Solution set 5:

Systems and Approximate Riemann Solvers

Exercise 5.1 **(a)** To write the scheme in explicit form we simply insert the flux in the conservation law and reduce to get

$$U_j^{n+1} = U_j^n - \frac{k}{h}\left[A^-\left(U_{j+1}^n - U_j^n\right) + A^+\left(U_j^n - U_{j-1}^n\right)\right] \tag{1}$$

(b) We know that $A^+ = R\Lambda^+R^{-1}$ and $A^- = R\Lambda^-R^{-1}$, so to find A^- and A^+ we need to first find the eigenvalues and eigenvectors of

$$A = \begin{bmatrix} v_0 & K_0 \\ 1/\rho_0 & v_0 \end{bmatrix} \tag{2}$$

This is fortunately a straight forward exercise, you can find the eigenvalues by solving the characteristic equation $\det(A - \lambda I) = 0$, for each eigenvalue you can find the corrospongding eigenvector by inserting and using Gaussian elemination.

$$\lambda_1 = u_0 - c_0,\ r_1 = \begin{bmatrix} -\rho_0 c_0 \\ 1 \end{bmatrix} \tag{3}$$

$$\lambda_2 = u_0 + c_0,\ r_2 = \begin{bmatrix} \rho_0 c_0 \\ 1 \end{bmatrix} \tag{4}$$

With $c_0 = \sqrt{K_0/\rho_0}$ as in the exercise description. From the above we get R and its inverse R^{-1} to be

$$R = \begin{bmatrix} -\rho_0 c_0 & \rho_0 c_0 \\ 1 & 1 \end{bmatrix} \text{ and } R^{-1} = \frac{1}{2\rho_0 c_0}\begin{bmatrix} 1 & -\rho_0 c_0 \\ -1 & -\rho_0 c_0 \end{bmatrix} \tag{5}$$

As stated in the exercise, you can assume that we are in the case of subsonic flow $u_0 > c_0$, we have

$$\Lambda^- = \begin{bmatrix} \min(u_0 - c_0, 0) & 0 \\ 0 & \min(u_0 + c_0, 0) \end{bmatrix} = \begin{bmatrix} u_0 - c_0 & 0 \\ 0 & 0 \end{bmatrix} \tag{6}$$

and

$$\Lambda^+ = \begin{bmatrix} \max(u_0 - c_0, 0) & 0 \\ 0 & \max(u_0 + c_0, 0) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & u_0 + c_0 \end{bmatrix} \tag{7}$$

We are now ready to assemble A^- and A^+ from the components given in Using eq. 5,6,7. For A^- we have

$$A^- = R\Lambda^-R^{-1} = \frac{u_0 - c_0}{2\rho_0 c_0}\begin{bmatrix} \rho_0 c_0 & -\rho_0^2 c_0^2 \\ -1 & \rho_0 c_0 \end{bmatrix} \tag{8}$$

and for A^+ we arrive at

$$A^+ = R\Lambda^+R^{-1} = \frac{u_0 + c_0}{2\rho_0 c_0}\begin{bmatrix} \rho_0 c_0 & \rho_0^2 c_0^2 \\ 1 & \rho_0 c_0 \end{bmatrix} \tag{9}$$

(c) The CFL condition is a necersssary condition for stability, we must ensure that for each eigenvalue λ_p in A we have $\left|\frac{\lambda_p k}{h}\right| \leq 1$. Since $u_0 > c_0$ we arrive at the condition

$$\frac{(u_0 + c_0)\,k}{h} \leq 1 \ \Rightarrow\ \frac{k}{h} \leq \frac{2}{3} \tag{10}$$

(d) See the Matlab/Octave code attached at the end of this solution manual. **(e)** See figures generated by the Matlab/Octave code. The conservation law is a linear system, we know that in this case shocks move only along characteristics and can not spontaniously arise. If the initial condition contains no shocks, the solution at some timestep T also contains no shocks. When using the second set of initial conditions with discontinuities, these discontinuities seem to disappear as time evolves, this is however due to numerical difussion in the scheme, not because the discontinuty disappear from the actual solution. **(f)** See figures generated by the Matlab/Octave code attached at the end of this solution manual. With this third set of initial conditions, it is not straightforward to locate the shocks as these are smeared out to a point where they are indistinguishable

from other smooth curves in the solution. Using the generalized upwind method for the linear system, we resolve and propagate information in a proper up-winded manner yielding less numerical diffusion than other schemes that return monotone solutions over discontinuities, such as the Lax-Friedrichs method. However, the upwind method is still only a first order method and in general first order methods are not suitable for long time integration or resolving fine details, in later exercises and lectures we will work on developing higher order methods. Note that in the case of a linear system we can actually construct the true solution trough the eigenvalue decomposition by solving the system exactly in a space spanned by the eigenvectors of A and then only later transforming back to to our standard basis. See Matlab/Octave code attached for an example of how to do this.

Exercise 5.2 **(a)** Roe's conditions are the following:

1. The Matrix $\hat{A} = \hat{A}(u_l, u_r)$ satisfies
$$f(u_r) - f(u_l) = \hat{A}(u_r - u_l) \ . \tag{11}$$
2. For each v,
$$\lim_{u_r, u_l \to v} \hat{A}(u_l, u_r) = f'(v) \ . \tag{12}$$
3. For each u_l, and u_r, $\hat{A}(u_l, u_r)$ is diagonalizable and has only real eigenvalues.

(b) See page 150 in LeVenque's book. **(c)** For scalar a conservation law, condition 1. uniquely determines the matrix
$$\hat{a} = \frac{f(u_r) - f(u_l)}{u_r - u_l} \ . \tag{13}$$

(d) Godunov's method requires the calculation of the exact solution of a Riemann problem at each cell interface, and each time step. We already know that it is reasonable to replace this exact solution by an approximation. One method of deriving such approximations is Roe linearization. In this exercise you were asked to derive the numerical flux for a scheme which employs Roe linearization to approximate the solutions of Riemann problems. For the derivation, suppose we wish to approximate the solution u of the Riemann problem which consists of the conservation law
$$u_t + f(u)_x = 0 \ , \tag{14}$$
and the initial condition $u|_{t=0} = u_0$, where u_0 is a piecewise constant function that takes the value u_l at each $x < 0$, and the value u_r at each $x > 0$. Here $x = 0$ corresponds to the cell interface. The general idea of approximate Riemann solvers is to approximate the solution u of a given Riemann problem by an exact solution $\hat{u}$ of a modified problem which consists of a conservation law
$$\hat{u}_t + \hat{f}(\hat{u})_x = 0 \ , \tag{15}$$
and the same initial condition $\hat{u}|_{t=0} = u_0$. In Roe's method $\hat{f}(w)$ is given by $\hat{f}(w) = \hat{A}w$, where $\hat{A}$ is a Roe matrix. We use this fact only near the end of this construction. We start by deriving an expression for Godunov's flux for (14). Let k and M be some positive constants, and suppose that M is sufficiently large compared to k. By integrating (14) over the rectangle $[0, M] \times [0, k]$, we get
$$\begin{aligned} \int_0^M u(x,k)\, \mathrm{d}x &= Mu_r - kf(u_r) + \int_0^M f(u(0,t))\, \mathrm{d}t \\ &= Mu_r - kf(u_r) + kf(u^*(u_l, u_r)) \end{aligned} \tag{16}$$
where u^* determines the numerical flux of Godunov's method. Similarly, by integrating (14) over $[-M, M] \times [0, k]$, we have
$$\int_{-M}^M u(x,k)\, \mathrm{d}x = M(u_r + u_l) - kf(u_r) + kf(u_l) \ . \tag{17}$$
Since in Roe's method we require
$$\hat{f}(u_r) - \hat{f}(u_l) = f(u_r) - f(u_l) \ , \tag{18}$$

$\hat{u}$ also satisfies

$$\int_{-M}^{M} \hat{u}\,(x,k)\ \mathrm{d}x = M\,(u_r + u_l) - kf\,(u_r) + kf\,(u_l)\ . \tag{19}$$

To get our approximation, we replace u in the left hand side of (16) by the approximation $\hat{u}$, which solves the modified conservation law (15). To account for this change we also replace the numerical flux $f\,(u^*\,(u_l,u_r))$ of Godunov's method by the approximate numerical flux $F\,(u_l,u_r)$. Thus, we get

$$\int_{0}^{M} \hat{u}\,(x,k)\ \mathrm{d}x = Mu_r - kf\,(u_r) + kF\,(u_l,u_r)\ , \tag{20}$$

or equivalently

$$F\,(u_l,u_r) = f\,(u_r) - \frac{M}{k}u_r + \frac{1}{k}\int_{0}^{M} \hat{u}\,(x,k)\ \mathrm{d}x\ . \tag{21}$$

It follows from (21), and (19) that F also satisfies

$$F\,(u_l,u_r) = f\,(u_l) + \frac{M}{k}u_l - \frac{1}{k}\int_{-M}^{0} \hat{u}\,(x,k)\ \mathrm{d}x\ . \tag{22}$$

Since $\hat{u}$ is the exact solution of (15), $\hat{u}$ satisfies

$$\begin{aligned}\int_{0}^{M} \hat{u}\,(x,k)\ \mathrm{d}x &= Mu_r - k\hat{f}\,(u_r) + \int_{0}^{k} \hat{f}\,(\hat{u}^*(u_l,u_r))\ \mathrm{d}t \\ &= Mu_r - k\hat{f}\,(u_r) + k\hat{f}\,(\hat{u}^*(u_l,u_r))\ .\end{aligned} \tag{23}$$

By substituting (23) into (21), we get

$$F\,(u_l,u_r) = f\,(u_r) - \hat{f}\,(u_r) + \hat{f}\,(\hat{u}^*(u_l,u_r))\ , \tag{24}$$

which is equivalent to

$$F\,(u_l,u_r) = f\,(u_l) - \hat{f}\,(u_l) + \hat{f}\,(\hat{u}^*(u_l,u_r))\ , \tag{25}$$

by (18). Notice that $\hat{f}\,(\hat{u}^*(u_l,u_r))$ is Godunov's flux for (15). Finally, recalling that in Roe linearization, $\hat{f}\,(v) = \hat{A}v$, we get

$$\hat{u}^*(u_l,u_r) = u_l + \sum_{\lambda_j<0} \alpha_j r_j = u_r - \sum_{\lambda_j>0} \alpha_j r_j\ , \tag{26}$$

where $u_r - u_l = \sum_j \alpha_j r_j$. Therefore,

$$F\,(u_l,u_r) = f\,(u_r) - \hat{A}u_r + \hat{A}\left(u_l + \sum_{\lambda_j<0} \alpha_j r_j\right) = f\,(u_r) - \hat{A}^+\,(u_r - u_l) \tag{27}$$

$$= f\,(u_l) - \hat{A}u_l + \hat{A}\left(u_r - \sum_{\lambda_j>0} \alpha_j r_j\right) = f\,(u_l) + \hat{A}^-\,(u_r - u_l) \tag{28}$$

By taking the average of the right most expressions in (27) and (28), we get an equivalent symmetric expression

$$F\,(u_l,u_r) = \frac{1}{2}\,(f\,(u_r) + f\,(u_l)) - \frac{1}{2}|\hat{A}|\,(u_r - u_l)\ . \tag{29}$$

```
% This script was written for EPFL MATH459, Numerical Methods for Conservation Laws and tested
%with Octave 3.6.4.
% The one dimensional linearized acoustic equations are solved with periodic boundary conditions
% and initial data as given in the exercise.

% Function declarations
function U = Godunov(An,Ap,U1,U2,U3,k,h)
    U = U2-(k/h)*(An*(U3-U2)+Ap*(U2-U1));
end

function U = initialData(data,X,nX)
    U = zeros(2,nX);
    % Initial pressure distribution
    switch data
        case 1
            U(1,:) = sin(2*pi*X);
        case 2
            U(1,find(X<0)) = 0;
            U(1,find(X>=0)) = 1;
        case 3
            U(1,find(X<0)) = 1;
            U(1,find(X>=0)) = sin(2*pi*X(find(X>=0)));
    end
    % Initial velocity distribution
    U(2,find(X<0)) = 0;
    U(2,find(X>=0)) = 0;
end

function [u,M,x] = decomposition(data,R,eign,k)
    h = abs(eign)*k;
    x = -1:h:1;
    nX = numel(x);

    u = initialData(data,x,nX);
    for i = 1:nX
        u(:,i) = inv(R) * u(:,i);
    end
    if eign<0
        M = diag(ones(nX-1,1),1);
        M(end-1,1) = M(1,2);
    elseif eign>0
        M = diag(ones(nX-1,1),-1);
        M(1,end-1) = M(2,1);
    end
end
```

```
% Initial data set
data = 3;

% Physical constants
u0 = 1/2;
k0 = 1;
p0 = 1;
c0 = sqrt(k0/p0);
z0 = p0*c0;

% Define Discretization and initial data
h = 0.01;
k = 0.6*h;
X = -1:h:1;
nX = numel(X);
T = 0:k:1.2;
nT = numel(T);
U = initialData(data,X,nX);

% Flux matrices
An = (u0-c0)/(2*z0) * [ z0 , -z0^2 ; -1 , z0 ];
Ap = (u0+c0)/(2*z0) * [ z0 ,  z0^2 ; 1 , z0 ];

% Eigenvalue decomposition so to construct the true solution
eign = [u0-c0;u0+c0];
R = [ -z0 , z0 ; 1 , 1 ];
[up,Mp,Xp] = decomposition(data,R,eign(1),k);
[uv,Mv,Xv] = decomposition(data,R,eign(2),k);

% Solve
for i = 1:nT
    % Generalized upwind / Godunov's method on a linear system of equations
    UT = U;
    U(:,1) = Godunov(An,Ap,UT(:,nX-1),UT(:,1),UT(:,2),k,h);
    for j = 2:nX-1
        U(:,j) = Godunov(An,Ap,UT(:,j-1),UT(:,j),UT(:,j+1),k,h);
    end
    U(:,nX) = U(:,1);

    % Reconstruction of the true solution at t = i*k
    up(1,:) = (Mp*up(1,:)')';
    uv(2,:) = (Mv*uv(2,:)')';
    u = [up(1,:);interp1(Xv,uv(2,:),Xp)];
    for i = 1:numel(Xp)
        u(:,i) = R * u(:,i);
    end

    % Visualize the solution
    plot(X,U(1,:),'-r',X,U(2,:),'-b',Xp,u(1,:),'-k',Xp,u(2,:),'-k');
    ylim([-2 2]);xlim([-1 1]);
    set(gca,'XTick',-1:0.5:1);set(gca,'YTick',-2:0.5:2)
    grid on;title('Test');
    legend('Pressure','Velocity','Solution')
    drawnow
end
```