Mathematical Modelling in the Natural Sciences SS20
Solutions to Exercises on Sheet 8

Exercises und Lecture Notes

Contents

It is recommended to use an explicit upwind differencing approach for numerical solutions to these problems, which can be described as follows. To solve a scalar conservation law $u_t + f(u)_x = 0$ on a space-time grid $\{(x_i, t^n) : x_i = i\Delta x, t^n = n\Delta t, i \in \mathbb{Z}, n \in \mathbb{N}_0\}$ the explicit upwind scheme is given by

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + \frac{F_{i+\frac{1}{2}}^n - F_{i-\frac{1}{2}}^n}{\Delta x} = 0$$

where $U_i^k \approx u(x_i, t^n)$ are cell-centered approximations to the solution and the *numerical flux function* F is an approximation to the flux on cell interfaces given by

$$F_{i+\frac{1}{2}}^n = \frac{f(U_i) + f(U_{i+1})}{2} - |a_{i+\frac{1}{2}}| \frac{U_{i+1} - U_i}{2}, \quad a_{i+\frac{1}{2}} = \frac{f(U_{i+1}) - f(U_i)}{U_{i+1} - U_i}.$$

Note for $f(u) = au$,

$$F_{i+\frac{1}{2}} = \frac{a}{2}[U_i + U_{i+1}] - \frac{|a|}{2}[U_{i+1} - U_i]$$

that if $a > 0$ holds, then backward differencing is used for the term $f(u)_x$ (since the *wind* or *information* is flowing from left to right),

$$F_{i+\frac{1}{2}} = aU_i, \quad \frac{F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}}{\Delta x} = a\frac{U_i - U_{i-1}}{\Delta x} = a\frac{U_{i+1} - U_{i-1}}{2\Delta x} - \frac{a\Delta x}{2}\frac{U_{i+1} - 2U_i + U_{i-1}}{\Delta x^2}$$

and if $a < 0$ holds, then forward differencing is used (since the *wind* or *information* is flowing from right to left),

$$F_{i+\frac{1}{2}} = aU_{i+1}, \quad \frac{F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}}{\Delta x} = a\frac{U_{i+1} - U_i}{\Delta x} = a\frac{U_{i+1} - U_{i-1}}{2\Delta x} + \frac{a\Delta x}{2}\frac{U_{i+1} - 2U_i + U_{i-1}}{\Delta x^2}.$$

In both cases, the viscous counterpart $u_t + f(u)_x = \nu u_{xx}$ of the scalar conservation law can be seen with numerical artificial viscosity $\nu = |a|\Delta x/2$. For explicit programming details, see this code for an implementation of the approach for Burger's equation.

• Exercise 1: Traffic Flow, Shock Wave

◦ Task

Let traffic velocity be related to density according to $u(\rho) = u_{\max}(1 - \rho_{\max})$ for $u_{\max}, \rho_{\max} > 0$. The traffic density is then given by $f(\rho) = \rho u(\rho)$. Write a Matlab code to perform the following calculations to model the reaction of traffic to a narrowing road.

- Solve the Riemann Problem with $\rho_{\mathrm{l}} = \rho_{\max}/4$ and $\rho_{\mathrm{r}} = 3\rho_{\max}/4$.
- Sketch the trajectories and the characteristics.
- Sketch $\rho(x,t)$ and $u(x,t)$ for a fixed times $t > 0$.
- Determine the vehicle velocity $v(t)$ along fixed trajectories.

◦ Solution

These calculations are carried out with the following Matlab code.

```
% max velocity and max density
    umax = 1;
    rmax = 1;

% grid parameters
    Nx   = 51;
    xmin = 0;
    xmax = 2;
    xmid = (xmin + xmax)/2;
    x    = linspace(xmin,xmax,Nx);
    hx   = x(2)-x(1);

% iteration parameters
    itmax = 1000;
    tol = 1.0e-6;
```

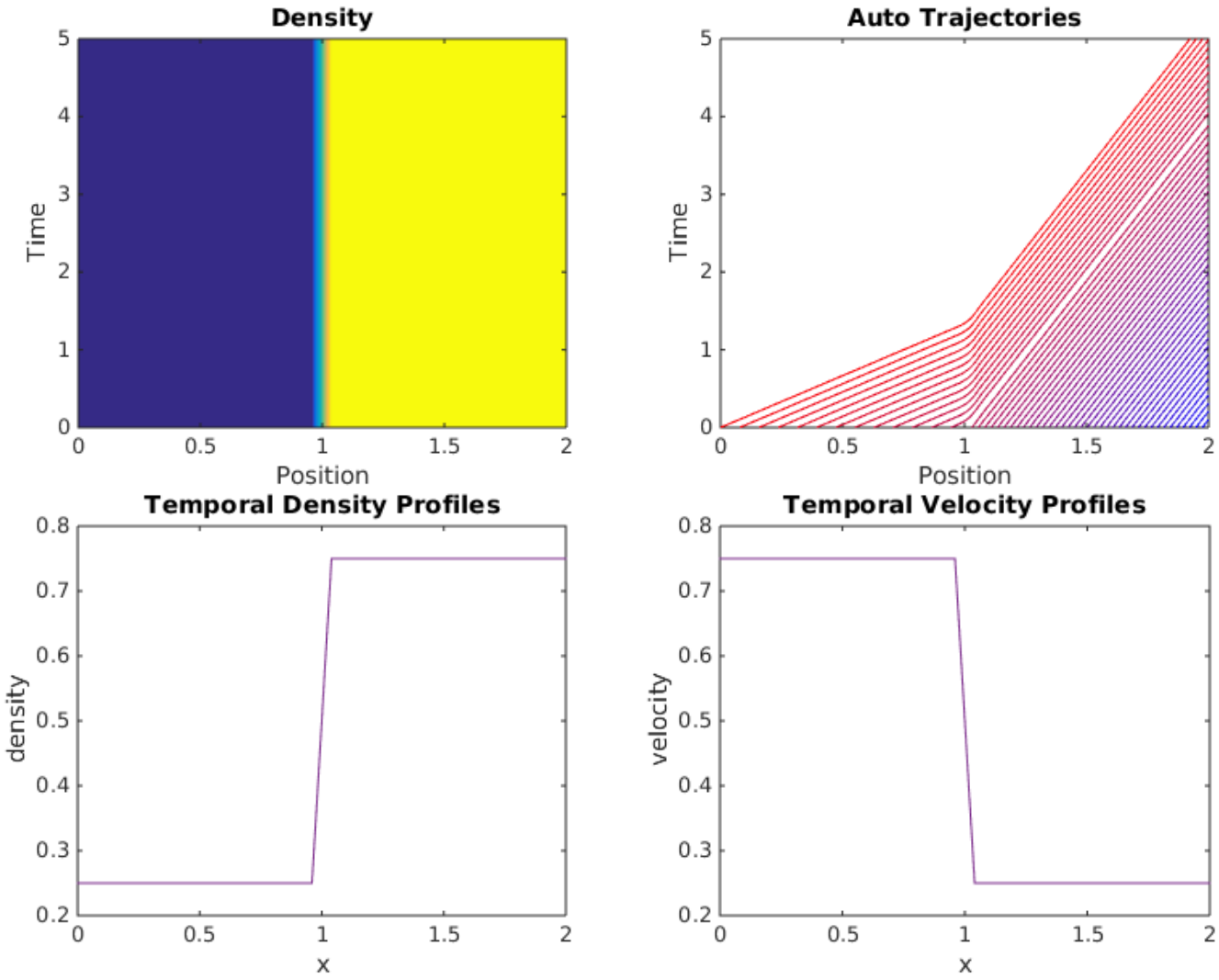
```
% stable time stepping
    dt = hx/10;
    Nt = round(5/dt);
    T  = Nt*dt;

% initial conditions
    rl  = 0.25; rr = 0.75; rm = (rl + rr)/2;   % shock
    rho = rl*(x < xmid) + rm*(x == xmid) + rr*(x > xmid);

% initial auto positions, adjusted to initial density
    X    = zeros(1,Nx);
    X(1) = xmin;
    s    = 1/sum(1./rho);
    for it=1:itmax
        for i=2:Nx
            X(i) = X(i-1) + s/interp1(x,rho,X(i-1),'linear','extrap');
            X(i) = max(X(i),X(i-1));
        end
        if (X(Nx) > xmax)
            sb = s;
            break;
        end
        s = 2*s;
    end
    if (it == itmax)
        error('placement of autos did not converge')
    end
    sa = 0;
    s  = (sa + sb)/2;
    for it=1:itmax
        for i=2:Nx
            X(i) = X(i-1) + s/interp1(x,rho,X(i-1),'linear','extrap');
            X(i) = max(X(i),X(i-1));
        end
        if (X(Nx) < xmax)
            sa = s;
        else
            sb = s;
        end
        s = (sa + sb)/2;
        if (abs(X(Nx)-xmax)<tol)
            break;
        end
    end
    if (it == itmax)
        error('placement of autos did not converge')
    end
```

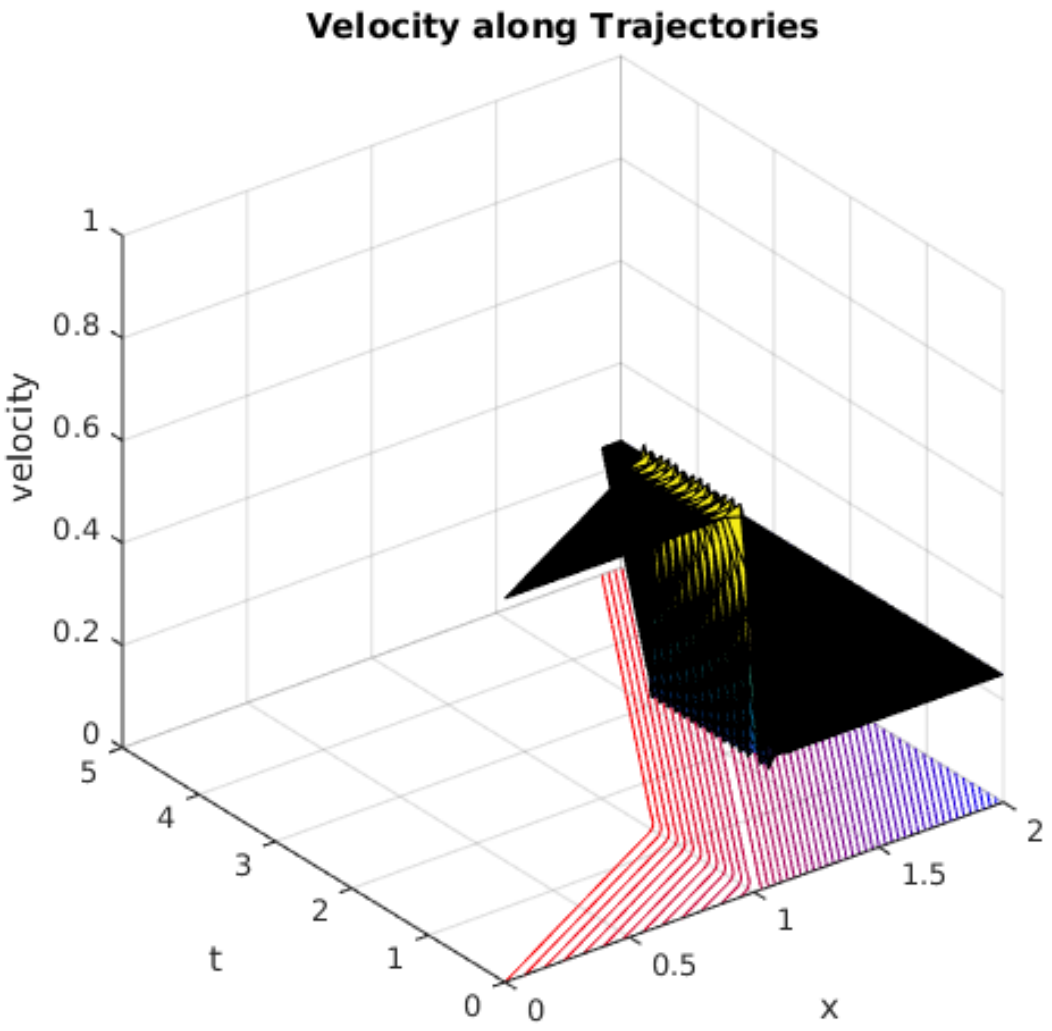
```
% initial plot vectors
    tv   = 0;
    rhov = rho(:);
    Xv   = X(:);

% setup figure for density and trajectories
    h1 = figure(1); close(h1); h1 = figure(1);
    set(h1,'Position',[10 10 900 300]);

% start time stepping
    for k=1:Nt
        rl = rho(1:(Nx-1));                    % density left cell
        fl = umax*rl.*(1-rl/rmax);             % flux left cell
        rr = rho(2:Nx);                        % density right cell
        fr = umax*rr.*(1-rr/rmax);             % flux right cell
        df = fr-fl;                            % flux difference
        dr = rr-rl;                            % density difference
        a  = df.*(dr ~= 0)./(dr + (dr == 0));  % characteristic speed
        F  = (fl + fr)/2 - abs(a).*dr/2;       % upwinded flux function
        dF = [0,F(2:(Nx-1))-F(1:(Nx-2)),0];    % finite differences of flux

        rho = rho - dt*dF/hx;                  % update density

        rho(1) = rho(1) ...
            - (dt/hx)*(a(1) < 0)*df(1);       % characteristic BC left
        rho(Nx) = rho(Nx) ...
            - (dt/hx)*(a(Nx-1) > 0)*df(Nx-1);  % characteristic BC right

        u = umax*(1 - rho/rmax);               % velocity from density
        dX = interp1(x,u,X,'linear','extrap'); % velocity for autos
        X = X + dt*dX;                         % update auto positions

        tv   = [tv,k*dt];                      % update plot vectors
        rhov = [rhov,rho(:)];
        Xv   = [Xv,X(:)];

        if (k == Nt)                           % plot results only at the end
            subplot(1,2,1)
            contour(kron(ones(k+1,1),x), ...
                kron(tv',ones(1,Nx)), ...
                rhov',linspace(0,1,100),'Fill','on');
            title('Density');
            ylabel('Time');
            xlabel('Position');
            axis([min(x) max(x) 0 T]);
            drawnow;
```

```
            subplot(1,2,2)
            i = 1;
                plot(Xv(i,:),tv,'Color',[1-i/Nx,0,i/Nx]); hold on;
            for i=2:Nx
                plot(Xv(i,:),tv,'Color',[1-i/Nx,0,i/Nx]);
            end
            hold off;
            title('Auto Trajectories')
            ylabel('Time')
            xlabel('Position');
            axis([xmin xmax 0 T]);
            drawnow;

        end
    end

% setup figure for density and velocity
    h2 = figure(2); close(h2); h2 = figure(2);
    set(h2,'Position',[10 10 900 300]);

    subplot(1,2,1)
    dN = round(Nt/10);
    rv = rhov(:,1:dN:Nt+1);
    plot(x,rv)
    xlabel('x')
    ylabel('density')
    title('Temporal Density Profiles')

    subplot(1,2,2)
    uv = umax*(1-rv/rmax);
    plot(x,uv)
    xlabel('x')
    ylabel('velocity')
    title('Temporal Velocity Profiles')

% setup figure for velocity along trajectories
    h3 = figure(3); close(h3); h3 = figure(3);
    set(h3,'Position',[10 10 500 500]);

    vv = [Xv(:,2) - Xv(:,1), ...
         (Xv(:,3:Nt+1) - Xv(:,1:Nt-1))/2, ...
          Xv(:,Nt+1) - Xv(:,Nt)];
    vv = vv/dt;
    surf(Xv(:,1:dN:Nt+1)',tv(1:dN:Nt+1),vv(:,1:dN:Nt+1)');
    axis([xmin xmax 0 T 0 umax])
    xlabel('x')
    ylabel('t')
    zlabel('velocity')
```

```
title('Velocity along Trajectories')
hold on;
i = 1;
    plot(Xv(i,:),tv,'Color',[1-i/Nx,0,i/Nx]);
for i=2:Nx
    plot(Xv(i,:),tv,'Color',[1-i/Nx,0,i/Nx]);
end
hold off;
```

The results are shown grafically as follows.





- **Exercise 2: Traffic Flow, Expansion Wave**

  - **Task**

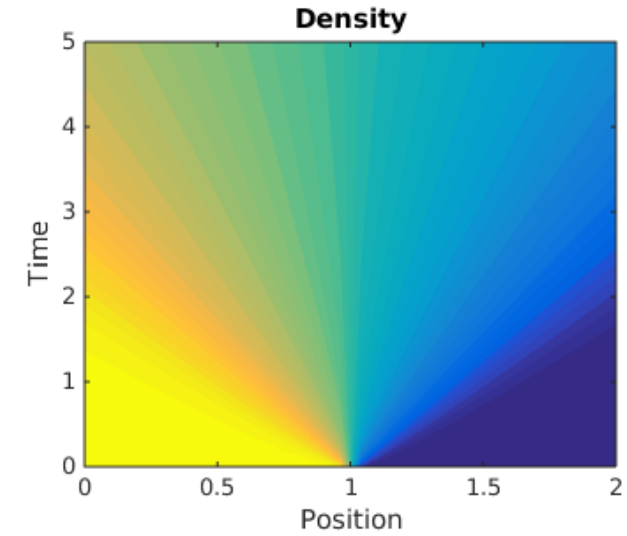
Repeat the last exercise with $\rho_{\mathrm{l}} = 3\rho_{\max}/4$ and $\rho_{\mathrm{r}} = \rho_{\max}/4$ to model the reaction of traffic to a widening road. How would the velocity model have to be changed so that autos further back from the widening would not have to drive initially more slowly than those autos nearer to the widening?

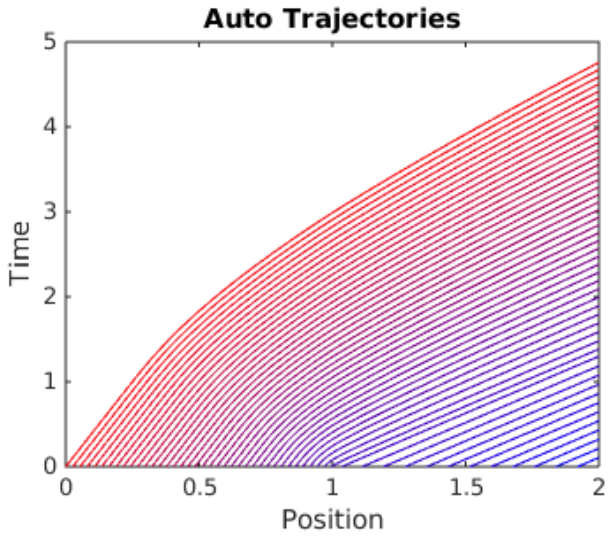
  - **Solution**

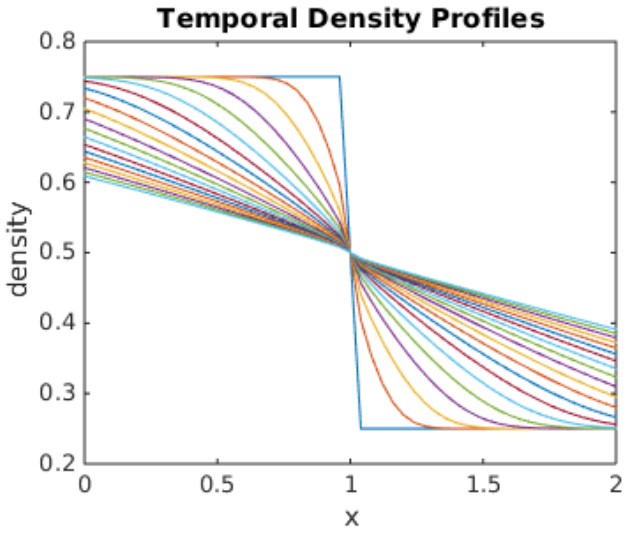
hese calculations are carried out with the Matlab code above but with the following lines changed.

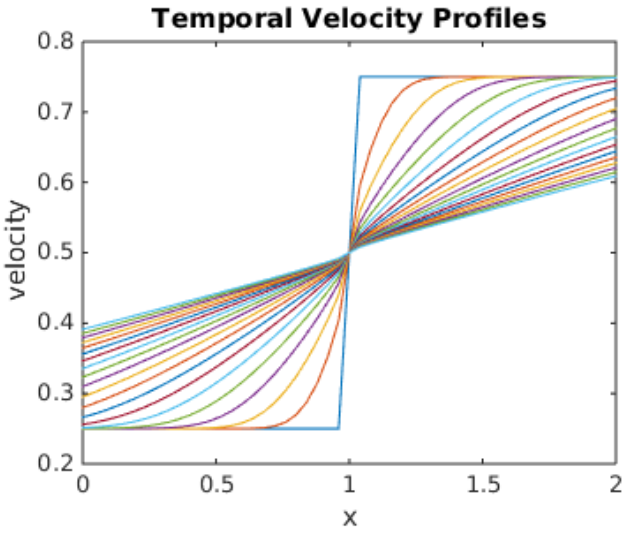
```
% initial conditions
    rl  = 0.25; rr = 0.75; rm = (rl + rr)/2;   % shock
    rho = rl*(x < xmid) + rm*(x == xmid) + rr*(x > xmid);
```

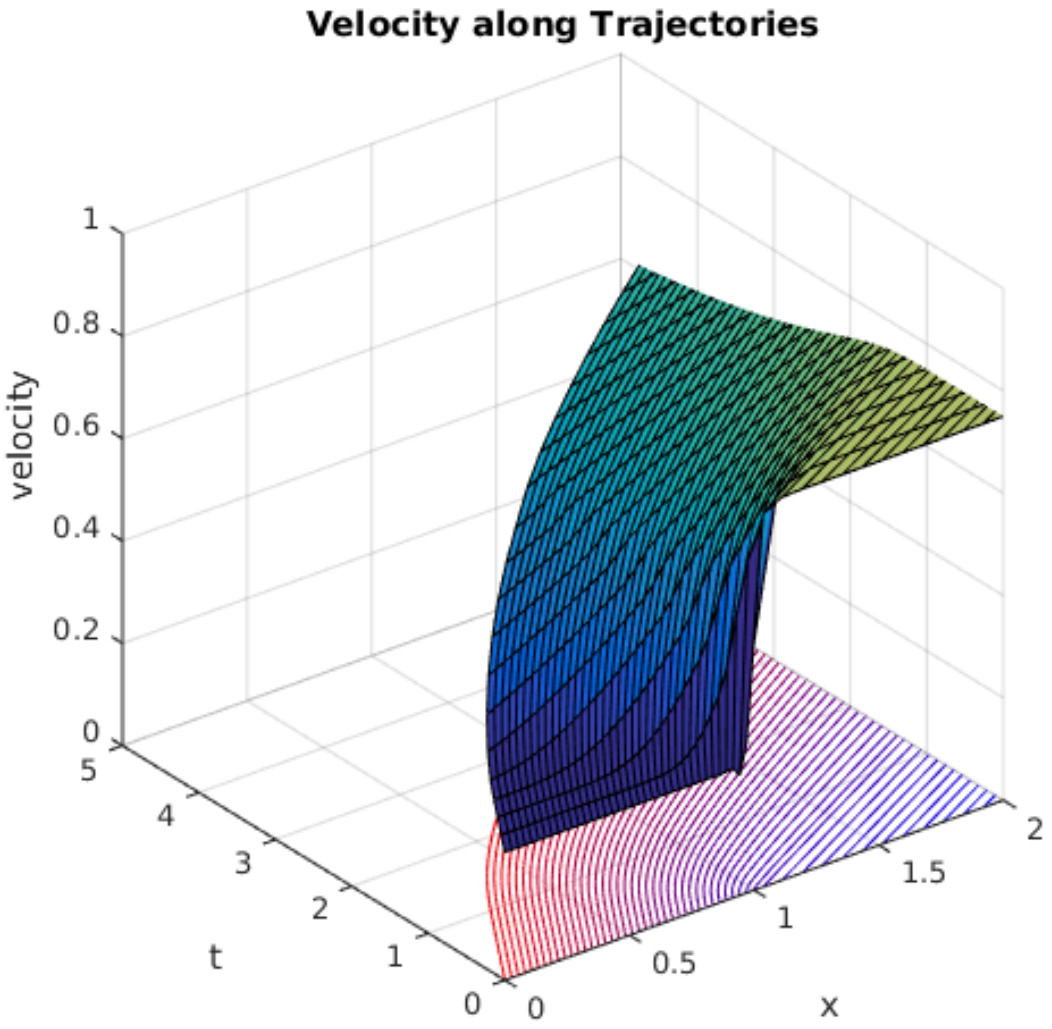
The results are shown grafically as follows.











If the velocity model is given by

$$u(\rho) = u_{\max} \begin{cases} u_{\max}, & 0 \le \rho < \rho_{\max} \\ 0, & \text{otherwise} \end{cases}$$

then with $f(\rho) = \rho u(\rho)$ and $f'(\rho) = u_{\max}$ for $0 \le \rho < \rho_{\max}$, the solution to the Riemann problem

$$\rho_t + u_{\max}\rho_x = 0, \quad \rho(x,0) = \rho_0(x) = \begin{cases} \rho_{\mathrm{l}}, & x < 0 \\ (\rho_{\mathrm{l}} + \rho_{\mathrm{r}})/2, & x = 0 \\ \rho_{\mathrm{r}}, & x > 0 \end{cases}$$

is given by $\rho(x,t) = \rho_0(x - u_{\max}t)$.

• Exercise 3: Traffic Flow, Perturbations

◦ Task

Let traffic velocity be related to density according to $u(\rho) = u_{\max}(1 - \rho_{\max})$ for $u_{\max}, \rho_{\max} > 0$. The traffic flux is then given by $f(\rho) = \rho u(\rho)$. Write a Matlab code implementing an explicit upwinding approach to solve the initial value problem,

$$\rho_t + f(\rho)_x = 0, \quad \rho(x, 0) = \hat{\rho} + \sigma(x)$$

where σ is a local perturbation chosen so that its resulting wave optionally

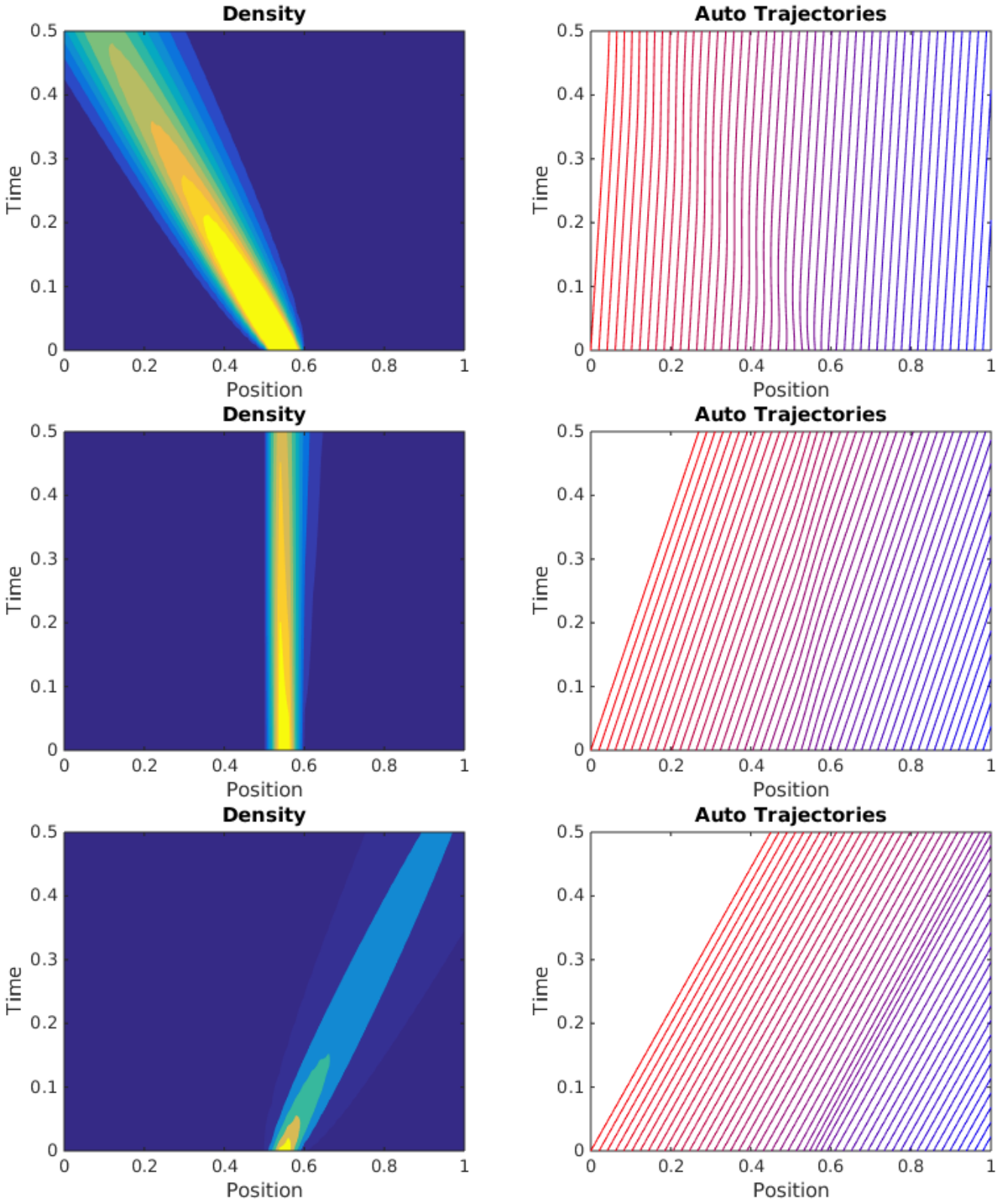
- travels in the opposite direction as traffic,
- stands still or
- travels in the same direction as traffic.

◦ Solution

These calculations are carried out with the Matlab code above but with the following lines changed.

```
% grid parameters
    Nx   = 51;
    xmin = 0;
    xmax = 1;
    xmid = (xmin + xmax)/2;
    x    = linspace(xmin,xmax,Nx);
    hx   = x(2)-x(1);

% stable time stepping
    dt = hx/10;
    Nt = round(0.5/dt);
    T  = Nt*dt;

% initial conditions
   x1  = xmin + 0.5*(xmax - xmin);
   x2  = xmin + 0.6*(xmax - xmin);
% subsonic
   r0 = 0.9;  r1  = (x2-x1)^4/16; r1 = 0.3/r1;
% sonic
%  r0 = 0.46; r1  = (x2-x1)^4/16; r1 = 0.1/r1;
% supersonic
%  r0 = 0.1;  r1  = (x2-x1)^4/16; r1 = 0.05/r1;

   rho = r0*ones(1,Nx) ...
       + r1*(x-x1).^2.*(x-x2).^2.*((x >= x1) & (x <= x2));
```

The results are shown grafically as follows.



• Exercise 4: Traffic Flow, Night-Time Driving, Lagrangian Formulation

◦ Task

Let traffic velocity be related to density according to the model of night-time driving,

$$u(\rho) = \begin{cases} U_0 & \rho < \rho_{\rm a} \\ c\rho & \rho_{\rm a} \le \rho \le \rho_{\rm b} \\ U_1(\rho_{\max} - \rho) & \rho > \rho_{\rm b}. \end{cases}$$

Let $U_0 = 1$, $\rho_{\rm a} = \frac{1}{10}$, $\rho_{\rm b} = \frac{3}{10}$, $c = 10$, $U_1 = \frac{30}{7}$ and $\rho_{\max} = 1$. As usual, the traffic flux is given by $f(\rho) = \rho u(\rho)$. Let $\hat{\rho}$ be the initial density of a column of N autos. Taking a Lagrangian perspective let auto trajectories be determined by a solution to the initial-value problem

$$\begin{cases} x_k'(t) = u(\rho_k(t)), \quad k = 1, \ldots, N-1 \\ x_N'(t) = u(0) \end{cases} \qquad \frac{1}{x_{k+1}(0) - x_k(0)} = \hat{\rho}\left(\frac{x_{k+1}(0) + x_k(0)}{2}\right)$$

where $\rho_k(t)$ is chosen on the one hand to be a density function $\rho_k^{(\rm n)}$ designed to obtain a realistic solution to the night-time driving problem,

$$\rho_k^{(\rm n)}(t) = \frac{1}{x_{k+1}(t) - x_k(t)}$$

and on the other hand chosen to be a density function $\rho_k^{(\rm e)}$ be designed to obtain an entropy solution to the night-time driving problem,

$$\rho_k^{(\rm e)}(t) = \frac{1}{2}\left\{\frac{1}{x_{k+1}(t) - x_k(t)} + \frac{1}{x_k(t) - x_{k-1}(t)}\right\}.$$

(a) Compare the results for the two density functions by solving the Riemann problem with each using $\rho_{\rm l} = 1$ and $\rho_{\rm r} = 0$.

(b) Using $\rho^{(\rm n)}$ demonstrate instability and clustering by starting optionally with the constant initial densities $\hat{\rho} = \rho_{\rm a}$, $(\rho_a + \rho_b)/2$ and $\rho_{\rm b}$.

◦ Solution

The Lagrangian approach is implemented in the following Matlab code.

```
function B08Exp4

    global N U0 ra rb U1 c tv Xv rhov T night

% setup figure
    h1 = figure(1); close(h1); h1 = figure(1);
    set(h1,'Position',[10 10 900 300]);

% night-time driving parameters
    U0    = 1;
    rmin  = 0;
```

```
    rmax  = 1;
    ra    = 0.1;
    rb    = 0.3;
    c     = 10;
    U1    = 3/0.7;
    night = true;

% night vs entropy
    rho0 = rmax;
    xmin = 0; xmax = 10;
    X = [(xmin:(1/rho0):xmax),xmax+(1:3)/ra]';
% clustering
%   rho0 = 1/12; % 1/8, 1/6, 1/5, 1/4, 1/3
%   xmin = 0; xmax = 100;
%   X = (xmin:(1/rho0):xmax)';
%
    N = length(X);
    T = 50;

    if (night)
% night-time solution
        rho = [1./(X(2:N) - X(1:(N-1)));0];
    else
% entropy solution
        rho = 1./(X(2:N) - X(1:(N-1)));
        rho = [rho(1);(rho(1:(N-2))+rho(2:(N-1)))/2;rho(N-1)];
    end

% save for plotting
    tv = 0;
    Xv = X(:);
    rhov = rho(:);

% call to ode solver
    opts = odeset('OutputFcn',@myodeplot,'RelTol',1.0e-10);
    ode15s(@cars,[0 T],X,opts);

end

function dX = cars(t,X)

    global N U0 ra rb U1 c tv Xv rhov T night

    if (night)
% night-time solution
        rho = [1./(X(2:N) - X(1:(N-1)));0];
    else
% entropy solution
```

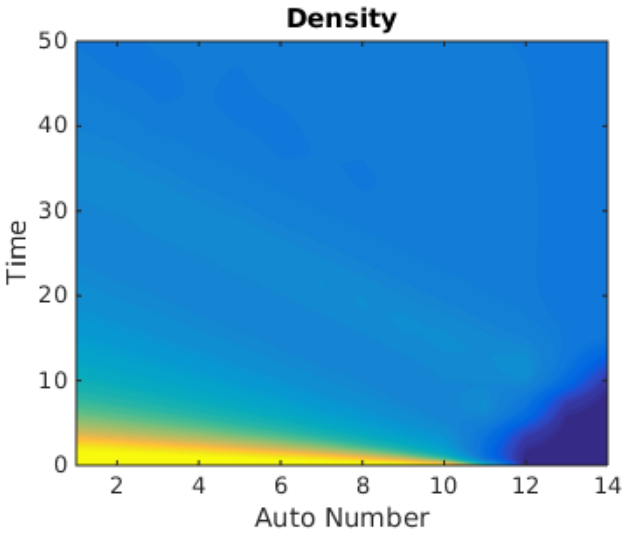
```
        rho = 1./(X(2:N) - X(1:(N-1)));
        rho = [rho(1);(rho(1:(N-2))+rho(2:(N-1)))/2;rho(N-1)];
    end

% velocity
    dX = U0*(rho < ra) ...
       + c*rho.*((ra <= rho) & (rho <= rb)) ...
       + U1*(1 - rho).*(rho > rb);

end

function status = myodeplot(t,X,flag,varargin)

    global N U0 ra rb U1 c tv Xv rhov T night

% no problems
    status = 0;
% at the end, X has zero length...?
    if (size(X,1) ~= N)
        return;
    end

    if (night)
% night-time solution
        rho = [1./(X(2:N) - X(1:(N-1)));0];
    else
% entropy solution
        rho = 1./(X(2:N) - X(1:(N-1)));
        rho = [rho(1);(rho(1:(N-2))+rho(2:(N-1)))/2;rho(N-1)];
    end

% save for plotting
    tv = [tv,t(1)];
    Xv = [Xv,X(:)];
    rhov = [rhov,rho(:)];

% mysteriously necessary correction
    if (length(tv) == 3)
        tv(2)   = (tv(1) + tv(3))/2;
        Xv(:,2) = (Xv(:,1) + Xv(:,3))/2;
    end

%    if (length(tv) > 2)
    if (tv(end) == T)
        subplot(1,2,1)
        contour(1:N,tv,rhov',linspace(0,1,100),'Fill','on')
        title('Density')
        ylabel('Time')
```

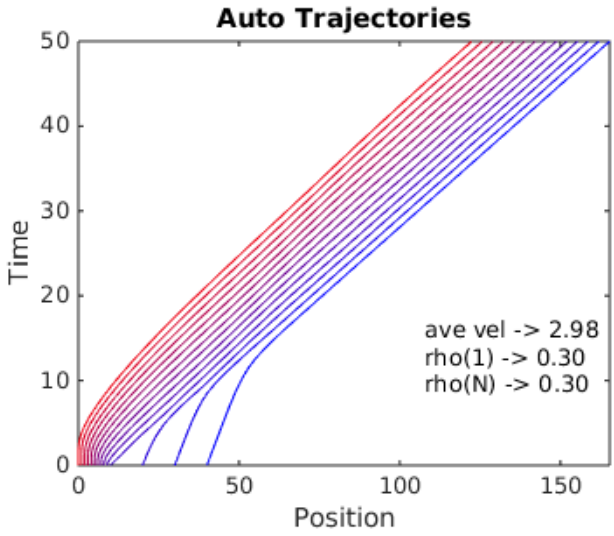
```
        xlabel('Auto Number');
        axis tight;
        drawnow;

        subplot(1,2,2)
        i = 1;
            plot(Xv(i,:),tv,'Color',[1-i/N,0,i/N]);
        hold on;
        for i=2:N
            plot(Xv(i,:),tv,'Color',[1-i/N,0,i/N]);
        end
        hold off;
        title(['Auto Trajectories'])
        ylabel('Time')
        xlabel('Position');
        axis tight;
        ave_vel = (Xv(:,end)-Xv(:,end-1))/(tv(end)-tv(end-1));
        ave_vel = mean(ave_vel);
        dens_l  = 1./(X(2) - X(1));
        dens_r  = 1./(X(N) - X(N-1));
        text(0.65*max(Xv(:)),0.25*max(tv(:)), ...
            sprintf('ave vel -> %0.2f\nrho(1) -> %0.2f\nrho(N) -> %0.2f', ...
            ave_vel,dens_l,dens_r));
        drawnow;
    end
end
```

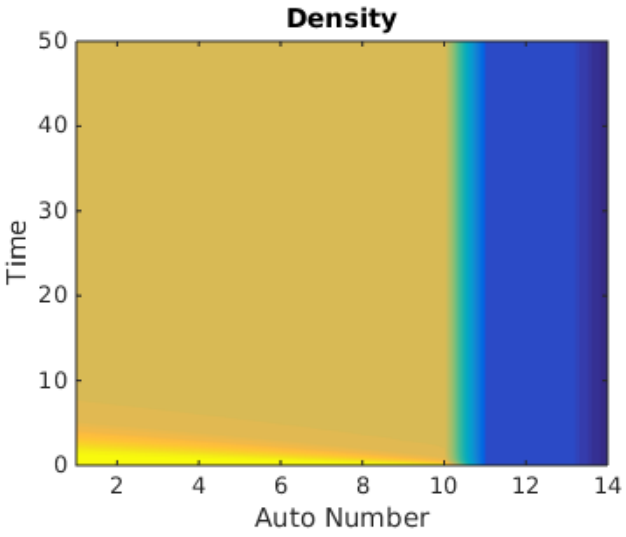
The results are shown grafically as follows.

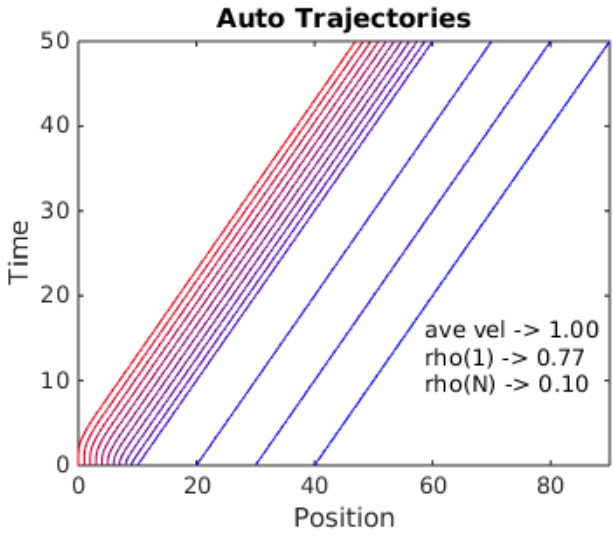
* Part (a) Entropy and Night-Time Solutions

The entropy solution is shown first, followed by the night-time driving solution.









In both cases, an artificial strategy is used to treat the front of the auto column. Otherwise, simply setting the leading density to zero in the simulation is problematic and does not lead to the natural results shown. Instead, some some autos at the front of the auto column are set to such a low density that, according to the velocity model, no neighboring autos are influenced by them. As a result, the interface between these and the autos situated behind them with a lower density exhibits a natural dynamic, as if there were no autos at all at the front of the denser part of the column.

Since there is a sudden difference between the densities at the leading and trailing edges of the auto column, a Riemann problem is solved. The initial state has the two values

$$\rho_{\mathrm{l}} = 1 \quad \text{and} \quad \rho_{\mathrm{r}} \in [0, \rho_{\mathrm{a}}].$$

Since $\rho_{\mathrm{l}} > \rho_{\mathrm{r}}$ holds, an expansion wave emerges initially. After all velocities reach their average, a shock emerges. In the case of the entropy solution, the shock is implicit and corresponds to $\rho = \rho_{\mathrm{b}}$ behind the leading edge of the column and $\rho = 0$ ahead of the leading edge. In the case of the night-time solution, the shock is explicit and corresponds to the traveling interface already present initially between the more and less densely situated autos. In the steady state there is $\rho = \rho_{\mathrm{c}}$ behind the interface and $\rho = \rho_{\mathrm{r}}$ in front of the interface.

∗ Part (b) Clustering and Instability

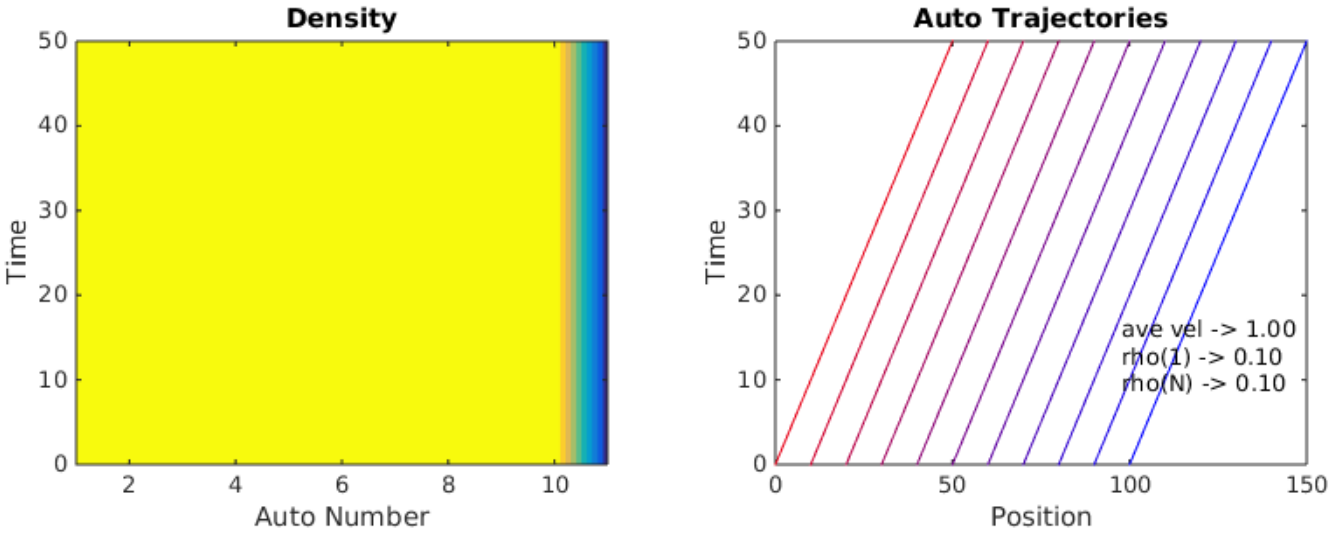
The clustering cases with

$$\hat{\rho} = \tfrac{1}{10}, \tfrac{2}{10} \text{ and } \tfrac{3}{10}$$

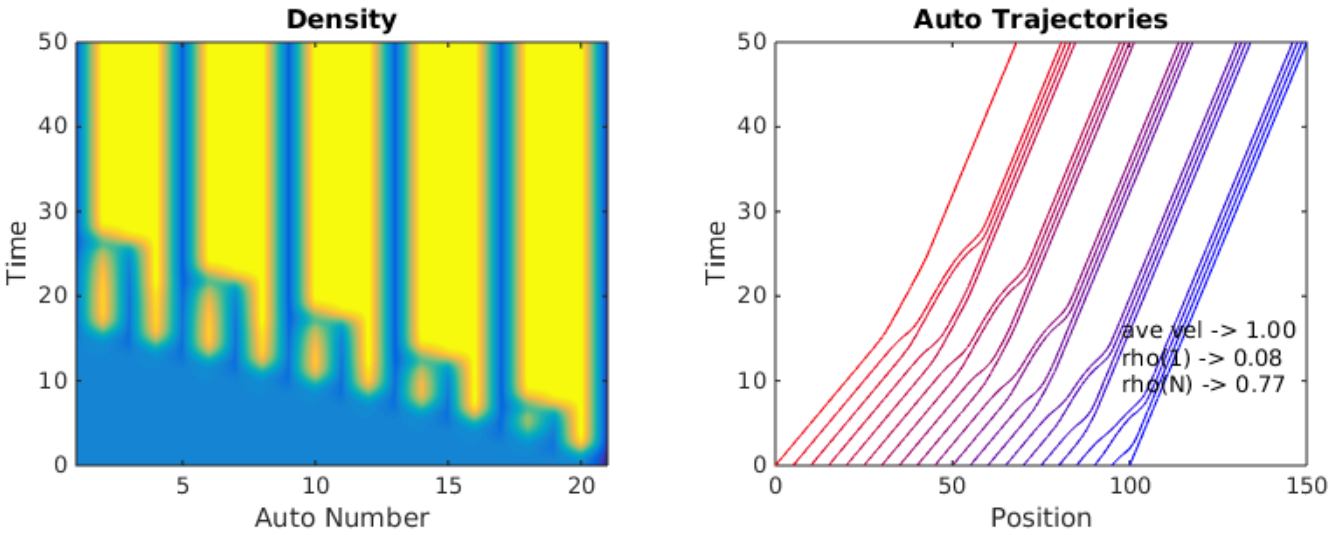
are shown in the following graphs respectively.

In the first case with $\hat{\rho} = \frac{1}{10}$ the initial trailing edge state $\rho_{\mathrm{l}} = \frac{1}{10} = \rho_{\mathrm{a}}$ gives a velocity of $u(\frac{2}{10}) = 2$ for all trailing autos. The leading edge auto acts on the state $\rho_{\mathrm{r}} = 0$ ahead of it and so maintains the speed $u(\rho_{\mathrm{r}}) = u(0) = 1$. Thus, all autos maintain this same constant velocity. The jump from $\rho = \rho_{\mathrm{l}} = \frac{1}{10}$ within the auto column to $\rho = \rho_{\mathrm{r}} = 0$ out in front of the auto column can be seen as a shock whose speed satisfies the Rankine-Hugoniot condition $s = (f(\rho_{\mathrm{r}}) - f(\rho_{\mathrm{l}}))/(\rho_{\mathrm{r}} - \rho_{\mathrm{l}}) = 1$ in the

region $[0, \rho_a]$ of consistent curvature behavior for f.

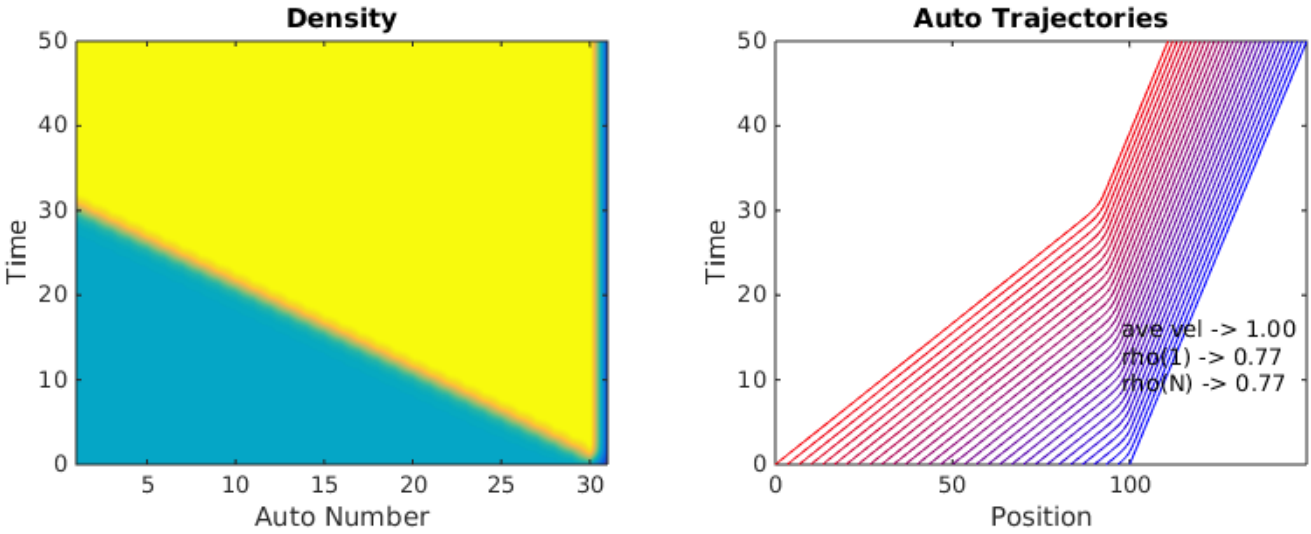


In the next case with $\hat{\rho} = \frac{2}{10}$ the initial trailing edge state $\rho_l = \frac{2}{10} = (\rho_a + \rho_b)/2$ gives an initial velocity of $u(\frac{2}{10}) = 2$ for all trailing autos. Yet the leading edge auto acts on the state $\rho_r = 0$ ahead of it and so maintains the speed $u(\rho_r) = u(0) = 1$. The second auto responds to the increased density ahead of it by accelerating to a maximum speed of $u(\rho_a) = 3$, causing its density to continue increasing finally up to $\rho_c = \frac{23}{30}$ where $u(\rho_c) = 1$. Continuing in this way, transiently changing densities lead to sequences of accelerations and decelerations causing clusters of autos to emerge which finally all travel at the same steady state velocity agreeing with the lead auto $u(0) = 1$. The jumps between the densities ρ_c and 0 at cluster fronts travel with a speed s satisfing the condition in the lecture notes: $s \leq f'(0) = 1$.



In the finaly case with $\hat{\rho} = \frac{3}{10}$ the trailing edge initial state $\rho_l = \frac{3}{10} = \rho_b$ corresponds to a velocity of $u(\frac{3}{10}) = 3$. Yet the leading edge auto acts on the state $\rho_r = 0$ ahead of it and so maintains the speed $u(\rho_r) = u(0) = 1$. The second auto is already traveling at the maximum speed of $u(\frac{3}{10}) = 3$, but the density ahead of it increases finally up to $\rho_c = \frac{23}{30}$ where $u(\rho_c) = 1$. In the same way, all succeeding

autos start from an already maximum speed of $u(\rho_b) = 3$ and reach a maximum density of ρ_c where $u(\rho_c) = 1$. The shock speed satisfies the Rankine-Hugoniot condition $s = (f(\rho_c) - f(\rho_b))/(\rho_c - \rho_b) = -2/7$ in the region $[\rho_b, 1]$ of consistent curvature behavior for f.



- **Exercise 5: Traffic Flow, Night-Time Driving, Eulerian Formulation**

  - **Task**

Repeat the last exercise by taking an Eulerian perspective to compute densities according to an upwind differencing approach by solving the scalar conservation law $\rho_t + f(\rho)_x = 0$ with $f(\rho) = \rho u(\rho)$, where $u(\rho)$ is given by the night-time velocity model. Note that with the upwind scheme the entropy solution will be computed. In order to obtain the more realistic solution seek a numerical flux function which is stable but has more of a focus on the upstream state.

  - **Solution**

The strategy first considered involves a so-called non-local flux approach,

$$f(\rho) = \rho u(\omega \star \rho), \quad \omega(x) = 0, \quad x < 0, \quad \int_0^{+\infty} \omega(x)dx = 1.$$

For the realistic implementation of night-time driving, the numerical flux function was taken simply to be

$$f_i = \rho_i u(\rho_{i+2}), \quad F_{i+\frac{1}{2}} = \frac{f(U_i) + f(U_{i+1})}{2} - |a_{i+\frac{1}{2}}|\frac{U_{i+1} - U_i}{2}, \quad a_{i+\frac{1}{2}} = \frac{f(U_{i+1}) - f(U_i)}{U_{i+1} - U_i}.$$

where there is more focus on the downwind state. States required outside of the domain are created by linear extrapolation. Yet apparently due to the mixed curvature behavior of the flux function the results were rather poor and resembled those of the previous exercise only in a crude way. For the entropy solution $f_i = \rho_i u(\rho_i)$ was used, and results are resemble those of the previous exercise quite well as expected.

Then the following approach led to an Eulerian formulation which consistently reflects the Lagrangian formulation. The development of the successful discretization begins by recalling the Lagrangian semi-discretization

$$x_i'(t) = u(\rho_{i+\frac{1}{2}}(t)), \quad \rho_{i+\frac{1}{2}}(t) = \frac{1}{x_{i+1}(t) - x_i(t)}$$

and noting

$$-\frac{\rho_{i+\frac{1}{2}}'(t)}{\rho_{i+\frac{1}{2}}^2(t)} = \frac{d}{dt}\left(\frac{1}{\rho_{i+\frac{1}{2}}(t)}\right) = x_{i+1}'(t) - x_i'(t) = u(\rho_{i+\frac{3}{2}}(t)) - u(\rho_{i+\frac{1}{2}}(t))$$

or

$$\rho_{i+\frac{1}{2}}'(t) + \rho_{i+\frac{1}{2}}(t)\frac{u(\rho_{i+\frac{3}{2}}(t)) - u(\rho_{i+\frac{1}{2}}(t))}{x_{i+1}(t) - x_i(t)} = 0$$

which suggests the following full discretization of the Eulerian conservation law $\rho_t + (\rho u(\rho))_x = 0$,

$$\frac{\rho_i^{n+1} - \rho_i^n}{\Delta t} + \rho_i^n \frac{u(\rho_{i+1}^n) - u(\rho_i^n)}{\Delta x} = 0.$$

This is the approach used successfully below for the realistic simulation of night-time driving. The development of a discretization for the entropy solution begins by recalling the Lagrangian semi-discretization in this case,

$$x_i'(t) = u(\rho_i(t)), \quad 2\rho_i(t) = \left[\frac{1}{x_{i+1}(t) - x_i(t)} + \frac{1}{x_i(t) - x_{i-1}(t)}\right] = \frac{x_{i+1}(t) - x_{i-1}(t)}{[x_{i+1}(t) - x_i(t)][x_i(t) - x_{i-1}(t)]}$$

and noting

$$\begin{aligned}
-\frac{\rho_i'(t)}{2\rho_i^2(t)} = & \frac{d}{dt}\left(\frac{1}{2\rho_i(t)}\right) \\
= & \frac{[x_{i+1}'(t) - x_i'(t)][x_i(t) - x_{i-1}(t)]}{x_{i+1}(t) - x_{i-1}(t)} \\
+ & \frac{[x_{i+1}(t) - x_i(t)][x_i'(t) - x_{i-1}'(t)]}{x_{i+1}(t) - x_{i-1}(t)} \\
- & \frac{[x_{i+1}(t) - x_i(t)][x_i(t) - x_{i-1}(t)]}{[x_{i+1}(t) - x_{i-1}(t)]^2}[x_{i+1}'(t) - x_{i-1}'(t)] \\
= & \frac{x_i(t) - x_{i-1}(t)}{x_{i+1}(t) - x_{i-1}(t)}[u(\rho_{i+1}(t)) - u(\rho_i(t))] \\
+ & \frac{x_{i+1}(t) - x_i(t)}{x_{i+1}(t) - x_{i-1}(t)}[u(\rho_i(t)) - u(\rho_{i-1}(t))] \\
- & \frac{[x_{i+1}(t) - x_i(t)][x_i(t) - x_{i-1}(t)]}{[x_{i+1}(t) - x_{i-1}(t)]^2}[u(\rho_{i+1}(t)) - u(\rho_{i-1}(t))]
\end{aligned}$$

or

$$
\begin{aligned}
-\rho_i'(t) = &\quad \rho_i(t)\left[\frac{x_{i+1}(t)-x_{i-1}(t)}{[x_{i+1}(t)-x_i(t)][x_i(t)-x_{i-1}(t)]}\right]\frac{x_i(t)-x_{i-1}(t)}{x_{i+1}(t)-x_{i-1}(t)}[u(\rho_{i+1}(t))-u(\rho_i(t))] \\
&+ \quad \rho_i(t)\left[\frac{x_{i+1}(t)-x_{i-1}(t)}{[x_{i+1}(t)-x_i(t)][x_i(t)-x_{i-1}(t)]}\right]\frac{x_{i+1}(t)-x_i(t)}{x_{i+1}(t)-x_{i-1}(t)}[u(\rho_i(t))-u(\rho_{i-1}(t))] \\
&- \quad \rho_i(t)\left[\frac{x_{i+1}(t)-x_{i-1}(t)}{[x_{i+1}(t)-x_i(t)][x_i(t)-x_{i-1}(t)]}\right]\frac{[x_{i+1}(t)-x_i(t)][x_i(t)-x_{i-1}(t)]}{[x_{i+1}(t)-x_{i-1}(t)]^2}[u(\rho_{i+1}(t))-u(\rho_{i-1}(t))] \\
&= \quad \rho_i(t)\frac{u(\rho_{i+1}(t))-u(\rho_i(t))}{x_{i+1}(t)-x_i(t)} \\
&+ \quad \rho_i(t)\frac{u(\rho_i(t))-u(\rho_{i-1}(t))}{x_i(t)-x_{i-1}(t)} \\
&- \quad \rho_i(t)\frac{u(\rho_{i+1}(t))-u(\rho_{i-1}(t))}{x_{i+1}(t)-x_{i-1}(t)}
\end{aligned}
$$

or finally

$$
\rho_i'(t)+\rho_i(t)\frac{u(\rho_{i+1}(t))-u(\rho_i(t))}{x_{i+1}(t)-x_i(t)}+\rho_i(t)\frac{u(\rho_i(t))-u(\rho_{i-1}(t))}{x_i(t)-x_{i-1}(t)}=\rho_i(t)\frac{u(\rho_{i+1}(t))-u(\rho_{i-1}(t))}{x_{i+1}(t)-x_{i-1}(t)}
$$

which suggests the following full discretization of the Eulerian conservation law $\rho_t+(\rho u(\rho))_x=0$

$$
\frac{\rho_i^{n+1}-(\rho_{i+1}^n+\rho_{i-1}^n)/2}{\Delta t}+\rho_i^n\frac{u(\rho_{i+1}^n)-u(\rho_{i-1}^n)}{2\Delta x}=0
$$

a variant of the so-called Lax Friedrichs scheme.

These Eulerian approaches are implemented in the following Matlab code.

```
% example 1: xmin = 0, xmax = 160, xmid ~= 10,  rl = rmax, night = false
% example 2: xmin = 0, xmax =  90, xmid ~= 10,  rl = rmax, night = true
% example 3: xmin = 0, xmax = 150, xmid ~= 100, rl = 1/10, night = true
% example 4: xmin = 0, xmax = 150, xmid ~= 100, rl = 2/10, night = true
% example 5: xmin = 0, xmax = 150, xmid ~= 100, rl = 3/10, night = true

% grid parameters
    Nx   = 21;
    xmin = 0;
    xmax = 150;
    x    = linspace(xmin,xmax,Nx);
    xmid = 100;
    i    = find(abs(x-xmid) == min(abs(x-xmid)),1);
    xmid = x(i);
    hx   = x(2)-x(1);

% night-time driving parameters
```

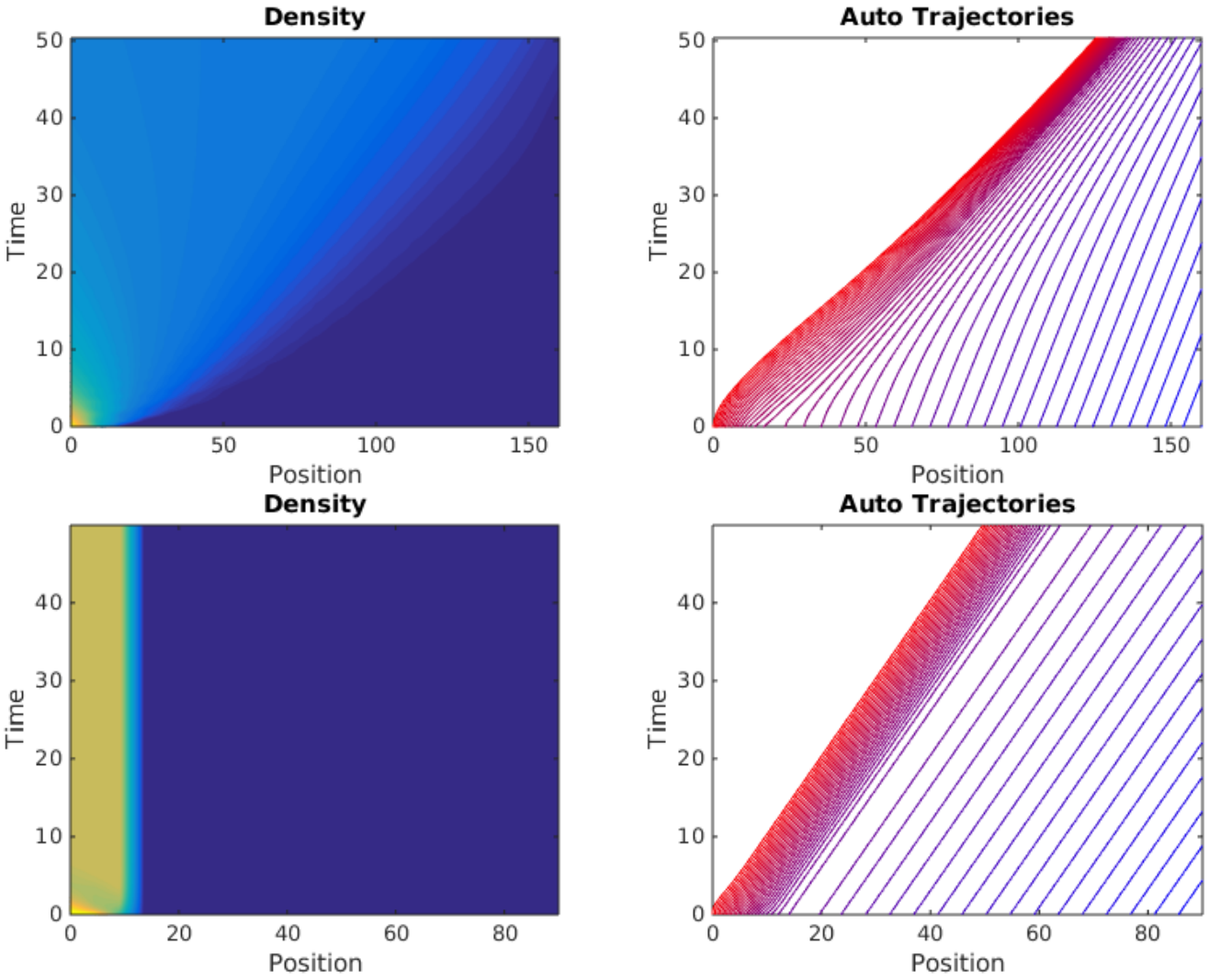
```
    U0    = 1;
    rmin  = 0;
    rmax  = 1;
    ra    = 0.1;
    rb    = 0.3;
    c     = 10;
    U1    = 30/7;
    U     = @(r) U0*((rmin < r) & (r <= ra)) ...
          + c*r.*((ra < r) & (r < rb)) ...
          + U1*(rmax - r).*((rb <= r) & (r < rmax));
    night = true;

% iteration parameters
    itmax = 1000;
    tol = 1.0e-6;

% stable time stepping
    dt = hx/10;
    Nt = round(50/dt);
    T  = Nt*dt;

% initial conditions
    rl  = rmax; rl=0.2;
    rr  = ra;
    rm  = (rl + rr)/2;
    rho = rl*(x < xmid) + rm*(x == xmid) + rr*(x > xmid);

% initial auto positions, adjusted to initial density
    Mx   = 41;
    X    = zeros(1,Mx);
    X(1) = xmin;
    s    = 1/sum(1./rho);
    for it=1:itmax
        for i=2:Mx
            X(i) = X(i-1) + s/interp1(x,rho,X(i-1),'linear','extrap');
            X(i) = max(X(i),X(i-1));
        end
        if (X(Mx) > xmax)
            sb = s;
            break;
        end
        s = 2*s;
    end
    if (it == itmax)
        error('placement of autos did not converge')
    end
    sa = 0;
    s  = (sa + sb)/2;
```

```
    for it=1:itmax
        for i=2:Mx
            X(i) = X(i-1) + s/interp1(x,rho,X(i-1),'linear','extrap');
            X(i) = max(X(i),X(i-1));
        end
        if (X(Mx) < xmax)
            sa = s;
        else
            sb = s;
        end
        s = (sa + sb)/2;
        if (abs(X(Mx)-xmax)<tol)
            break;
        end
    end
    if (it == itmax)
        error('placement of autos did not converge')
    end

% initial plot vectors
    tv   = 0;
    rhov = rho(:);
    Xv   = X(:);

% setup figure for density and trajectories
    h1 = figure(1); close(h1); h1 = figure(1);
    set(h1,'Position',[10 10 900 300]);

% start time stepping
    for k=1:Nt
      if (night)
        re = [rho,2*rho(Nx)-rho(Nx-1)];        % extrapolated density
        rc = re(1:Nx);                         % density cell center
        uc = U(rc);                            % velocity cell center
        rp = re(2:Nx+1);                       % density next cell center
        up = U(rp);                            % velocity next cell center
        dF = rc.*(up - uc);                    % follow-the-leader differencing
        rho = rho - dt*dF/hx;                  % update density
      else
        re = [2*rho(1)-rho(2),rho,2*rho(Nx)-rho(Nx-1)];
        rm = re(1:Nx);                         % density cell center
        um = U(rm);                            % velocity cell center
        rp = re(3:Nx+2);                       % density next cell center
        up = U(rp);                            % velocity next cell center
        dF = rho.*(up - um)/2;                 % central differencing velocity
        rho = (rm+rp)/2 - dt*dF/hx;            % update density
      end
```

```
    rho = max(0,min(rho,rmax));            % ensure bounds

    u  = U(rho);                           % velocity from density
    Y  = max(xmin,min(X,xmax));
    dX = interp1(x,u,Y,'linear','extrap'); % velocity for autos
    X  = X + dt*dX;                        % update auto positions

    tv   = [tv,k*dt];                      % update plot vectors
    rhov = [rhov,rho(:)];
    Xv   = [Xv,X(:)];

    if (k == Nt)                           % plot results only at the end
        subplot(1,2,1)
        contour(kron(ones(k+1,1),x), ...
                kron(tv',ones(1,Nx)), ...
                rhov',linspace(0,1,100),'Fill','on');
        title('Density');
        ylabel('Time');
        xlabel('Position');
        axis([min(x) max(x) 0 T]);
        drawnow;

        subplot(1,2,2)
        i = 1;
            plot(Xv(i,:),tv,'Color',[1-i/Mx,0,i/Mx]); hold on;
        for i=2:Mx
            plot(Xv(i,:),tv,'Color',[1-i/Mx,0,i/Mx]);
        end
        hold off;
        title('Auto Trajectories')
        ylabel('Time')
        xlabel('Position');
        axis([xmin xmax 0 T]);
        drawnow;

    end

end
```

The results are shown grafically as follows.

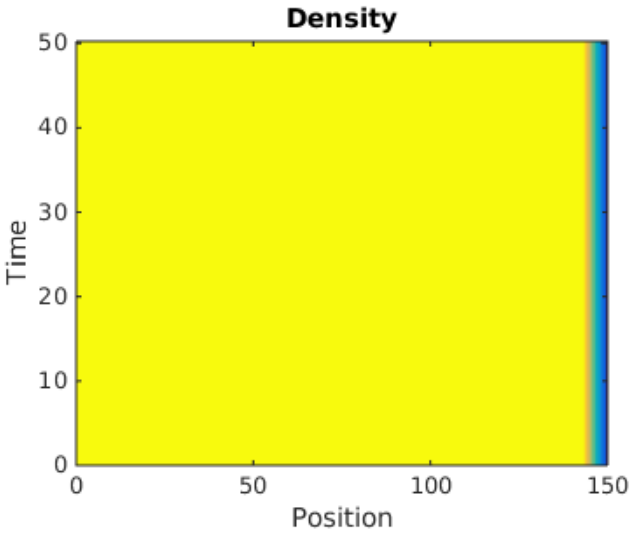
(a) The entropy solution is shown first, followed by the night-time driving solution.

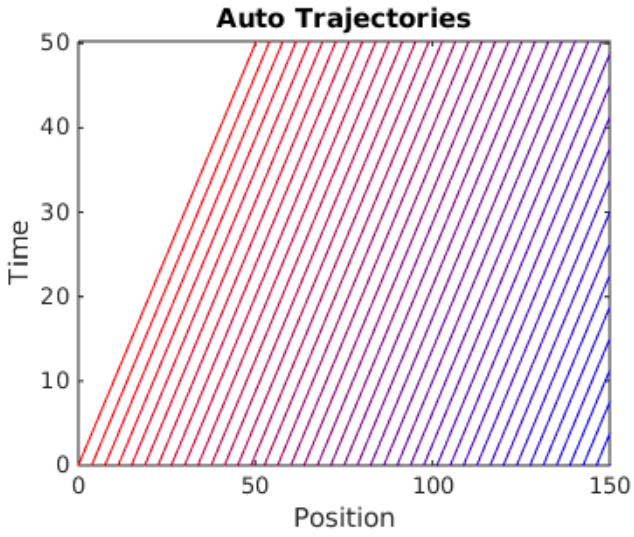


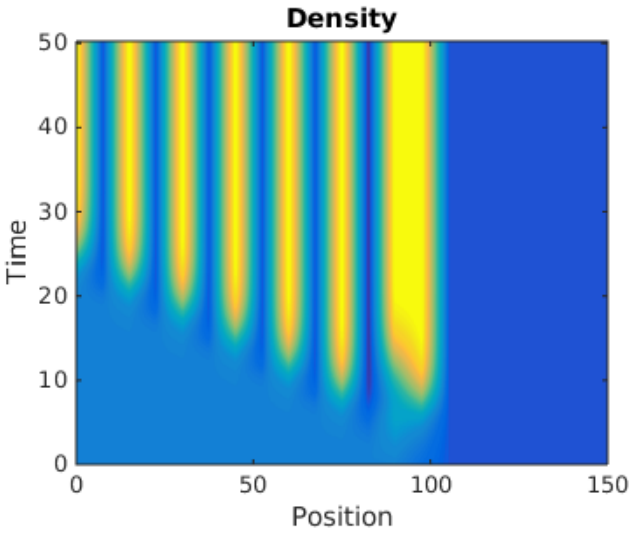
(b) The clustering cases with

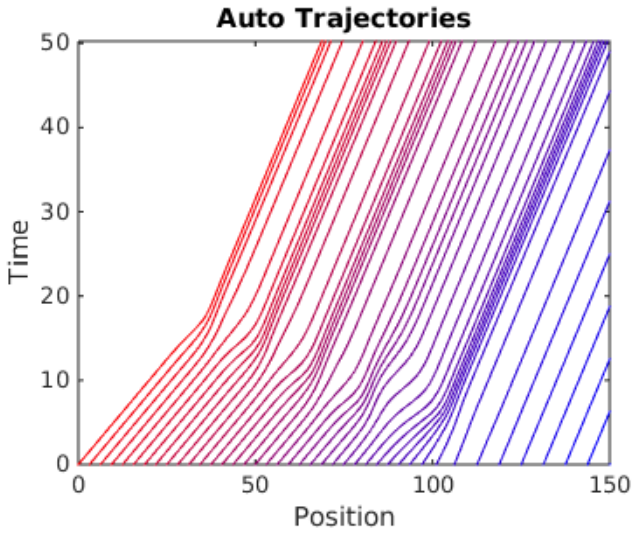
$$\hat{\rho} = \frac{1}{10}, \frac{2}{10} \text{ and } \frac{3}{10}$$

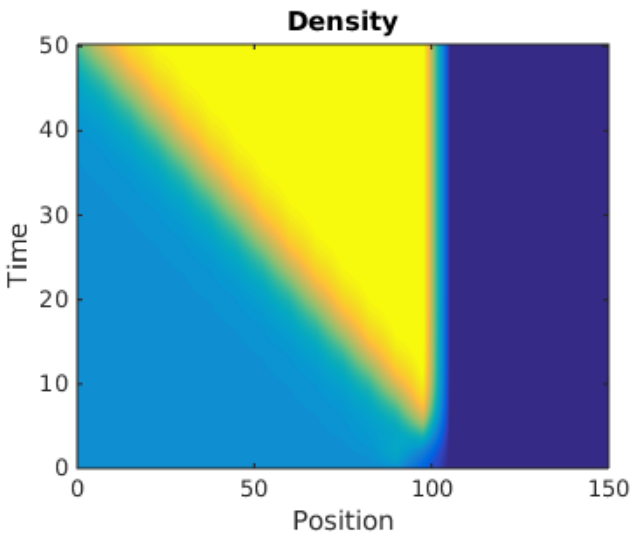
are shown in the following graphs respectively.

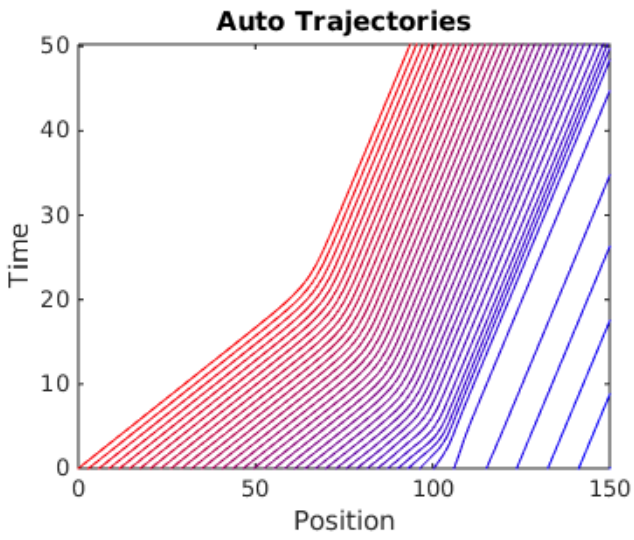












All results generally resemble those of the previous exercise.

• Exercise 6: Traffic Flow, Moving and Waiting Autos

◦ Task

Let traffic velocity be related to density according to $u(\rho) = u_{\max}(1 - \rho_{\max})$ for $u_{\max}, \rho_{\max} > 0$. The traffic flux is then given by $f(\rho) = \rho u(\rho)$. In addition to the moving column of autos there are standing autos waiting for a spot in the column. A waiting auto enters only when the traveling density ρ reaches the threshold $\rho > \rho_{\mathrm{i}}$. Let ω be the density of standing autos. The interaction between the two types of autos is modelled by the initial value problem,

$$\omega_t = -\alpha(\rho)\omega, \quad \rho_t + f(\rho)_x = \alpha(\rho)\omega, \quad \alpha(\rho) = \hat{\alpha}(\rho > \rho_{\mathrm{i}}), \quad \rho(x,0) = \rho_0(x), \quad \omega(x,0) = \omega_0(x)$$

where ω_0 and ρ_0 are the respective initial densities. Write a Matlab code implementing an explicit upwinding approach to solve the initial value problem given initial conditions

$$\omega_0(x) = \begin{cases} \omega_{\mathrm{l}}, & x < 0 \\ \omega_{\mathrm{r}}, & x > 0 \end{cases} \qquad \rho_0(x) = \begin{cases} \rho_{\mathrm{l}}, & x < 0 \\ \rho_{\mathrm{r}}, & x > 0 \end{cases}$$

where, for instance, $\omega_{\mathrm{r}} = 0 < \omega_{\mathrm{l}}$ and $\rho_{\mathrm{l}} < \rho_{\mathrm{i}} < \rho_{\mathrm{r}}$. With $\rho_{\mathrm{l}} < \rho_{\mathrm{r}}$ there emerges a traffic jam. With $\rho_{\mathrm{r}} > \rho_{\mathrm{i}}$ the density of the slowed autos in the region $x > 0$ exceeds the threshold to bring the autos waiting with density ω_{l} into the column.

◦ Solution

These calculations are carried out with the following Matlab code.

```
% setup figure
    h1 = figure(1); close(h1); h1 = figure(1);
    set(h1,'Position',[10 10 1200 300]);

% max velocity and max density
    umax = 1;
```

```
    rmax = 1;

% grid parameters
    Nx   = 100;
    xmin = -5;
    xmax = +5;
    xmid = (xmin + xmax)/2;
    x    = linspace(xmin,xmax,Nx);
    hx   = x(2)-x(1);

% iteration parameters
    itmax = 1000;
    tol = 1.0e-6;

% stable time stepping
    dt = hx/10;
    Nt = round(5/dt);
    T  = Nt*dt;

% initial conditions, shock and jump
    rl   = 0.6;
    rr   = 0.85;
    rm   = (rl + rr)/2;
    rho  = rl*(x < xmid) + rm*(x == xmid) + rr*(x > xmid);
    ome  = 0.2*(x < xmid);
    rhoI = 0.65;
    alf  = @(r) 3*(r > rhoI);

% initial plot vectors
    tv   = 0;
    rhov = rho(:);
    omev = ome(:);

% start time stepping
    for k=1:Nt
        rl = rho(1:(Nx-1));                    % density left cell
        fl = umax*rl.*(1-rl/rmax);             % flux left cell
        rr = rho(2:Nx);                        % density right cell
        fr = umax*rr.*(1-rr/rmax);             % flux right cell
        df = fr-fl;                            % flux difference
        dr = rr-rl;                            % density difference
        a  = df.*(dr ~= 0)./(dr + (dr == 0));  % characteristic speed
        F  = (fl + fr)/2 - abs(a).*dr/2;       % upwinded flux function
        dF = [0,F(2:(Nx-1))-F(1:(Nx-2)),0];    % finite differences of flux

        al  = alf(rho);                        % transition rate, stat to move
        rl  = rho(1);
        rr  = rho(Nx);
```

```
    rho = rho - dt*dF/hx + dt*al.*ome;     % update density of moving
    ome = ome - dt*al.*ome;                % update density of stationary

    rho(1) = rl + dt*al(1)*ome(1) ...
        - (dt/hx)*(a(1) < 0)*df(1);        % characteristic BC left
    rho(Nx) = rr + dt*al(Nx)*ome(Nx) ...
        - (dt/hx)*(a(Nx-1) > 0)*df(Nx-1);  % characteristic BC right

    tv   = [tv,k*dt];                      % update plot vectors
    rhov = [rhov,rho(:)];
    omev = [omev,ome(:)];

    if (k == Nt)                           % plot results only at the end
        subplot(1,3,1)
        contour(kron(ones(k+1,1),x), ...
                kron(tv',ones(1,Nx)), ...
                rhov',linspace(0,1,100),'Fill','on');
        title('Traveling Density');
        ylabel('Time');
        xlabel('Position');
        axis([min(x) max(x) 0 T]);
        drawnow;

        subplot(1,3,2)
        contour(kron(ones(k+1,1),x), ...
                kron(tv',ones(1,Nx)), ...
                omev',linspace(0,1,100),'Fill','on');
        title('Waiting Density');
        ylabel('Time')
        xlabel('Position');
        axis([min(x) max(x) 0 T]);
        drawnow;

        subplot(1,3,3)
        plot(x,rhov(:,1:25:end),'b',x,rhoI*ones(Nx,1),'r:')
        xlabel('x')
        xlabel('Density')
        title('Temporal Density Profiles')
        axis([xmin xmax 0 1.5*rmax])
        drawnow;

    end
end
```

The results are shown grafically as follows. The density of waiting autos is higher $\omega_{\rm l} = 0.2$ to the left and lower $\omega_{\rm r} = 0$ to the right. This shock moves to the left with time as waiting autos enter the column of moving autos. The density of moving autos is initially lower $\rho_{\rm l} = 0.6$ to the left and higher $\rho_{\rm r} = 0.85$ to the right. The higher density to the right leads to a lower velocity, which allows waiting

autos to enter traffic. This shock moves to the left with time, but in the transition zone the density ρ ceases to be simply a step function. As seen from the intermediate contours between those for ρ_{l} and ρ_{r}, and particularly in the temporal profiles, the density ρ_{r} is already over the ignition threshold $\rho_{\mathrm{i}} = 0.65$, and the increase in density $\rho > \rho_{\mathrm{r}}$ behind the shock wave results from waiting autos entering the column.