# Advanced Numerical Methods for PDEs

Homework 2

Francisco Castillo

March 24, 2021

Solve the advection-diffusion equation

$$\partial_t u(x,t) + \partial_x \left[ v u(x,t) - d \partial_x u(x,t) \right] = 0, \quad u(x,t=0) = u^I(x), \quad x \in [0,L],\ t \in [0,T], \quad (1)$$

with periodic boundary conditions $u(x+L,t) = u(x,t), \quad \forall x,t.$

## Problem 1

1. Write a program that solves the problem with the particle iteration

$$\xi(t+\Delta t) = \xi(t) + \Delta t q(\xi), \quad dP[q(\xi) = x] = \phi(x,\xi)dx. \quad (2)$$

Choose $\phi(x,\xi)$ to be uniformly distributed in an interval $[a, a+b]$ with $a, b$ chosen such that the mean and variance of $\phi$ match the transport coefficients in (2). Explain how you compute $q$ from a random variable $\eta$, uniformly distributed in $[0,1]$.

**Solution:** The matlab program is shown after Problem 2. Here, we explain how we choose $\phi(x,\xi)$, $a$, and $b$. We start by noting that a $\phi(x)$ is uniformly distributed,

$$\phi(x,\xi) = \begin{cases} \frac{1}{b} & x \in [a, a+b] \\ 0 & other \end{cases} \quad (3)$$

It's cumulative probability function

$$\begin{aligned} \mathcal{P}(q(\xi) < x) = P(x) &= \int_{-\infty}^{x} \phi(x,\xi)dx, \\ &= \begin{cases} \frac{x-a}{b} & x \in [a, a+b] \\ 0 & other \end{cases} \end{aligned} \quad (4)$$

Let $\eta$ be a random variable uniformly distributed on $[0,1]$. Choose $q(\xi) = g(\eta)$. To find $g$ and the placement of the particles we note that

$$\mathcal{P}(q(\xi) < x) = \mathcal{P}(g(\eta) < x) = \mathcal{P}(\eta < g^{-1}(x)) = g^{-1}(x) = P(x). \tag{5}$$

Hence,

$$g^{-1}(x) = \frac{x-a}{b}, \quad x \in [a,b]. \tag{6}$$

We now can determing $g$,

$$q_j = q(\xi_j) = g(\eta_j) = a + \eta_j(b-a). \tag{7}$$

Now that we know how to compute $q$, we have to determine the size of the interval $[a, a+b]$. To do so, we recall that the convective constant, $v$, is equal to the mean of the distribution and the diffusion coefficient, $d$, is half of its variance scaled by $\Delta t$:

$$v = \mathrm{E}(q(\xi)) = \frac{1}{2}(b+2a), \tag{8}$$

$$d = \frac{1}{2}\sigma'^2(q(\xi)) = \frac{\Delta t}{2}\sigma^2(q(\xi)) = \frac{\Delta t}{2}\frac{1}{12}b^2 = \frac{\Delta t}{24}b^2, \tag{9}$$

where the scaled variance $\sigma'^2(q(\xi)) = \Delta t\sigma^2(q(\xi))$. Finally, we obtain $a$ and $b$,

$$b = \sqrt{\frac{24d}{\Delta t}}, \tag{10}$$

$$a = v - \frac{1}{2}b. \tag{11}$$

Now that we know how to compute $q(\xi)$ from the coefficientes $v$ and $d$ (from the PDE), we can solve the problem via particle methods. See problem 2 for the numerical solution.

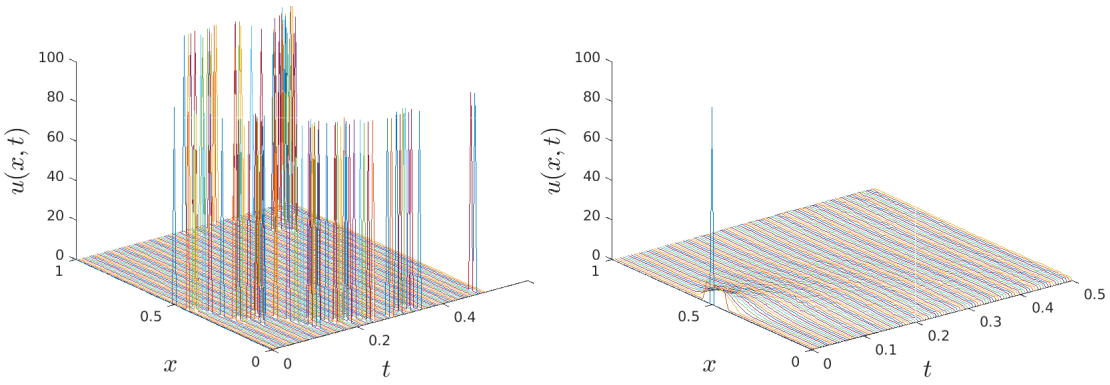
# Problem 2

1. Solve the resulting stochastic process numerically for the parameter values $v = -1$, $d = 0.1$, $L = 1$, $T = 0.5$, $\Delta t = T/100$, and

$$u^I(x) = \delta(x - 0.5) \iff \xi(0) = 0.5.$$

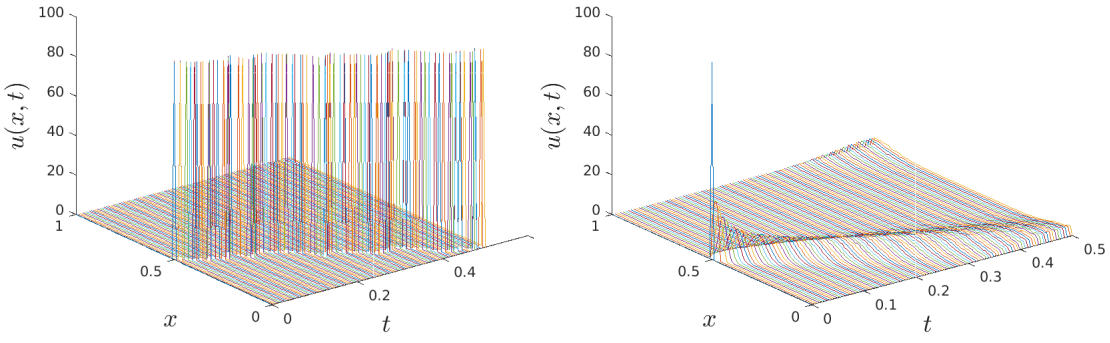
Solve it once with 1 realization and once with the average over $10,000$ realizations. Compute and plot the corresponding density $u(x, t)$.

**Solution:** In the following results, we can see how, by reducing the diffusion coefficient, the particle is more likely to follow a path and the density gets higher around it.



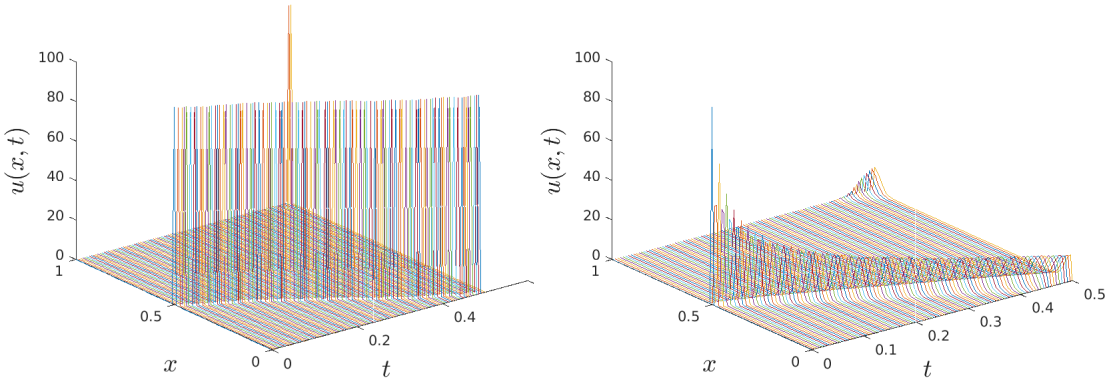

(a) $d = 0.1,\ R = 1.$ (b) $d = 0.1,\ R = 10000.$

Figure 1: Solution $u(x, t)$ against $x$ and $t$ for different number of realizations.




(a) $d = 0.01,\ R = 1.$ (b) $d = 0.01,\ R = 10000.$

Figure 2: Solution $u(x, t)$ against $x$ and $t$ for different number of realizations.




(a) $d = 0.001,\ R = 1.$ (b) $d = 0.001,\ R = 10000.$

Figure 3: Solution $u(x,t)$ against $x$ and $t$ for different number of realizations.

Matlab code:

```
clear all; close all; clc

%% Parameter values
v = -1;
d = 0.001;
L = 1;

T = 0.5;
Ntsteps = 100;
dt = T/Ntsteps;
t = linspace(0,T,Ntsteps+1);

R = 10000; % Number of realizations

M = 100; % Number of grid cells
dx = 1/M;
x = linspace(0,1,(M+1))';

% Distribution size
b = sqrt(24*d/dt);
a = v - 0.5*b;

%% Initial condition
xi_0 = 0.5;
u_0  = calculate_density(x,xi_0);
u(:,1) = u_0;

for r = 1:R

    n = 1; % time-step counter
    xi(1,r) = xi_0;
```

```
    u(:,1,r) = u_0;

    while t(n)<T
        eta = rand(1);
        q = a + eta*b;
        xi_new = xi(n,r) + dt*q;
        % Periodic Boundary conditions
        if (xi_new > L)
            xi(n+1,r) = xi_new - L;
        elseif (xi_new < 0)
            xi(n+1,r) = xi_new + L;
        else
            xi(n+1,r) = xi_new;
        end
        u(:,n+1,r) = calculate_density(x,xi(n+1,r));

        n = n + 1;
    end
end

% Average across realizations
u = mean(u,3);

% Plot solution
linewidth = 2;
labelfontsize = 18;

[tt,xx] = meshgrid(t,x);

figure(2)
plot3(tt,xx,u)
xlabel('$t$','interpreter','latex','fontsize',labelfontsize)
ylabel('$x$','interpreter','latex','fontsize',labelfontsize)
zlabel('$u(x,t)$','interpreter','latex','fontsize',labelfontsize)
figName = create_figName(d,R);
saveas(gcf,figName,'png')

function figName = create_figName(d,R)
    exponent_d = floor(log10(d));
    base_d = d/10^exponent_d;
    path = '../figures/';

    exponent_R = floor(log10(R));
    base_R = R/10^exponent_R;
    figName = append(path,'p2_sol_d',num2str(base_d),'e',num2str(
   exponent_d),'_R',num2str(base_R),'e',num2str(exponent_R));
end

function u = calculate_density(x, xi)
    M = length(x)-1;
    dx = (x(end)-x(1))/M;
    x_staggered = linspace(x(1)-dx,x(end)+dx,M+2);
```

```
    N = 1; % Only one particle moving
%      dx = 1;
    for m=1:M+1
        I=find((x_staggered(m)<xi)&(xi<x_staggered(m+1)));
        u(m)=length(I)/N/dx;
    end
end
```

Consider an ensemble of cars, distributed on a circular course of length $L$ (so $x \in [0, L]$), trying to travel at a desired speed $v_0(x)$. Assume that there is a minimal distance $d$ between cars and a corresponding congestion density $c$. Simulate for a time $t \in [0, T]$. Plot a solution of the congestion backward wave and the distribution of cars at $t = T$.

# Problem 3

1. Consider the Lighthill - Whitham - Richardson (LWR) mean field model for this problem

$$\partial_t u(x,t) + \partial_x [v(u,x)u(x,t)] = 0,$$
$$v(u,x) = v_0(x)\left(1 - \frac{u}{c}\right),$$

with the initial and periodic boundary conditions

$$u(x, t = 0) = u^I(x),$$
$$u(x + L, t) = u(x).$$

Write a code to solve the problem with the Lax-Friedrichs method for $t \in [0, T]$ with the parameters $L = 50(km)$, $c = 100(cars/km)$, $T = 1(h)$,

$$v_0(x) = \begin{cases} 20(km/h) & 30 < x < 40 \\ 100(km/h) & else \end{cases}$$

and

$$u^I(x) = \begin{cases} 50(cars/km) & 0 < x < 10 \\ 0 & else \end{cases}$$

Use $N = 100$ gridpoints ($\Delta x = L/N$) and compute the time step $\Delta t$ adaptively according to the CFL condition.

**Solution:** We start with the Lax-Friedrichs scheme,

$$u_j^{n+1} = \frac{1}{2}\left(u_{j+1}^n + u_{j-1}^n\right) - \frac{\Delta t}{2\Delta x}\left(f_{j+1}^n - f_{j-1}^n\right), \tag{12}$$

where the flux function

$$f_j^n = v_j^n u_j^n, \tag{13}$$

with

$$v_j^n = v_{0j}\left(1 - \frac{u_j^n}{c}\right), \tag{14}$$

Note that we use (16) only for the interior $j = 2 : N$. We need to calculate $\Delta t$. To do so, since $v(u, x)$ is clearly not constant, we use the most restrictive of cases, i.e., the maximum value of $v$.

$$\Delta t = \frac{\Delta x}{\max_j\{v_j\}}, \qquad j = 1, \ldots, N+1 \tag{15}$$

To close the problem, we enforce periodic boundary conditions:

$$u_0 = u_{N+1},$$
$$u_{N+2} = u_1.$$

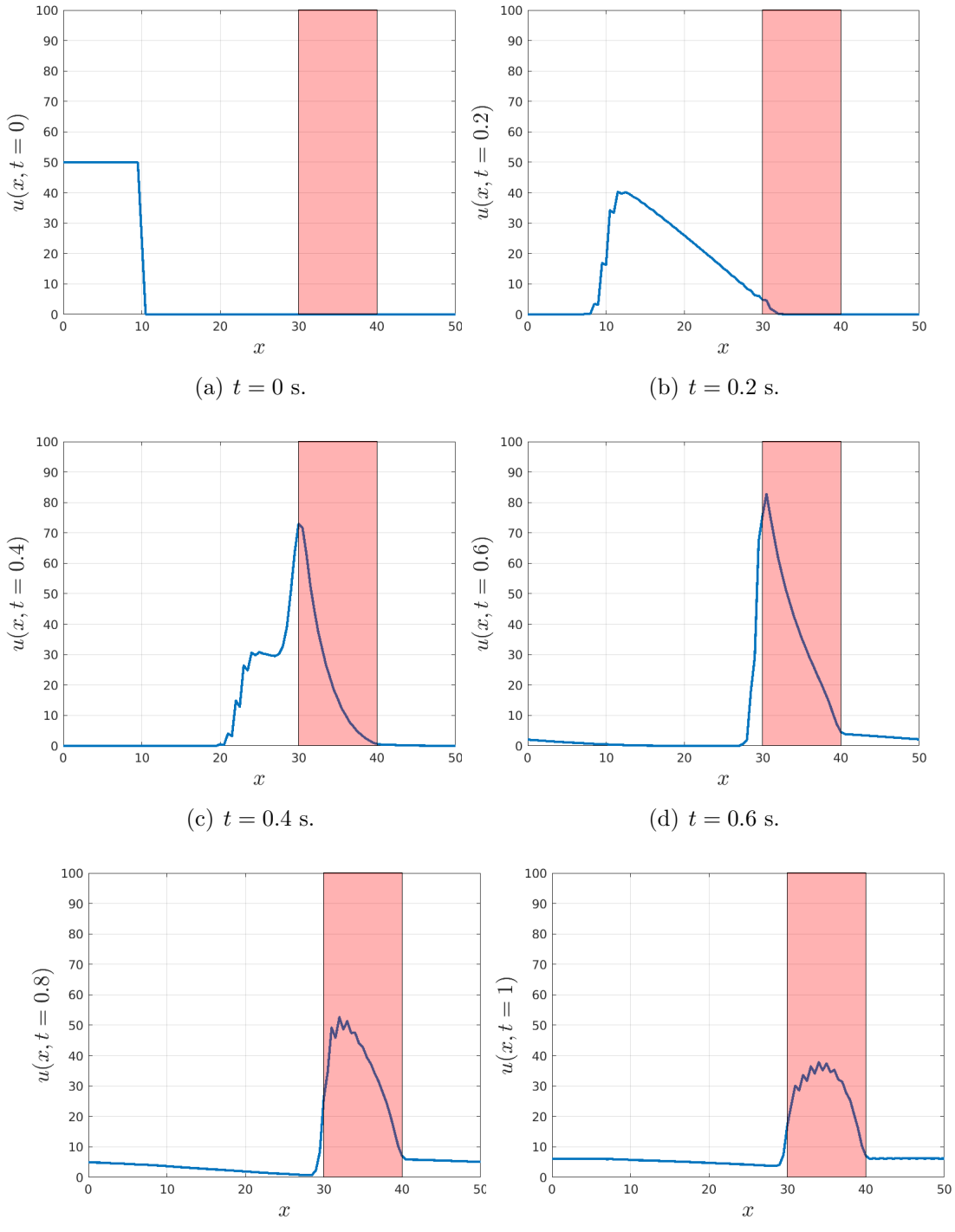
Which, applied to (16), gives us

$$u_1^{n+1} = \frac{1}{2}\left(u_2^n + u_{N+1}^n\right) - \frac{\Delta t}{2\Delta x}\left(f_2^n - f_{N+1}^n\right), \tag{16}$$
$$u_{N+1}^{n+1} = \frac{1}{2}\left(u_1^n + u_N^n\right) - \frac{\Delta t}{2\Delta x}\left(f_1^n - f_N^n\right). \tag{17}$$

To finish, we wish to see the behaviour of the phase velocity $v_P$, defined as

$$v_P(x,t) = \frac{\partial}{\partial u}\left[v(u,x)u(x,t)\right], \tag{18}$$
$$= \frac{\partial}{\partial u}\left[v_0(x)\left(1 - \frac{u(x,t)}{c}\right)u(x,t)\right], \tag{19}$$
$$= v_0(x)\frac{\partial}{\partial u}\left[u(x,t) - \frac{u^2(x,t)}{c}\right], \tag{20}$$
$$= v_0(x)\left(1 - \frac{2}{c}u(x,t)\right). \tag{21}$$

See the matlab code at the end of the problem after the results. In figures 4 and 5 I show the solutions $u(x,t)$ and $v_P(x,t)$, respectively, every 0.2 seconds.

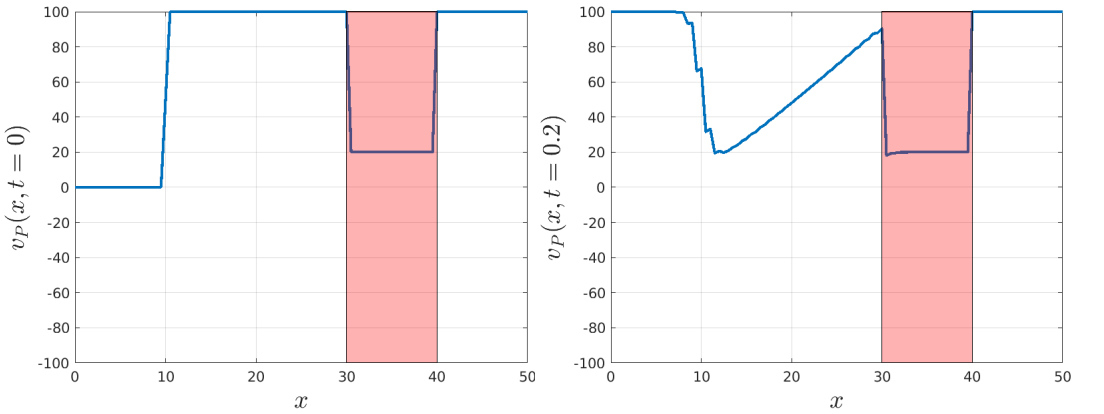



(a) $t = 0$ s. (b) $t = 0.2$ s.

(c) $t = 0.4$ s. (d) $t = 0.6$ s.

(e) $t = 0.8$ s. (f) $t = 1$ s.

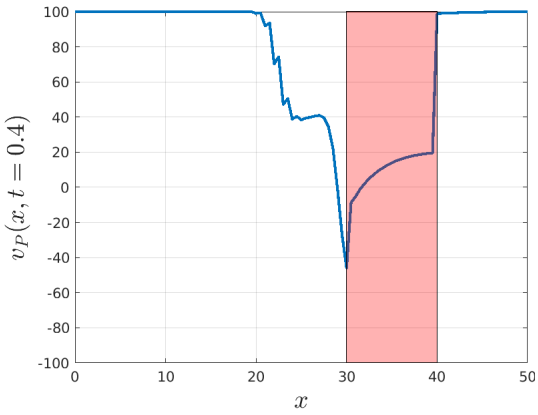
Figure 4: Density $u(x, t)$ against $x$ for different values of time $t$. The red shaded area corresponds to the zone where the cars have to go slower.



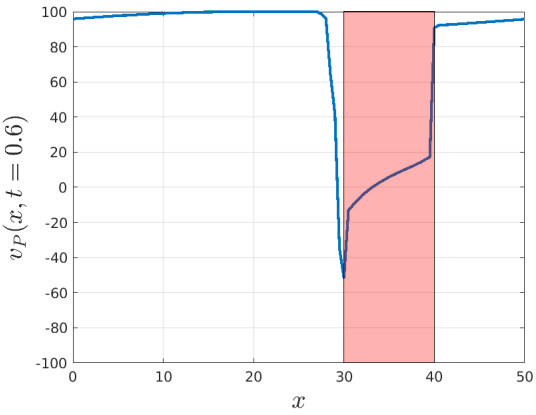

(a) $t = 0$ s.



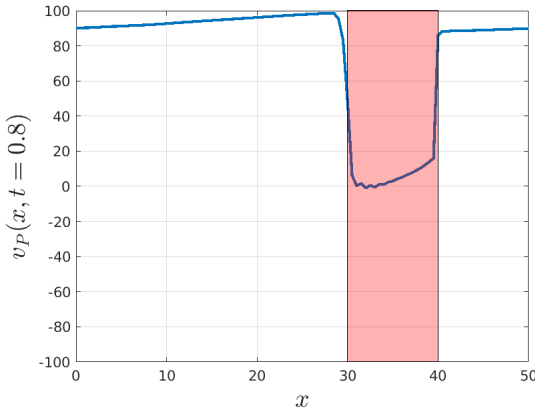

(b) $t = 0.2$ s.



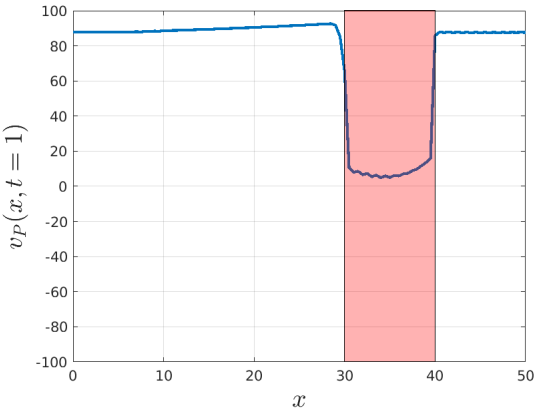

(c) $t = 0.4$ s.




(d) $t = 0.6$ s.




(e) $t = 0.8$ s.




(f) $t = 1$ s.

Figure 5: Phase velocity $v_P(x, t)$ against $x$ for different values of time $t$. The red shaded area corresponds to the zone where the cars have to go slower.

Matlab code:

```
clear all; close all; clc
format long

%% Parameter Values
L = 50;
c = 100;
N = 100; % Mesh size
T = 1;   % Final time
x0 = 0;
xN = L;
dx = (xN-x0)/N;
x = linspace(x0,xN,N+1)';

%% Initial condition and Free velocity
% Initial Condition
step_width = 10;
initial_speed = 50;
u0 = calculate_u0(x,initial_speed,step_width);
% Free velocity v0 (density u=0)
v_slow_interval = [30, 40];
v_slow = 20;
v_fast = 100;
v0 = calculate_v0(x,v_slow_interval,v_slow,v_fast);


%% Prepare for the loop
t = 0;
u = u0;
vp = v0.*(1-2*u/c);
u_new = zeros(size(u)); % Just to allocate for its size

saveTimes = 0.2:0.2:T;
save_idx = 1;

% Plot initial condition
plot_u(x,t,u,c,L,v_slow_interval)
saveas(gcf,'../figures/p3_u_t0','png')
% Plot initial phase velocity
plot_vp(x,t,vp,c,L,v_slow_interval)
saveas(gcf,'../figures/p3_vp_t0','png')

while t<T
    v_g = calculate_group_speed(v0,u,c); % Speed of the particles
    flux = u.*v_g;
    dt = calculate_dt(v_g,dx);

    % Check for plotting times
    if ( saveTimes & t+dt > saveTimes(save_idx))
        dt = saveTimes(save_idx) -t;
        save_idx = save_idx +1;
    elseif ( t+dt > T)
```

```
        dt = T -t;
    end

    C = dt/dx; % Courant Number
    % Advance in time
    for j=2:N % just the interior
        u_new(j) = ( u(j+1) + u(j-1) )/2 - C * ( flux(j+1) - flux(j-1) )/2;
    end

    % Periodict Boundary Conditions
    u_new(1) = ( u(2) + u(N+1) )/2 - C * ( flux(2) - flux(N+1) )/2; % BC on the left
    u_new(N+1) = ( u(1) + u(N) )/2 - C * ( flux(1) - flux(N) )/2; % BC on the right

    % Update u and t
    u = u_new;
    t = t + dt;
    % Update phase velocity
    vp = v0.*(1-2*u/c);


    % Plot solutions and save if appropriate
    plot_u(x,t,u,c,L,v_slow_interval)  % Density u
    plot_vp(x,t,vp,c,L,v_slow_interval) % Phase velocity
    if (ismember(t,saveTimes))
        % Density solution
        figName = create_figName('u',t);
        figure(1)
        saveas(gcf,figName,'png')
        disp(['Saved figure -> "',figName,'.png"'])

        figName = create_figName('vp',t);
        figure(2)
        saveas(gcf,figName,'png')
        disp(['Saved figure -> "',figName,'.png"'])
    end
end

function v = calculate_group_speed(v0,u,c) % Speed of the particles
    v = v0.*(1-u/c);
end

function u0 = calculate_u0(x,initial_speed,step_width) % Initial solution
    u0 = initial_speed * heaviside(step_width - x);
end

function v0 = calculate_v0(x,slow_interval,v_slow,v_fast) % Free velocity (density u=0)
    is_slow = x>slow_interval(1) & x<slow_interval(2);
```

```
    is_fast = ~is_slow;
    v0 = v_slow*is_slow + v_fast*is_fast;
end

function dt = calculate_dt(v,dx)
    a = max(v);
    dt = 0.9*round_down(dx/abs(a),6);
end

function round_number = round_down(number,decimals)
    multiplier = 10^decimals;
    round_number = floor(number * multiplier)/multiplier;
end

function plot_u(x,t,u,c,L,v_slow_interval)
    % Plot parameters
    a = v_slow_interval(1);
    b = v_slow_interval(2);
    linewidth = 2;
    labelfontsize = 18;

    figure(1)
    plot(x,u,'linewidth',linewidth)
    grid on
    axis([0 L 0 c])
    xlabel('$x$','interpreter','latex','fontsize',labelfontsize)
    ylabel(['$u(x,t=',num2str(t),')$'],'interpreter','latex','
   fontsize',labelfontsize)

    % Plot shaded area
    xx = [a a b b];
    yy = [0 c c 0];
    patch(xx,yy,'red')
    alpha(0.3)
end

function plot_vp(x,t,vp,c,L,v_slow_interval)
    % Plot parameters
    a = v_slow_interval(1);
    b = v_slow_interval(2);
    linewidth = 2;
    labelfontsize = 18;

    figure(2)
    plot(x,vp,'linewidth',linewidth)
    grid on
    axis([0 L -c c])
    xlabel('$x$','interpreter','latex','fontsize',labelfontsize)
    ylabel(['$v_P(x,t=',num2str(t),')$'],'interpreter','latex','
   fontsize',labelfontsize)

    % Plot shaded area
```

```
    xx = [a a b b];
    yy = [-c c c -c];
    patch(xx,yy,'red')
    alpha(0.3)
end

function figName = create_figName(field,t)
    exponent = floor(log10(t));
    base = t/10^exponent;
    path = '../figures/';
    figName = append(path,'p3_',field,'_t',num2str(base),'e',num2str
   (exponent));
end
```

# Problem 4

1. Simulate the same scenario by using a particle method, according to the Follow The Leader (FTL) model, given by the rule

$$\xi_j(t + \Delta t) = \xi_j(t) + \Delta t v(\xi_j(t), \xi_{j+1}(t)). \tag{22}$$

The velocity $v$ is determined by

$$v(\xi_j, \xi_{j+1}) = \begin{cases} v_0(\xi_j) & if \quad \xi_j(t) + \Delta t v_0(\xi_j) + d \leq \xi_{j+1}, \\ v(\xi_{j+1}, \xi_{j+2}) & if \quad \xi_j(t) + \Delta t v_0(\xi_j) + d > \xi_{j+1}, \end{cases} \tag{23}$$

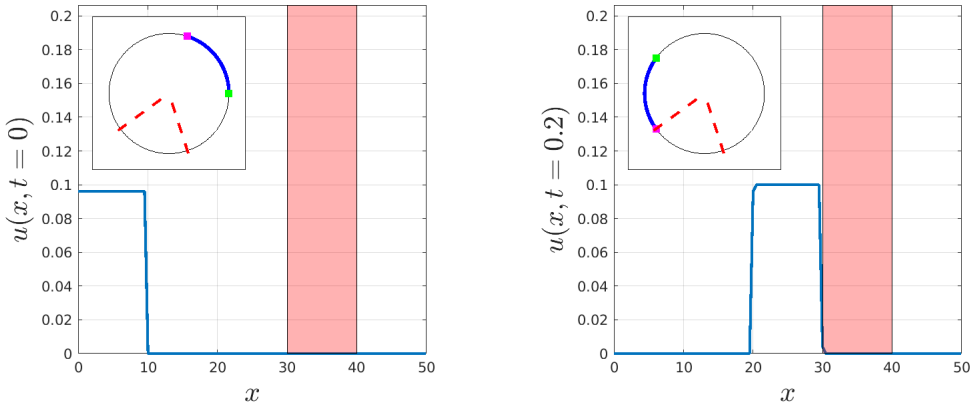
with $j = 1 : Npart$. Solve the problem for $Npart = 500$ cars, $d = 0.01$ km, and an initial condition (corresponding to Problem 3) $\xi_j(0) = 0.02j$, $j = 1 : Npart$.

Repeat the computation by artificially splitting one car into ten (virtual) cars, $Npart = 5000$, occupying a space $d = 0.001$ km and $\xi_j(0) = 0.002j$, $j = 1 : Npart$.

**Solution:** In the following figure we can see the solution every $0.2s$. We can see how the initial density of cars increases rapidly once we hit the "slow zone" (b), represented by the red shaded area. In the track we can see that the cars turn red to represent those ones travelling at a slower speed than they're allowed by the track, to not hit the one in front of them. The green and magenta points represent the first and last car, respectively. Once inside the slow zone, all cars are blue colored, which means that they're not limited by the one in front of them (c). This is because they all travel at the same lower speed imposed by the track. As particles leave the slow zone, the density diminishes even lower than the initial density (d),(e). This is because once the car leaves the slow zone, it travels much faster. In the time that it takes the next car to leave the slow zone, the car ahead has increased their distance, therefore lowering the density. It has now been reached an equilibrium, and the density will stay constant as is, inside and outside the slow zone. This may seem counter intuitive. The reason behind it is that every given period of time the one particle leaves the slow zone and another one enters, at the same rate. Therefore, the density will stay constant, given this conditions.

Since the system is ergodic, dividing the system into ten virtual cars will yield the same qualitative results. We could even do the opposite. We could contract 10 cars into 1, which gives better images. See this new scenarion in figure 7. We can see that we obtain the solution with just less nodes leading to poor resolution.

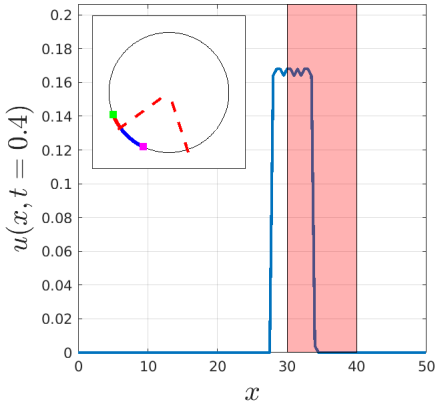
Finally, the peaks in the density are due to the way I have computed it. The moment one car exits a cell and enters the next one, the density may change if the measuring mesh is not properly chosen, which I have not.



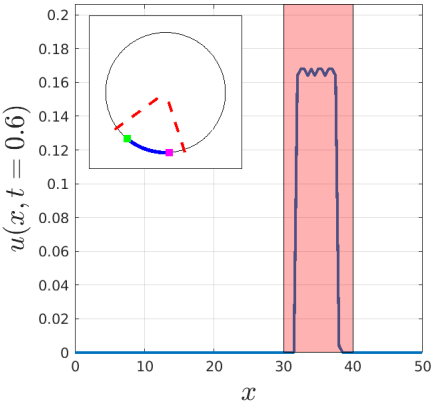

(a) $t = 0$ s.



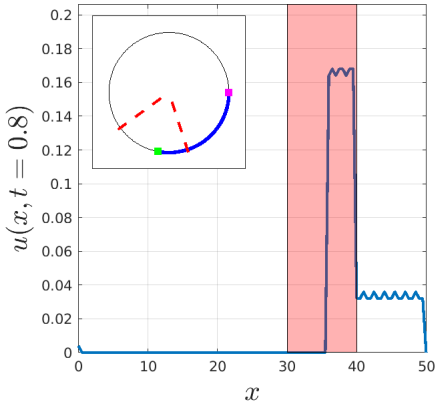

(b) $t = 0.2$ s.



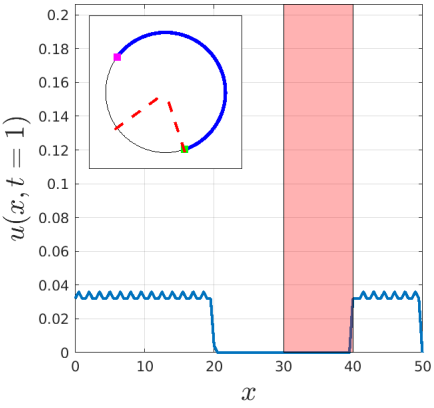

(c) $t = 0.4$ s.




(d) $t = 0.6$ s.

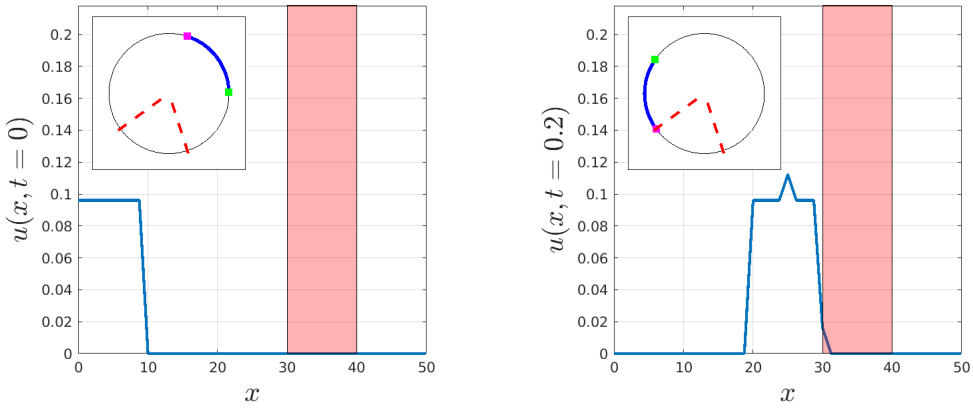



(e) $t = 0.8$ s.




(f) $t = 1$ s.

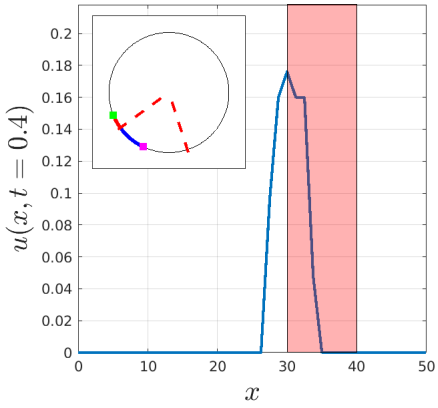
Figure 6: Density $u(x,t)$ against $x$ for different values of time $t$. The red shaded area corresponds to the zone where the cars have to go slower. The smaller figure represents the circular track and the particles travelling on it.



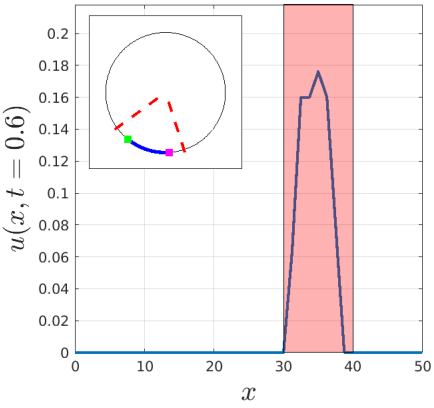

(a) $t = 0$ s.



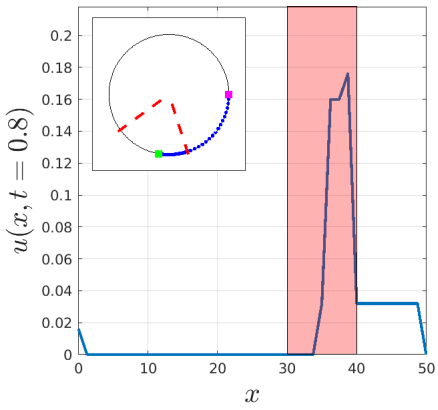

(b) $t = 0.2$ s.



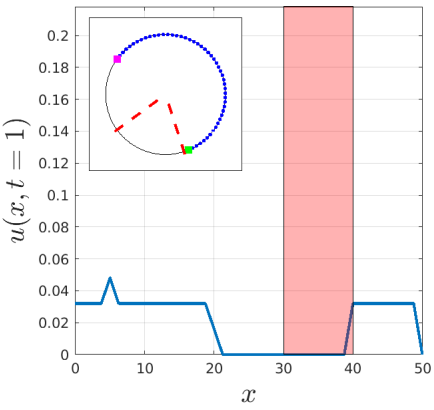

(c) $t = 0.4$ s.




(d) $t = 0.6$ s.




(e) $t = 0.8$ s.




(f) $t = 1$ s.

Figure 7: Density $u(x,t)$ against $x$ for different values of time $t$. The red shaded area corresponds to the zone where the cars have to go slower. The smaller figure represents the circular track and the particles travelling on it.

Matlab code:

```
close all; clear all; clc;

%% Global variables
global Npart L slow_interval v_slow v_fast
% figure(1)
% set(gcf, 'Position', get(0, 'Screensize'));
%% Parameter values
Npart = 500; % Number of particles
L = 50;
d = 0.0099; % Floating point error can cause the conditions to fail
    if exactly 0.01
T = 1;
dt = 0.0001;

% Fixed grid to compute density
M = 100; % Number of grid cells
dy = L/M;
y = 0:dy:L;
u_max = (1/d+1/dy)/Npart; % Maximum posible density

% Speeds of the track
slow_interval = [30 40];
v_slow = 20;
v_fast = 100;

% Initial placement of particles
alpha= 0.02;
xi_0 = alpha*linspace(1,Npart,Npart);
u  = calculate_density(y,dy,M,xi_0);
plot_density(y,u,0,u_max)
plot_particles(xi_0,1:Npart,[])
figName = create_figName(0,Npart);
saveas(gcf,figName,'png')
disp(['Saved figure -> "',figName,'.png"'])
clf

% Prepare for the loop
saveTimes = [0.2 0.4 0.6 0.8 1]*T;
save_idx = 1;
xi = xi_0;
v = zeros(size(xi));
t = 0;
tstep = 0; % Time step counter

while t < T
    [idx_leaders,idx_followers] = find_leaders(xi, d, dt); % Array
    with the indeces of the leaders

    n = 1;
    for i=1:length(idx_leaders)
        idx = idx_leaders(i);
```

```
50          v_leader = calculate_v0(mod(xi(idx),L));
51          v(n:idx) = v_leader; % Match the speed of the leader
52          n = idx + 1;
53      end
54      if (idx_leaders(end)<Npart) % This means the first particle is
      going to hit the last
55          v(n:Npart) = v(1); % Match the velocities of the first and
      last particle
56      end
57      % Update xi, density u, t and tstep
58      xi = mod(xi + dt*v,L);
59      u  = calculate_density(y,dy,M,xi);
60
61      t = round(t + dt,4); % Round avoids floating point errors, I don
      't know why they appear here
62    tstep = tstep + 1;
63
64      if (ismember(t,saveTimes))
65          plot_density(y,u,t,u_max)
66          plot_particles(xi,idx_leaders,idx_followers)
67          figName = create_figName(t,Npart);
68          saveas(gcf,figName,'png')
69          disp(['Saved figure -> "',figName,'.png"'])
70          clf
71      end
72  end
73
74  function [leaders,followers] = find_leaders(xi, d, dt)
75      global Npart L
76
77      leaders = [];
78    followers = [];
79
80      v0 = calculate_v0(mod(xi,L)); % Free track velocity
81      n = 1; % Particle counter
82
83      % Index of the last one in the current lap
84    last_in_lap = find(xi==max(xi));
85
86      while n <= length(xi)
87          % The particle in front of the last one is the first one
88          if n == Npart
89              next = 1;
90          else
91              next = n+1;
92          end
93
94          % Position if particle moved freely
95          futurePosition = xi(n) + dt*v0(n);
96
97          % Check if leader or follower and append to arrays
98          if( n == last_in_lap && futurePosition + d <= xi(next) + L )
```

```
            leaders = [leaders n];
        elseif ( futurePosition + d <= xi(next) )
            leaders = [leaders n];
        else
            followers = [followers n];
        end

        % Update counter
        n = n+1;
    end
end

function v0 = calculate_v0(x)
    global slow_interval v_slow v_fast

    is_slow = x>slow_interval(1) & x<slow_interval(2);
    is_fast = ~is_slow;

    v0 = v_slow*is_slow + v_fast*is_fast;
end

function u = calculate_density(y,dy,M,xi)
    global Npart
    u = zeros(size(y));
    for m=1:M
        I=find((y(m)<xi)&(xi<y(m+1)));
        u(m)=length(I)/Npart/dy;
    end
end

function plot_particles(xi,idx_leaders,idx_followers)
    global L slow_interval

    % Plotting parameters
    linewidth = 2;

    % Radius of the circular track
    R = L/2*pi;

    % Coordinates of the track in the circle
    theta = linspace(0,2*pi,500);
    x = R*cos(theta);
    y = R*sin(theta);

    % Coordinates of first particle in the circle
    theta_first = xi(1)*2*pi/L;
    x_first = R*cos(theta_first);
    y_first = R*sin(theta_first);
    % Coordinates of last particle in the circle
    theta_last = xi(end)*2*pi/L;
    x_last = R*cos(theta_last);
    y_last = R*sin(theta_last);
```

```
151
152     % Coordinates of leaders in the circle
153     theta_leaders = xi(idx_leaders(1:end))*2*pi/L;
154     x_leaders = R*cos(theta_leaders);
155     y_leaders = R*sin(theta_leaders);
156
157     % Coordinates of followers in the circle
158     theta_followers = xi(idx_followers(1:end))*2*pi/L;
159     x_followers = R*cos(theta_followers);
160     y_followers = R*sin(theta_followers);
161
162     % Slow interval
163     theta_slow = 2*pi*slow_interval/L;
164
165     figure(1)
166     axes('Position',[0.2 0.55 0.35 0.35])
167     box on
168
169     % Plot track
170     plot(x,y,'k', 'linewidth',linewidth/4)
171     hold on
172
173     % Plot followers
174     plot(x_followers,y_followers,'r.')
175
176     % Plot leaders
177     plot(x_leaders,y_leaders,'b.')
178
179     % Plot first and last particle
180     plot(x_first,y_first,'gs','MarkerFaceColor','g')
181     plot(x_last,y_last,'ms','MarkerFaceColor','m')
182
183     % Plot slow area
184     pl1 = line([1.05*R*cos(theta_slow(1)) 0],[1.05*R*sin(theta_slow
    (1)) 0],'linewidth',2);
185     pl2 = line([1.05*R*cos(theta_slow(2)) 0],[1.05*R*sin(theta_slow
    (2)) 0],'linewidth',2);
186     pl1.Color = 'red';
187     pl1.LineStyle = '--';
188     pl2.Color = 'red';
189     pl2.LineStyle = '--';
190
191     % Some options
192     grid on
193     pbaspect([1 1 1])
194     set(gca,'Xtick',[])
195     set(gca,'Ytick',[])
196 end
197
198 function plot_density(y, u, t, u_max)
199     global L slow_interval
200     % Plotting parameters
```

```
    linewidth = 2;
    labelfontsize = 20;


    figure(1)
    plot(y,u, 'linewidth',linewidth)
    grid on
    xlabel('$x$','interpreter','latex','fontsize',labelfontsize)
    ylabel(['$u(x,t=',num2str(t),')$'],'interpreter','latex','
    fontsize',labelfontsize)
    axis([0 L 0 u_max])

    % Plot shaded area
    a = slow_interval(1);
    b = slow_interval(2);
    xx = [a a b b];
    yy = [0 u_max u_max 0];
    patch(xx,yy,'red')
    alpha(0.3)
    pbaspect([1 1 1])
end

function figName = create_figName(t,N)

    exponent_t = floor(log10(t));
    base_t = t/10^exponent_t;
    if t==0
        exponent_t = 0;
        base_t = 0;
    end
    exponent_N = floor(log10(N));
    base_N = N/10^exponent_N;
    path = '../figures/';
    figName = append(path,'p4_sol_t',num2str(base_t),'e',num2str(
    exponent_t),'_N',num2str(base_N),'e',num2str(exponent_N));
end
```