute the eigencvalues lambda_i
        lambda1 = uavg
```

```
        lambda2 = uavg+aavg
        lambda3 = uavg-aavg

        !!!! Step3: Compute wave strengths
        dv1 = (rho1-rho4)-(p1-p4)/(aavg*aavg)
        dv2 = (u1-u4)+(p1-p4)/(rhoavg*aavg)
        dv3 = (u1-u4)-(p1-p4)/(rhoavg*aavg)

        !!! Step4: Compute numerical fluxes(wave strengths*eigenvalues*right eigenvectors)

    !!!!!!!!!!!!!!!!!!!!!!!!!Alternative and equivalent flux formulae!!!!!!!!!!!!!!!!!!!!!!!!!!
        f1 = ru4&
      &     +dv1*MIN(0.0d0,lambda1)&
      &     +dv2*MIN(0.0d0,lambda2)*0.5d0*rhoavg/aavg&
      &     +dv3*ABS(MIN(0.0d0,lambda3))*0.5d0*rhoavg/aavg
        f2 = ru4*u4+p4&
      &     +dv1*MIN(0.0d0,lambda1)*uavg&
      &     +dv2*MIN(0.0d0,lambda2)*(uavg+aavg)*0.5d0*rhoavg/aavg&
      &     +dv3*ABS(MIN(0.0d0,lambda3))*(uavg-aavg)*0.5d0*rhoavg/aavg
        f3 = ret4*u4+p4*u4&
      &     +dv1*MIN(0.0d0,lambda1)*0.5d0*uavg*uavg&
      &     +dv2*MIN(0.0d0,lambda2)*(havg+aavg*uavg)*0.5d0*rhoavg/aavg&
      &     +dv3*ABS(MIN(0.0d0,lambda3))*(havg-aavg*uavg)*0.5d0*rhoavg/aavg


    !!!!!!!!!!!!!!!!!!!!!!!!!Alternative and equivalent flux formulae!!!!!!!!!!!!!!!!!!!!!!!!!!
    !   f1 = ru1&
    !  &     -dv1*MAX(0.0d0,lambda1)&
    !  &     -dv2*MAX(0.0d0,lambda2)*0.5d0*rhoavg/aavg&
    !  &     -dv3*ABS(MAX(0.0d0,lambda3))*0.5d0*rhoavg/aavg
    !   f2 = ru1*u1+p1&
    !  &     -dv1*MAX(0.0d0,lambda1)*uavg&
    !  &     -dv2*MAX(0.0d0,lambda2)*(uavg+aavg)*0.5d0*rhoavg/aavg&
    !  &     -dv3*ABS(MAX(0.0d0,lambda3))*(uavg-aavg)*0.5d-*rhoavg/aavg
    !   f3 = ret1*u1+p1*u1&
    !  &     -dv1*MAX(0.0d0,lambda1)*0.5d0*uavg*uavg&
    !  &     -dv2*MAX(0.0d0,lambda2)*(havg+aavg*uavg)*0.5d0*rhoavg/aavg&
    !  &     -dv3*ABS(MAX(0.0d0,lambda3))*(havg-aavg*uavg)*0.5d0*rhoavg/aavg
```

```
        return
        end subroutine Roe
```