1–6 (2019) 22 July 2019

Toro Notes - Chapter 4, Exact Solver

Amy Zou

June 11 2019

An exact Riemann solver, key issues:

- the variables selected
- the equations used
- the number of equations and the technique of the iterative procedure
- the initial guess and the handling of nonphysical iterates

1 SOLUTION STRATEGY

Riemann problem for 1-D, time-dependent Euler equations. Choose a different set of primitive variables instead of the conserved quantities **U**:

$$\mathbf{W} = (\rho, u, p)^T \tag{1}$$

Now the initial data are $\mathbf{W}_L = (\rho_L, u_L, p_L)^T$ to the left, and $\mathbf{W}_R = (\rho_R, u_R, p_R)^T$ to the right.

EOS in this chapter are restricted to the caloric EOS $e(\rho, p)$,

$$\text{idealgas} : e = \frac{p}{(\gamma - 1)\rho}, \tag{2}$$

$$\text{covolumegas} : e = \frac{p(1 - b\rho)}{(\gamma - 1)\rho} \tag{3}$$

where b is the covolume constant.

Exact solution if no vacuum present has three waves, with eigenvalues
$\lambda_1 = u - a$, $\lambda_2 = u$, and $\lambda_3 = u + a$;
Note, the speeds of these waves are not in general the characteristics speeds given by the eigenvalues.

The middle wave is a contact discontinuity, and the region on its left and right are the unknown. The left and right (nonlinear) waves are either shock or rarefaction waves. Therefore, four possible wave patterns for our solution scheme (see the book for further comments).

From Section §3.1.3, pressure and particle velocity are constant in the unknown region, while the density takes on two constant values. Finally, the unknown physical quantities we are solving for are:
p_*, u_*, ρ_{*L}, and ρ_{*R}.

2 EQUATIONS FOR PRESSURE AND PARTICLE VELOCITY

type in note from the blue notebook.

3 NUMERICAL SOLUTION FOR PRESSURE

type in note from the blue notebook.

4 THE COMPLETE SOLUTION

Shock Wave (upper sign for Left, and lower for Right)

$$\rho_{*K} = \rho_K \left[\frac{\frac{p_*}{p_K} + \frac{\gamma-1}{\gamma+1}}{\frac{\gamma-1}{\gamma+1}\frac{p_*}{p_K} + 1} \right] \tag{4}$$

shock speed:

$$S_K = u_K \mp Q_K/\rho_K \tag{5}$$

where:

$$Q_K = \left[\frac{p_* + B_K}{A_K} \right]^{\frac{1}{2}} \tag{6}$$

or, compute directly:

$$S_K = u_K \mp a_K \left[\frac{\gamma + 1}{2\gamma} \frac{p_*}{p_K} + \frac{\gamma + 1}{2\gamma} \right]^{\frac{1}{2}} \tag{7}$$

Rarefaction Wave (upper sign for Left, and lower for Right)

$$\rho_{*K} = \rho_K \left(\frac{p_*}{p_K} \right)^{\frac{1}{\gamma}} \tag{8}$$

sound speed behind the rarefaction:

$$a_{*K} = a_K \left(\frac{p_*}{p_K} \right)^{\frac{\gamma-1}{2\gamma}} \tag{9}$$

The rarefaction is enclosed by the Head and the Tail:

$$S_{HK} = u_K \mp a_K \,, \; S_{TK} = u_* \mp a_{*K} \tag{10}$$

For any point (x, t) within the fan region between the Head and the Tail, the solution for $W_{Kfan} = (\rho, u, p)^T$ (T means transpose here):

$$\begin{aligned} \rho &= \rho_K \left[\frac{2}{\gamma + 1} \pm \frac{\gamma - 1}{(\gamma + 1)a_K} \left(u_K - \frac{x}{t} \right) \right]^{\frac{2}{\gamma-1}}, \\ u &= \frac{2}{\gamma + 1} \left[\pm a_K + \frac{\gamma - 1}{2} u_K + \frac{x}{t} \right], \\ p &= p_K \left[\frac{2}{\gamma + 1} \pm \frac{\gamma - 1}{(\gamma + 1)a_K} \left(u_K - \frac{x}{t} \right) \right]^{\frac{2\gamma}{\gamma-1}} \end{aligned} \tag{11}$$

5 SAMPLING THE SOLUTION

See the code `Subroutine sample`. Flow chart on left and right sides is in the black notebook.

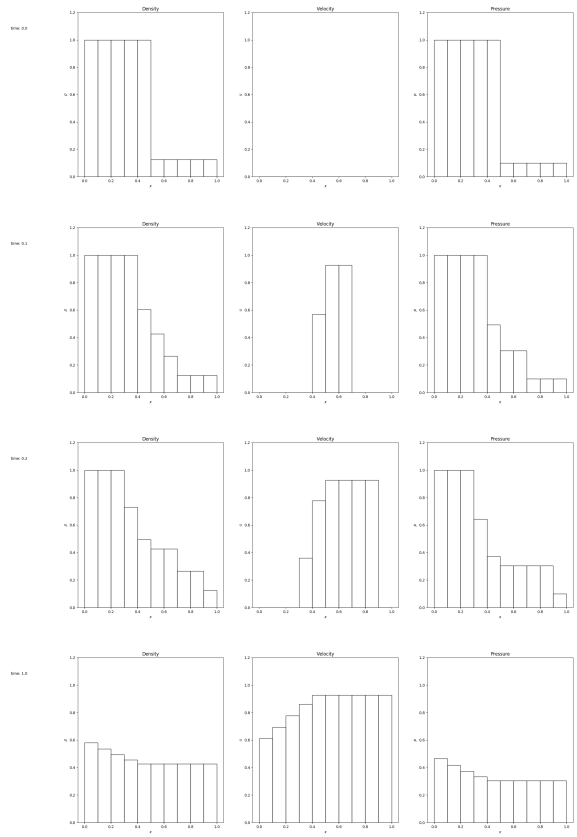

Figure 1. Test 1. Sod test problem. Left rarefaction and right shock.

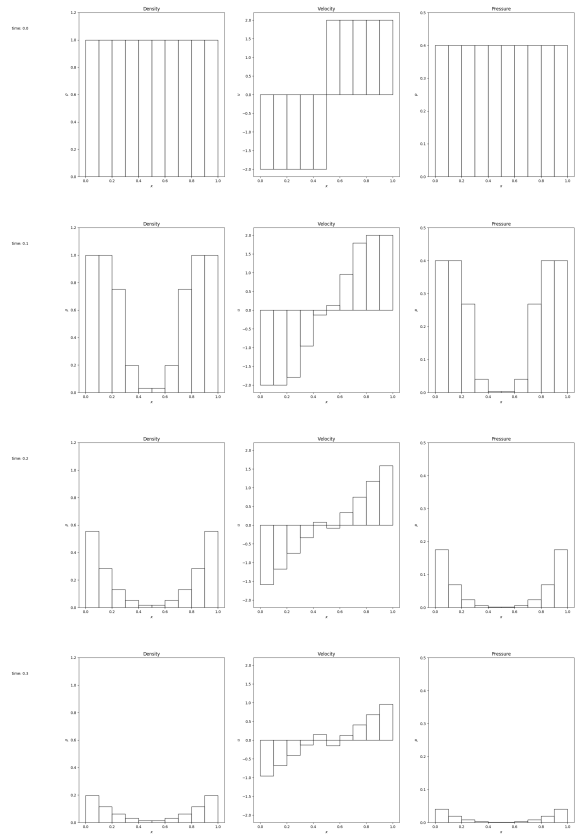

Figure 2. Test 2. 123 problem. Rarefaction on both sides, stationary contact discontinuity, cstar and rarefaction tail speed are both 0.

6 TEST RESULTS

See the five figures. Note different time steps are used in each test. Data can be found in ~/out/output.data

7 THE CODE

```
 program Riemann_solver

implicit none

! declare variables
INTEGER :: openstatus, readstatus
INTEGER :: cells, frames ! nubmer of cells in the
x-axis, and number of output frames in time
REAL(kind=8) :: DomLen, DisctPos, TimeOut, MPA !
read from data file
REAL(kind=8) :: rhoL, uL, pL, rhoR, uR, pR ! ini-
tial conditiosn
REAL(kind=8) :: pStar, uStar, rhoStarL, rhoStarR !
unknowns
REAL(kind=8) :: gamma ! EOS constant
REAL(kind=8) :: AL, BL, AR, BR ! constants in the
pressure sulution
REAL(kind=8) :: cL, cR ! soundspeed
REAL(kind=8) :: p0 ! guessed initial p value
REAL(kind=8) :: fnL, fnDL, fnR, fnDR ! pressure
functions and their derivatives
REAL(kind=8) :: deltau ! difference of the initial
particle speeds
INTEGER :: i=0, j=0, k=1 ! iteration counters
REAL(kind=8) :: pold, pnew, CHA, TOL=1d-6 ! vari-
ables for the iteration process
REAL(kind=8) :: dx, dt
REAL(kind=8) :: S, t, x, rhosol, usol, psol ! out-
put variables for the solution
CHARACTER(len=18) :: filename

! read in initial data
open (unit=10, file='exact.ini', status='old', ac-
tion='read', &
position='rewind', iostat=openstatus)
if (openstatus > 0) stop 'Cannot open file.'

read (10, *, iostat=readstatus) DomLen ! Domain
length
read (10, *, iostat=readstatus) DisctPos ! Initial
discontinuity position
read (10, *, iostat=readstatus) cells ! Number of
computing cells
read (10, *, iostat=readstatus) frames ! Number of
output frames
read (10, *, iostat=readstatus) gamma ! Adiabatic
index
read (10, *, iostat=readstatus) TimeOut ! Output
```

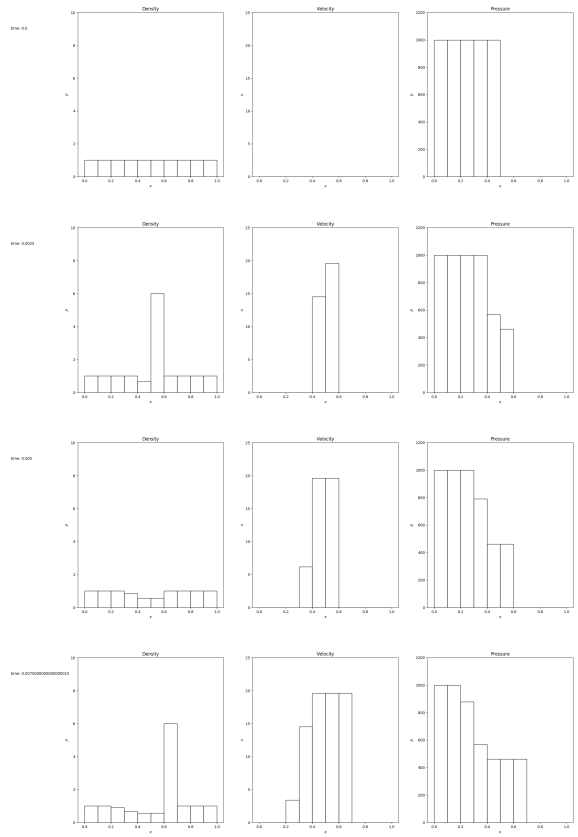

Figure 3. Test 3. Left rarefaction and right shock.

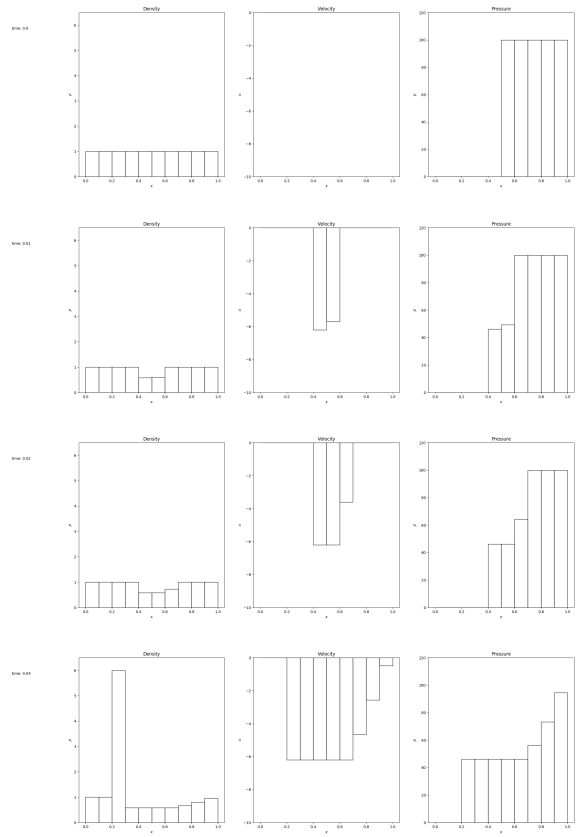

Figure 4. Test 4. Left shock and right rarefaction.

```
time
read (10, *, iostat=readstatus) rhoL ! Initial
density on the left
read (10, *, iostat=readstatus) uL ! Initial ve-
locity on the left
read (10, *, iostat=readstatus) pL ! Initial pres-
sure on the left
read (10, *, iostat=readstatus) rhoR ! Initial
density on the right
read (10, *, iostat=readstatus) uR ! Initial ve-
locity on the right
read (10, *, iostat=readstatus) pR ! Initial pres-
sure on the right
read (10, *, iostat=readstatus) MPA ! Normalising
constatn

close (10)

! compute the sound speeds
cL = Sqrt(gamma*pL/rhoL) ! Sound speed on the left
cR = Sqrt(gamma*pR/rhoR) ! Sound speed on the
right


! compute the data dependent constants
AL = 2.0/((gamma + 1.)*rhoL)
BL = (gamma - 1.)*pL/(gamma + 1.)
AR = 2.0/((gamma + 1.)*rhoR)
BR = (gamma - 1.)*pR/(gamma + 1.)

! compute the difference of the initial parti-
cle speed
deltau = uR - uL

! check the pressure positivity condition
if(2.*(cL + cR)/(gamma -1.) .LE. deltau) then
write (*,*) 'Vacuum is generated by data'
write (*,*) 'Program stopped'
STOP
end if

! do the initial guess of pressure
! use the average for now, but do test the other
methods
p0 = (pL + pR)/2.0

! iteration procedure to find pStar
pold = p0
do
call pressfn(pold, pL, AL, BL, cL, gamma, rhoL,
fnL, fnDL)
call pressfn(pold, pR, AR, BR, cR, gamma, rhoR,
fnR, fnDR)
pnew = pold - (fnL + fnR + deltau) / (fnDL + fnDR)
CHA = 2.*abs(pnew - pold)/(pnew + pold)
i = i + 1
```

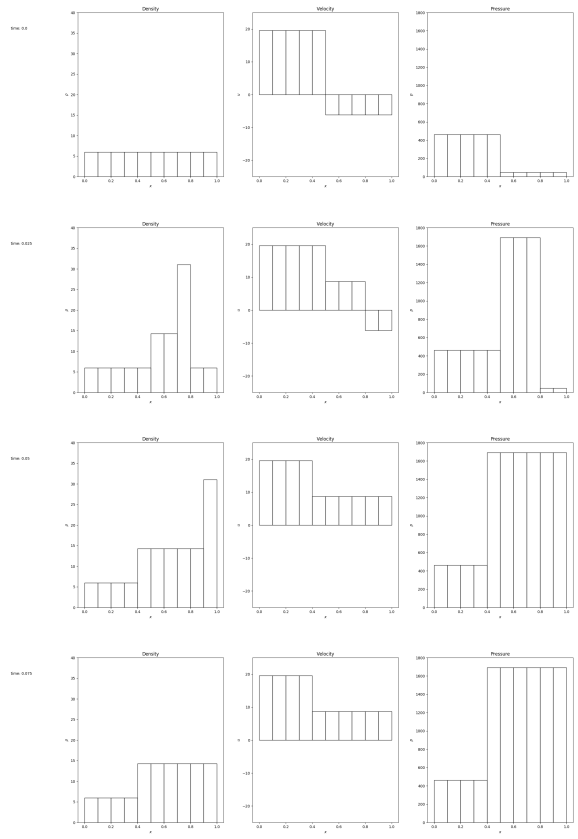

Figure 5. Test 5. Shock on both sides. Right shock is about 10 times faster than the left one.

```
pold = pnew
if (CHA .LE. TOL) exit
end do

pStar = pnew
write (*,*)
write (*,*) 'Solved p-star', pStar, 'with', i,
'iterations.'


! compute uStar
uStar = (uL + uR + fnR - fnL)/2.
write (*,*)
write (*,*) 'Solved u-star', uStar
write (*,*)

! complete solution at TimeOut
dx = DomLen / real(cells)
dt = TimeOut / real(frames)

do j=0, frames, 1
write(filename,fmt='("./out/frame", I3.3,
".out")')j
open(unit=j, file=filename, status='unknown')
t = dt*real(j)

do i=1, cells, 1
x = (real(i) - 0.5)*dx ! position at the center of
each cell
S = (x - DisctPos)/t
call sample(rhoL, uL, pL, cL, rhoR, uR, pR, cR,
gamma, pstar, ustar, S, rhosol, usol, psol)
write (j, *) x, rhosol, usol, psol
end do

close(unit=j)
end do

end program Riemann_solver


! ------------------------------

subroutine pressfn(p, pk, Ak, Bk, ck, g, rhok,
fn, fnD)
implicit none
real(kind=8), intent(in) :: p, pK, Ak, Bk, ck, g,
rhok
real(kind=8), intent(out) :: fn, fnD ! pressure
function and its derivative on one side

if (p .LE. pk) then
fn = 2.*ck*((p/pk)**((g - 1.)/(2.*g)) - 1.)/(g -
1.)
fnD = (p/pk)**(-(g + 1.)/(2.*g))/(rhok*ck)
else
fn = (p - pk)*sqrt(Ak/(p + Bk))
fnD = sqrt(Ak/(Bk + p)) * (1. - (p - pk)/(2.*(Bk +
p)))
end if

return
end subroutine pressfn
! ------------------------------

subroutine sample(rhoL, uL, pL, cL, rhoR, uR, pR,
cR, gamma, pstar, ustar, S, rhosol, usol, psol)
implicit none
real (kind=8), intent(in) :: rhoL, uL, pL, cL,
rhoR, uR, pR, cR, gamma, pstar, ustar
real (kind=8), intent(out) :: rhosol, usol, psol

real (kind=8) :: g1, g2, g3, g4, g5, g6, g7, g8
real (kind=8) :: S, SHL, STL, cstarL, SL, SHR,
STR, cstarR, SR

! gamma related constants
g1 = (gamma - 1.)/(gamma +1.)
g2 = (gamma + 1.)/(2.*gamma)
g3 = (gamma - 1.)/(2.*gamma)
g4 = 1./gamma
g5 = 2./(gamma + 1.)
g6 = 2./(gamma - 1.)
g7 = (gamma - 1.)/2.
g8 = 2*gamma/(gamma - 1.)

open(unit=22, file='./out/output.data', sta-
tus='unknown', position='append')
write(22,*) pstar, ' ! pstar'
```

```
write(22,*) ustar, ' ! ustar'

if (S .LE. ustar) then
! Sampling point lies to the Left of the contact
discontinuity
write(22,*) 'Left'

if (pstar .LE. pL) then
! Left rarefaction
write(22,*) 'Rarefaction'

cstarL = cL*(pstar/pL)**g3
SHL = uL - cL
STL = ustar - cstarL
write(22, *) cstarL, SHL, STL, ' ! cstarL, SHL,
STL'

if (S .LE. SHL) then
! left data state

rhosol = rhoL
usol = uL
psol = pL


else
if (S .GE. STL) then
! star left state

rhosol = rhoL*(pstar/pL)**g4
usol = ustar
psol = pstar

else
! left fan state

rhosol = rhoL* ( g5 + g1*(uL - S)/cL )**g6
usol = g5* ( cL + g7*uL + S )
psol = pL* ( g5 + g1*(uL - S)/cL )**g8
end if
end if

else
! Left Shock
write(22,*) 'Shock'

SL = uL - cL * sqrt( g2*pstar/pL + g2)
write(22,*) SL, ' ! speed of the shock wave'

if (S .LE. SL) then
! left data state


rhosol = rhoL
usol = uL
psol = pL

else
! start left state

rhosol = rhoL* (pstar/PL + g1) / (g1*pstar/pL +
1.)

usol = ustar
psol = pstar
end if
end if

else
! Sampling poitn lies to the Right of the contact
discontiuity
write(22,*) 'Right'

if (pstar .LE. pR) then
! Right rarefaction
write(22,*) 'Rarefaction'

cstarR = cR*(pstar/pR)**g3
SHR = uR + cR
STR = ustar + cstarR
write(22, *) cstarR, SHR, STR, ' ! cstarR, SHR,
STR'

if (S .GE. SHR) then
! right data state

rhosol = rhoR
usol = uR
psol = pR

else
if (S .LE. STR) then
! star right state

rhosol = rhoR*(pstar/pR)**g4
usol = ustar
psol = pstar

else
! right fan state

rhosol = rhoR* ( g5 - g1*(uR - S)/cR )**g6
usol = g5* ( - cR + g7*uR + S )
psol = pR* ( g5 - g1*(uR - S)/cR )**g8
end if
end if

else
! Right Shock
write(22,*) 'Shock'

SR = uR + cR * sqrt( g2*pstar/pR + g2)
write(22,*) SR, ' ! speed of the shock wave'

if (S .GE. SR) then
! Right data state

rhosol = rhoR
usol = uR
psol = pR

else
! start right state

rhosol = rhoR* (pstar/PR + g1) / (g1*pstar/pR +
```

```
1.)
usol = ustar
psol = pstar
end if
end if
end if
close(unit=22)

return

end subroutine sample
```

This paper has been typeset from a TEX/LATEX file prepared by the author.