Exercise Set 9: Linear Systems

Exercise 1
This exercise pertains to *Godunov's method* for linear systems with constant coefficients,

$$q_t + Aq_x = 0 \; . \tag{1}$$

On linear systems, Godunov's method reduces to a generalization of the upwind method where the numerical flux is given by the following equivalent expressions

$$\begin{aligned} F\left(Q_l, Q_r\right) &= AQ_l + A^- \left(Q_r - Q_l\right) & (2) \\ &= AQ_r - A^+ \left(Q_r - Q_l\right) = \frac{1}{2}A\left(Q_r + Q_l\right) - \frac{1}{2}|A|\left(Q_r - Q_l\right) \; . & (3) \end{aligned}$$

Where $|A| = A^+ - A^-$ and $A^\pm = S\Lambda^\pm S^{-1}$. Here Λ^+ and Λ^- are diagonal matrices with non-negative and non-positive entries, respectively, such that $S^{-1}AS = \Lambda$ is the spectral decomposition of A, with $\Lambda = \Lambda^+ + \Lambda^-$. Especially, notice that

$$A = A^+ + A^- \; . \tag{4}$$

Consider the one dimensional acoustics equation

$$\begin{pmatrix} p \\ v \end{pmatrix}_t + \begin{pmatrix} u_0 & K_0 \\ 1/\rho_0 & u_0 \end{pmatrix} \begin{pmatrix} p \\ v \end{pmatrix}_x = 0 \; . \tag{5}$$

This system is derived from the nonlinear Euler equation by linearizing around some fixed state, as sound waves are small perturbation in a background media. Here, K_0 is the compressibility modulus, and u_0 and ρ_0 are the the velocity and pressure, respectively. The speed of sound in the medium is given by

$$c_0 = \sqrt{K_0/\rho_0} \; . \tag{6}$$

1. For (5), calculate A^+ and A^-.
2. What is the CFL condition of Godunov's method for (5)?
3. Implement Godunov's method for (5) the following two sets of initial data

$$p(x,0) = \sin\left(2\pi x\right) \;, \quad v(x,0) = 0, \quad \text{with periodic BC} \tag{7}$$

$$p(x,0) = \begin{cases} 0 & x < 0 \\ 1 & x > 0 \end{cases} \;, \quad v(x,0) = 0,, \quad \text{with open BC.} \tag{8}$$

   Use $u_0 = 1/2$, $K_0 = 1$, $p_0 = 1$ and solve on the interval $x \in [-1,1]$ with $h = 0.01$ to $T = 0.4$ and an appropriate time-step satisfying the CFL condition.

4. In the solution driven by (8), are the discontinuities visible in the numerical solution at $T = 0.4$? Plot and compare with the exact solution at the final time.
5. Now, run your code with the initial data

$$p(x,0) = \begin{cases} 1 & x < 0 \\ \sin\left(2\pi x\right) & x > 0 \end{cases} \;, \quad v(x,0) = 0 \; . \tag{9}$$

   Does the exact solution preserve the discontinuities present in the initial condition? Are you able to observe the discontinuities in the numerical solution at $T = 0.4$?

Solution 1 1. We start by calculating the spectral decomposition of

$$A = \begin{pmatrix} u_0 & K_0 \\ 1/\rho_0 & u_0 \end{pmatrix} . \tag{10}$$

Since this is a 2×2 matrix, this can be done easily. We have

$$\Lambda_1 = u_0 - c_0, \quad S_1 = (-\rho_0 c_0 \ , \ 1)^T \ , \tag{11}$$

and

$$\Lambda_2 = u_0 + c_0, \quad S_2 = (\rho_0 c_0 \ , \ 1)^T \ , \tag{12}$$

where $c_0 = \sqrt{K_0/\rho_0}$. Thus,

$$S = \begin{pmatrix} -\rho_0 c_0 & \rho_0 c_0 \\ 1 & 1 \end{pmatrix} \quad S^{-1} = \frac{1}{2\rho_0 c_0} \begin{pmatrix} -1 & \rho_0 c_0 \\ 1 & \rho_0 c_0 \end{pmatrix} . \tag{13}$$

The matrices Λ^+ and Λ^- can be expressed as

$$\Lambda^+ = \begin{pmatrix} \Lambda_1^+ & 0 \\ 0 & \Lambda_2^+ \end{pmatrix} \qquad \Lambda^- = \begin{pmatrix} \Lambda_1^- & 0 \\ 0 & \Lambda_2^- \end{pmatrix} \tag{14}$$

where $\Lambda_s^+ = \max(\Lambda_s, 0)$ and $\Lambda_s^- = \min(\Lambda_s, 0)$ for $s = 1, 2$. Thus we obtain

$$A^+ = S\Lambda^+ S^{-1} = \frac{1}{2} \begin{pmatrix} (\Lambda_1^+ + \Lambda_2^+) & \rho_0 c_0(-\Lambda_1^+ + \Lambda_2^+) \\ (\rho_0 c_0)^{-1}(-\Lambda_1^+ + \Lambda_2^+) & (\Lambda_1^+ + \Lambda_2^+) \end{pmatrix} \tag{15}$$

and

$$A^- = S\Lambda^- S^{-1} = \frac{1}{2} \begin{pmatrix} (\Lambda_1^- + \Lambda_2^-) & \rho_0 c_0(-\Lambda_1^- + \Lambda_2^-) \\ (\rho_0 c_0)^{-1}(-\Lambda_1^- + \Lambda_2^-) & (\Lambda_1^- + \Lambda_2^-) \end{pmatrix} . \tag{16}$$

2. The CFL condition is a necessary condition for stability. Here, the CFL condition requires that for each eigenvalue Λ_s of A, the following must hold

$$\frac{|\Lambda_s| k}{h} \leq 1 \ . \tag{17}$$

Making use of the eigenvalue structure for the given system, we arrive at the condition

$$(|u_0| + c_0) \frac{k}{h} \leq 1 \ . \tag{18}$$

3. See the Matlab code attached at the end of this solution manual.

4. See figures generated by the Matlab code. Since the conservation law is a linear system, we know that discontinuities move only along characteristics and can not spontaneously form. If the initial condition is smooth and satisfies the boundary conditions, then the solution is also smooth. When using the second set of initial conditions, which is discontinuous, the discontinuities seem to get smeared out as the scheme advances. This is however due to numerical diffusion in the scheme, and not because discontinuities disappear.

5. See figures generated by the Matlab 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 regions of 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. Note that in the case of a linear system we can actually construct the exact solution. See the Matlab code attached for an example on how to do this.

```matlab
% Solution07
% This script was written for EPFL MATH459, Numerical Methods for
% Conservation Laws.
% The one dimensional linearized acoustic equations are solved with
% periodic/open boundary conditions, and initial data
% as given in the exercise.

clc
clear all
close all

% Initial data set
data = 3;
switch data
case 1
        pIC =@(x) sin(2*pi*x);
        vIC =@(x) 0*x;
        bc  = 'Periodic';
case 2
        pIC =@(x) 1*(x>0);
        vIC =@(x) 0*x;
        bc  = 'Open';
case 3
        pIC =@(x) 1*(x<0) + sin(2*pi*x).*(x>=0);
        vIC =@(x) 0*x;
        bc  = 'Open';
end

% Define Discretization and time parameters
h  = 0.01;
xf = -1:h:1;
xc = (-1+0.5*h):h:(1-0.5*h);
N  = length(xc);
Tfinal = 0.4;
CFL    = 0.5;

% Physical constants [u0,k0,p0]
u0 = 1/2;
K0 = 1;
p0 = 1;
c0 = sqrt(K0/p0);

% Find various matrices
A      = [u0,K0;1/p0,u0];
S      = [-p0*c0, p0*c0; 1,1];
Sinv   = [-1, p0*c0; 1, p0*c0]/(2*p0*c0);
Lambda = [u0-c0,0;0,u0+c0];
absA   = S*abs(Lambda)*Sinv;

% Averaging initial conditions
% Cell-center values sufficient for first-order schemes
U = [pIC(xc);vIC(xc)];
time = 0; iter = 0;
plot_every = 10;

% Solve
```

```
while time < Tfinal

k = CFL*h/(abs(u0) + c0);
if(time + k > Tfinal)
        k = Tfinal - time;
end

% Applying boundary conditions to obtain extended vector
U_ext = apply_bc(U,bc);

Flux = GodunovFlux(A,absA,U_ext(:,1:end-1),U_ext(:,2:end));
U = U - k/h*(Flux(:,2:end) - Flux(:,1:end-1));

time = time + k;
iter = iter + 1;

if(mod(iter,plot_every)==0 || time ==Tfinal)
        % Finding exact solution at the current time
        Uexact = find_exact(pIC,vIC,S,Sinv,Lambda,xc,time);

        % Visualize the solution
        figure(1)
        subplot(2,1,1)
        plot(xc,U(1,:),'-r','LineWidth',2);
        hold all
        plot(xc,Uexact(1,:),'--k','LineWidth',2);
        ylim([-2 2]);xlim([-1 1]);
        legend('Numerical Pressure','Exact Pressure','Location','Best')
        grid on;
        title(['Time = ',num2str(time)])
        hold off

        subplot(2,1,2)
        plot(xc,U(2,:),'-r','LineWidth',2);
        hold all
        plot(xc,Uexact(2,:),'--k','LineWidth',2);
        ylim([-2 2]);xlim([-1 1]);
        legend('Numerical Velocity','Exact Velocity','Location','Best')
        grid on;
        hold off
        end
end

        function Flux = GodunovFlux(A,absA,UL,UR)

        Flux = 0.5*A*(UL+UR) - 0.5*absA*(UR-UL);

        end

% Function returns an extended vector, based on
% the type of boundary condition requested

function U_ext = apply_bc(U,bc)
        switch bc
        case 'Periodic'
                U_ext = [U(:,end) , U, U(:,1)];
        case 'Open'
```

```
                U_ext = [U(:,1)    , U, U(:,end)];
        end

% Function evaluates the exact solution at a given time
function Uexact = find_exact(pIC,vIC,S,Sinv,Lambda,xc,time)
        VSelect = @(V,IDX) V(IDX,:);
        V0 =@(x) Sinv*[pIC(x);vIC(x)];
        VT_1 =@(x) VSelect(V0(x),1);
        VT_2 =@(x) VSelect(V0(x),2);

        Uexact = S*[VT_1(xc-Lambda(1,1)*time);
        VT_2(xc-Lambda(2,2)*time)];
```