MA5710 - Assignment 4

Abhinav T K, MA23M002

November 5, 2023

1. Implement Gudanov Method to numerically solve the first order hyperbolic partial differential equation, that is the scalar conservation law. Assume different speed density relations and initial values as mentioned in the slide provided, on case by case basis(when the signal is red, then red to green and then red again, with and without speed breaker before and after the signal). Generate the tables provided in the slide for each case and subcases.

   Code: **traffic_flow_model.m**

```
% Main file
clear;
clc;

rhoMax = 1;      % Maximum rho
rhoMin = 0;      % Minimum rho
h = 0.1;        % space step size
k = 0.1;        % time step size
x = -1:h:1;      % Road - (-1 to 1)
x_divs = length(x);          % No. of space steps

total_time= 3;               % Total time under consideration
t_steps = total_time / k;    % No. of time steps

% Initial conditions
rho0 = 0.65 * ones(1, x_divs);

u = zeros(t_steps, x_divs);      % Array to store u

% Initial and boundary conditions for u
u(1,:) =  1 - 2 .* rho0/rhoMax ;     % Burger's equation
u(:,1) = u(1,1)*ones(t_steps,1);

% Signal
x_signal = 0.2;      % Position of signal
x_signal_idx = floor((x_signal - x(1))/h);  % Index of signal position in x array

signal_times = [0.5, 1.5, 2.5];               % Signal timings (in min.)
signal_time_idx = signal_times/k;           % Index of signal time in time array


f = @(u) (u.^2)/2;  % Flux function
df = @(u) u;        % Derivative of Flux

% Apply boundary condition (1)
for i = 1:2:length(signal_time_idx)-1
   u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx) = -1;
   u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx + 1) = 1;
end

```

```
% Numerical calculation using Godunov method
for t = 1:t_steps-1
    % Apply boundary condition (2)
    for i = 1:2:length(signal_time_idx)-1
       u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx) = -1;
       u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx + 1) = 1;
    end

    for i = 2:x_divs - 1
        u(t+1,i) = u(t,i) - (k/h)*(f(u_star(t,i, u,f, df))-f(u_star(t,i-1, u,f, df))
           );
    end
end
u(:,1) = u(:,2);
u(:,end) = u(:,end-1);

% Get rho value from u
rho_tab =  (rhoMax / 2).*(1 - u);


% Plotting at different cases
figure;
% First subplot - when green light (initial cond.)
subplot(2, 2, 1);
plot(x, rho_tab(1,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('t = 0, Uniform traffic flow (Green signal)');
% Display grid
grid on;
hold on;
% Add a vertical line at signal position
xline(x_signal, 'g--', 'LineWidth', 2); % Signal position
hold off;

% Second subplot - when red light
subplot(2, 2, 2);
plot(x, rho_tab(10,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('During Red signal');
grid on;
hold on;
% Add a vertical line at signal position
xline(x_signal, 'r--', 'LineWidth', 2);
hold off;

% Third subplot - when red light (just before green light)
subplot(2, 2, 3);
plot(x, rho_tab(15,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('Just before Green signal');
grid on;
hold on;
% Add a vertical line at signal position
```

```
xline(x_signal, 'r--', 'LineWidth', 2);
hold off;

% forth subplot
subplot(2, 2, 4);
plot(x, rho_tab(25,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('Traffic flow 1 min after signal turned green');
grid on;
hold on;
% Add a vertical line at signal position
xline(x_signal, 'g--', 'LineWidth', 2);
hold off;



% Function for u_star calculation used in Godunov scheme
function u_star_out = u_star(t, i, u, f, df)
    gradF = df(u(t, i));         % Gradient of flux for u(t,i)
    gradF1 = df(u(t, i + 1));    % Gradient of flux for u(t,i+1)
    % Apply different conditions...
    if gradF >= 0 && gradF1 >= 0
        u_star_out = u(t, i);
    elseif gradF < 0 && gradF1 < 0
        u_star_out = u(t, i + 1);
    elseif gradF >= 0 && gradF1 < 0
        s = (f(u(t, i + 1))-f(u(t, i)))/(u(t, i + 1)- u(t, i)) ;
        if s >= 0
            u_star_out = u(t, i);
        else
            u_star_out = u(t, i + 1);
        end
    else
        u_star_out = 0;
    end
end
```

Output:

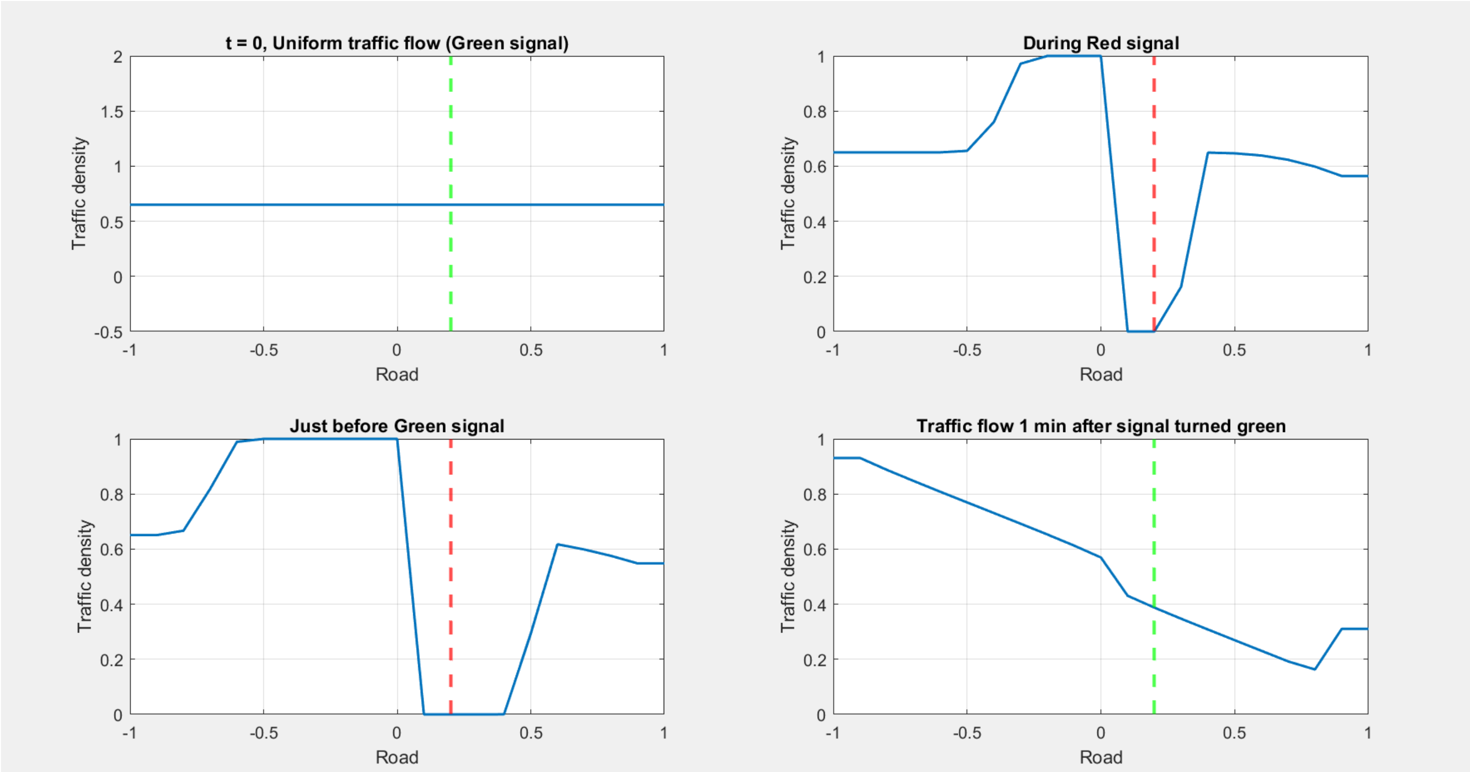


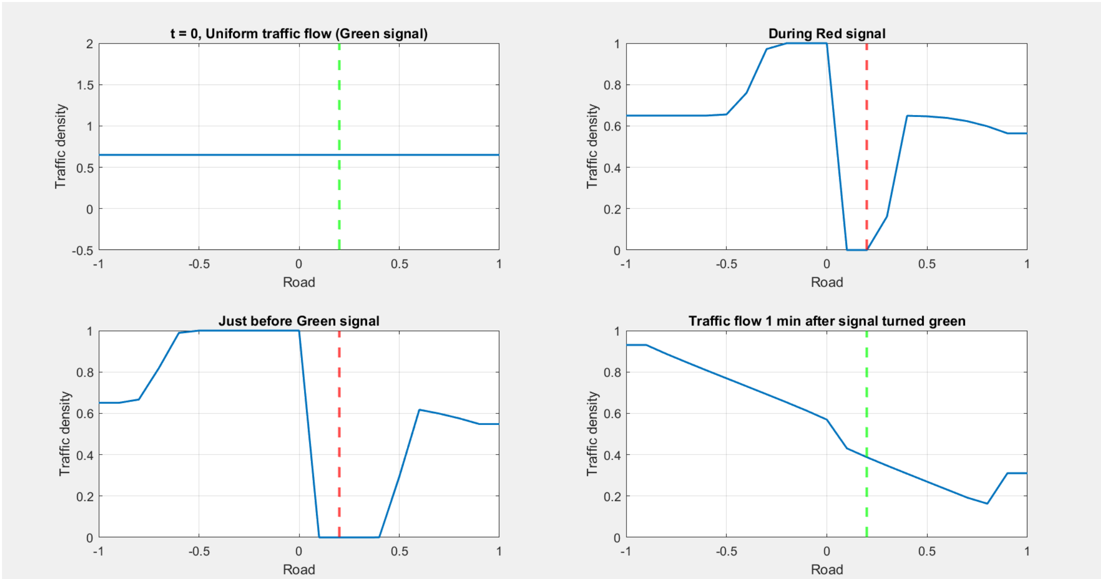
Figure 1: Density vs Road length for different traffic signal conditions

												Traffic signal position										
	t/x	-1	-0.9	-0.8	-0.7	-0.6	-0.5	-0.4	-0.3	-0.2	-0.1	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
	0	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
	0.1	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
	0.2	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.6275	0.6275
	0.3	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.643756	0.611244	0.611244
Signal turned red	0.4	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	1	0	0.65	0.65	0.65	0.65	0.65	0.648166	0.635466	0.598869	0.598869
	0.5	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.8775	1	0	0.4225	0.65	0.65	0.65	0.649453	0.644564	0.62689	0.589094	0.589094
	0.6	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.770006	0.984994	1	0	0.195	0.65	0.65	0.649836	0.648016	0.639766	0.618726	0.581156	0.581156
	0.7	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.700403	0.932322	0.999775	1	0	0.038025	0.579475	0.649951	0.649294	0.645641	0.634327	0.611217	0.57457	0.57457
	0.8	0.65	0.65	0.65	0.65	0.65	0.65	0.667662	0.847144	0.995195	1	1	0	0.001446	0.388539	0.649754	0.648217	0.642474	0.628653	0.604408	0.569009	0.569009
	0.9	0.65	0.65	0.65	0.65	0.65	0.65561	0.76006	0.971853	0.999977	1	1	0	2.09E-06	0.16241	0.649296	0.646547	0.638727	0.623002	0.598269	0.564247	0.564247
	1	0.65	0.65	0.65	0.65	0.651715	0.699027	0.915074	0.999185	1	1	1	0	4.37E-12	0.026379	0.556805	0.644316	0.634611	0.61753	0.59274	0.560119	0.560119
	1.1	0.65	0.65	0.65	0.650517	0.668309	0.831702	0.991973	0.999999	1	1	1	0	0	0.000696	0.353315	0.641609	0.630304	0.612317	0.587754	0.556505	0.556505
	1.2	0.65	0.65	0.650155	0.65619	0.750007	0.963713	0.999935	1	1	1	1	0	0	4.84E-07	0.125527	0.637072	0.62594	0.607403	0.583246	0.553312	0.553312
	1.3	0.650047	0.650047	0.652004	0.694298	0.902533	0.998618	1	1	1	1	1	0	0	2.34E-13	0.015758	0.512703	0.621615	0.602797	0.579158	0.55047	0.55047
	1.4	0.650638	0.650638	0.666651	0.818579	0.98912	0.999998	1	1	1	1	1	0	0	0	0.000248	0.293002	0.617392	0.598496	0.575439	0.547923	0.547923
Signal turned green	1.5	0.655719	0.655719	0.740371	0.956325	0.99988	1	1	1	1	1	0.75	0.25	0	0	6.17E-08	0.086098	0.584245	0.594486	0.572045	0.545626	0.545626
	1.6	0.689248	0.689248	0.890825	0.997972	1	1	1	1	1	0.8125	0.6875	0.3125	0.1875	0	3.77E-15	0.007413	0.421858	0.590748	0.568936	0.543544	0.543544
	1.7	0.806178	0.806178	0.986057	0.999996	1	1	1	1	0.847656	0.75	0.652344	0.347656	0.25	0.152344	0	5.50E-05	0.187451	0.587265	0.56608	0.541648	0.541648
	1.8	0.948685	0.948685	0.999801	1	1	1	1	0.870865	0.789291	0.710709	0.629135	0.370865	0.289291	0.210709	0.129135	3.02E-09	0.035193	0.493945	0.563448	0.539914	0.539914
	1.9	0.997168	0.997168	1	1	1	1	0.887541	0.817014	0.75	0.682986	0.612459	0.387541	0.317014	0.25	0.182986	0.112459	0.001239	0.281925	0.561015	0.538321	0.538321
	2	0.999992	0.999992	1	1	1	0.900188	0.837851	0.779016	0.720984	0.662149	0.599812	0.400188	0.337851	0.279016	0.220984	0.162149	0.099814	0.080719	0.514927	0.536852	0.536852
	2.1	1	1	1	1	0.91015	0.854181	0.801557	0.75	0.698443	0.645819	0.58985	0.41015	0.354181	0.301557	0.25	0.198443	0.14582	0.096366	0.340489	0.535494	0.535494
	2.2	1	1	1	0.918223	0.867371	0.819674	0.773121	0.726879	0.680326	0.632629	0.581777	0.418223	0.367371	0.319674	0.273121	0.226879	0.180326	0.133843	0.203013	0.51005	0.51005
	2.3	1	1	0.924911	0.878274	0.834601	0.792077	0.75	0.707923	0.665399	0.621726	0.575089	0.424911	0.378274	0.334601	0.292077	0.25	0.207923	0.165723	0.157143	0.421849	0.421849
	2.4	0.930549	0.930549	0.887453	0.84714	0.807952	0.769268	0.730732	0.692048	0.65286	0.612547	0.569451	0.430549	0.387453	0.34714	0.307952	0.269268	0.230732	0.192155	0.162953	0.310406	0.310406
	2.5	0.895296	0.895296	0.85784	0.821468	0.785623	0.75	0.714377	0.678532	0.64216	0.604704	0.564627	0.435373	0.395296	0.35784	0.321468	0.285623	0.25	0.214418	0.181785	0.232751	0.232751
	2.6	0.867086	0.867086	0.833132	0.799707	0.766542	0.733458	0.700293	0.666868	0.632914	0.597918	0.560451	0.439549	0.402082	0.367086	0.333132	0.299707	0.266542	0.233475	0.201489	0.202912	0.202912
	2.7	0.843311	0.843311	0.811979	0.780928	0.75	0.719072	0.688021	0.656689	0.624836	0.591984	0.556796	0.443204	0.408016	0.375164	0.343311	0.311979	0.280928	0.250008	0.219562	0.202064	0.202064
	2.8	0.82278	0.82278	0.793569	0.764507	0.735493	0.706431	0.67722	0.647721	0.617713	0.586749	0.553571	0.446429	0.413251	0.382287	0.352279	0.32278	0.293569	0.264511	0.235712	0.212185	0.212185
	2.9	0.804776	0.804776	0.77735	0.75	0.72265	0.695224	0.667635	0.639756	0.611382	0.582093	0.550701	0.449299	0.417907	0.388618	0.360244	0.332365	0.304776	0.277352	0.250105	0.225174	0.225174

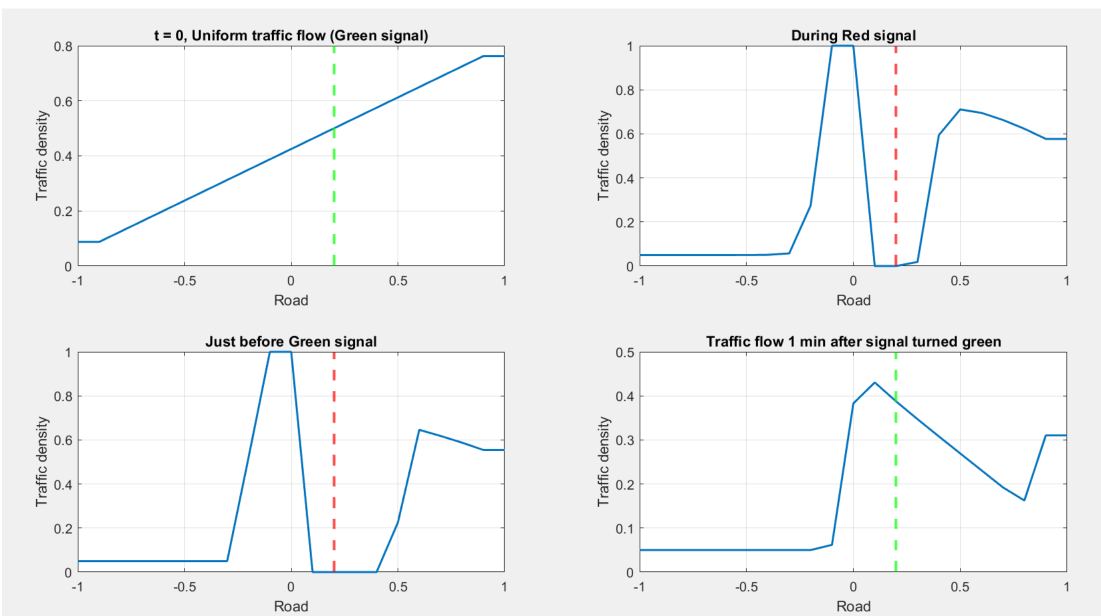
Figure 2: Output Density data

Different initial conditions (Comparisons):

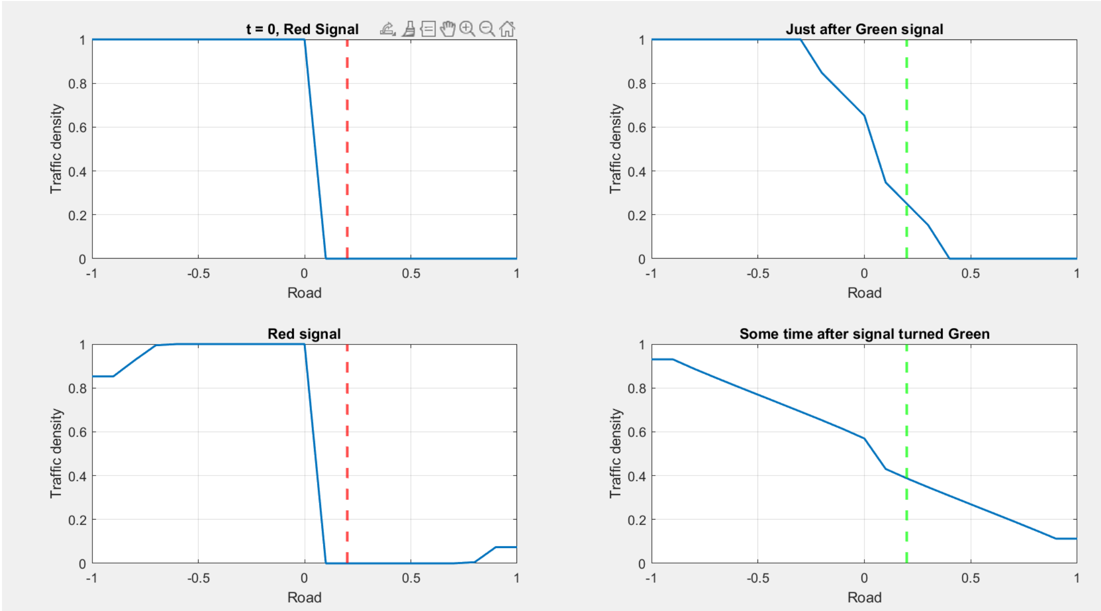
1. **rho0 = 0.65 * ones(1, x_divs)**



2. **rho0 = linspace(0.05, 0.8, x_divs)**



3. **rho0(**$x <= 0$**) = 1; rho0(**$x > 0$**) = 0;**


t = 0, Red Signal
Just after Green signal
Red signal
Some time after signal turned Green
Traffic density
Road

Traffic flow with speed breaker before signal:
Code: **traffic_flow_model_spdbrk_left.m**

```
% Traffic flow model with speed breaker before signal
clear;
clc;

rhoMax = 1;      % Maximum rho
rhoMin = 0;      % Minimum rho
h = 0.1;        % space step size
k = 0.1;        % time step size
x = -1:h:1;      % Road - (-1 to 1)
x_divs = length(x);          % No. of space steps

total_time= 3;               % Total time under consideration
t_steps = total_time / k;    % No. of time steps

% Initial conditions
rho0 = 0.65 * ones(1, x_divs);

% rho0 = np.linspace(0.1, 0.5, num_divs)
%  rho0 = np.zeros(num_divs)
%  rho0[x <= 0] = 1
%  rho0[x > 0] = 0

u = zeros(t_steps, x_divs);      % Array to store u

% Initial and boundary conditions for u
u(1,:) =  1 - 2 .* rho0/rhoMax ;
u(:,1) = u(1,1)*ones(t_steps,1);

% Signal
x_signal = 0.2;      % Position of signal
x_signal_idx = floor((x_signal - x(1))/h);  % Index of signal position in x array

signal_times = [0.5, 1.5, 2.5];              % Signal timings (in min.)
signal_time_idx = signal_times/k;            % Index of signal time in time array

% Speed breaker
x_spd_brk = -0.2;                                % Position of speed breaker
x_spd_brk_idx = floor((x_spd_brk - x(1))/h);     % Index of speed breaker position in
    x array
v_spd_brk = 0.15;                                % Velocity at speed breaker

f = @(u) (u.^2)/2;  % Flux function
df = @(u) u;        % Derivative of Flux

% Apply boundary condition (1)
for i = 1:2:length(signal_time_idx)-1
   u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx) = -1;
   u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx + 1) = 1;
end
u(:, x_spd_brk_idx) = -1;

% Numerical calculation using Godunov method
for t = 1:t_steps-1
    % Apply boundary condition (2)
```

```
    for i = 1:2:length(signal_time_idx)-1
       u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx) = -1;
       u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx + 1) = 1;
    end
    u(:, x_spd_brk_idx) = -1;

    for i = 2:x_divs - 1
        u(t+1,i) = u(t,i) - (k/h)*(f(u_star(t,i, u,f, df))-f(u_star(t,i-1, u,f, df))
            );
        if i == x_spd_brk_idx
            u(t + 1, i) = v_spd_brk * u(t + 1, i);
        end
    end
end
u(:,1) = u(:,2);
u(:,end) = u(:,end-1);

% Get rho value from u
rho_tab =  (rhoMax / 2).*(1 - u);


% Plotting at different cases
figure;
% First subplot - when green light (initial cond.)
subplot(2, 2, 1);
plot(x, rho_tab(1,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('t = 0, Uniform traffic flow (Green signal)');
% Display grid
grid on;
hold on;
% Add a vertical line at signal position
xline(x_signal, 'g--', 'LineWidth', 2); % Signal position
xline(x_spd_brk, 'm--', 'LineWidth', 1); % Speed breaker position
hold off;

% Second subplot - when red light
subplot(2, 2, 2);
plot(x, rho_tab(10,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('During Red signal');
grid on;
hold on;
xline(x_signal, 'r--', 'LineWidth', 2); % Add a vertical line at signal position
xline(x_spd_brk, 'm--', 'LineWidth', 1); % Speed breaker position
hold off;

% Third subplot - when red light (just before green light)
subplot(2, 2, 3);
plot(x, rho_tab(15,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('Just before Green signal');
grid on;
```

```
hold on;
xline(x_signal, 'r--', 'LineWidth', 2); % Add a vertical line at signal position
xline(x_spd_brk, 'm--', 'LineWidth', 1); % Speed breaker position
hold off;

% forth subplot
subplot(2, 2, 4);
plot(x, rho_tab(25,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('Traffic flow 1 min after signal turned green');
grid on;
hold on;
xline(x_signal, 'g--', 'LineWidth', 2); % Add a vertical line at signal position
xline(x_spd_brk, 'm--', 'LineWidth', 1); % Speed breaker position
hold off;


% Function for u_star calculation used in Godunov scheme
function u_star_out = u_star(t, i, u, f, df)
    gradF = df(u(t, i));        % Gradient of flux for given u(n,i)
    gradF1 = df(u(t, i + 1));   % Gradient of flux for given u(n,i+1)
    % Apply different conditions...
    if gradF >= 0 && gradF1 >= 0
        u_star_out = u(t, i);
    elseif gradF < 0 && gradF1 < 0
        u_star_out = u(t, i + 1);
    elseif gradF >= 0 && gradF1 < 0
        s = (f(u(t, i + 1))-f(u(t, i)))/(u(t, i + 1)- u(t, i)) ;
        if s >= 0
            u_star_out = u(t, i);
        else
            u_star_out = u(t, i + 1);
        end
    else
        u_star_out = 0;
    end
end
```

Output:

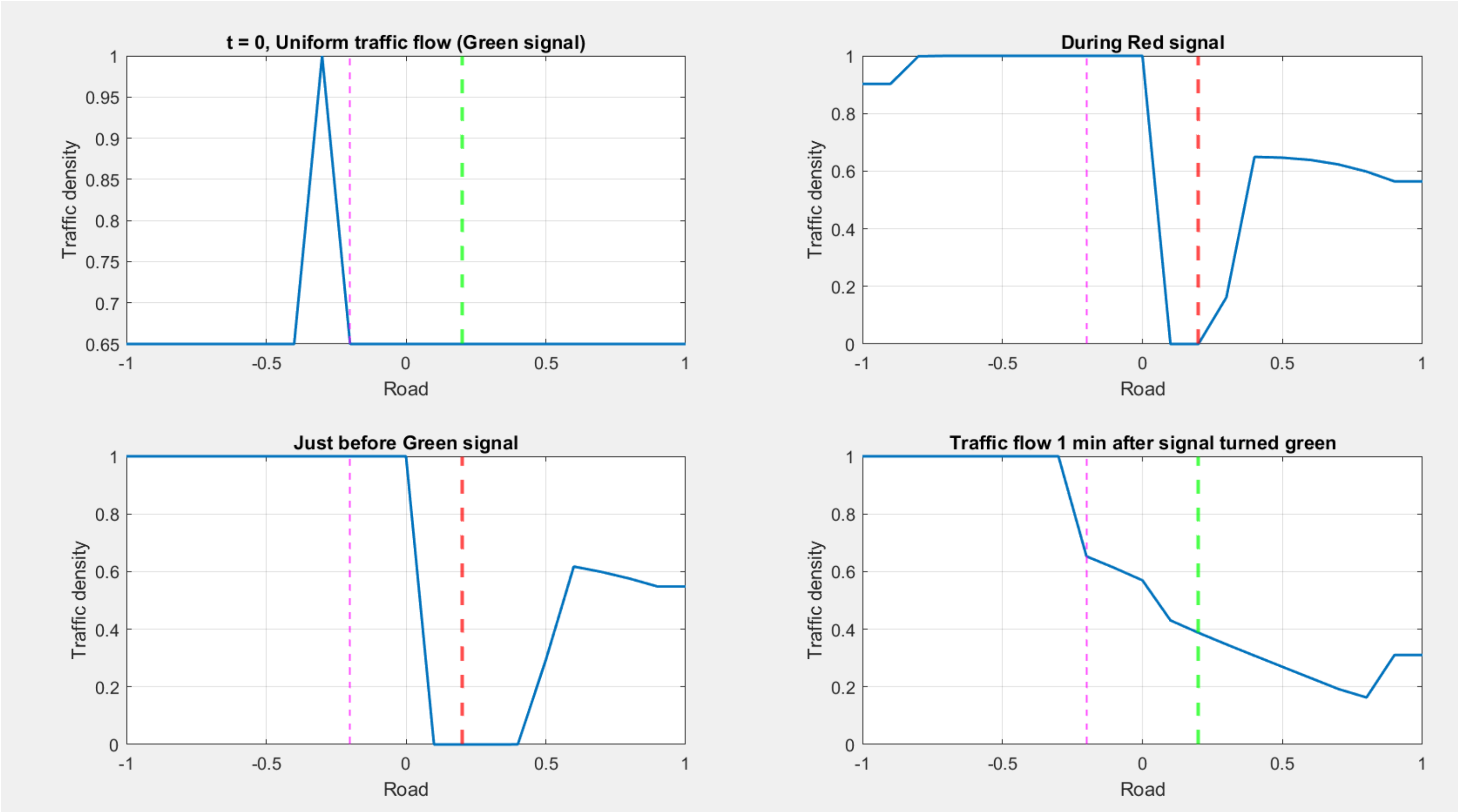


Figure 3: Density plot for different traffic signal conditions with Speed breaker before signal

									Spd brk pos.			Traffic signal position										
	t/x	-1	-0.9	-0.8	-0.7	-0.6	-0.5	-0.4	-0.3	-0.2	-0.1	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
	0	0.65	0.65	0.65	0.65	0.65	0.65	0.65	1	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
	0.1	0.65	0.65	0.65	0.65	0.65	0.65	0.8775	1	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65
	0.2	0.65	0.65	0.65	0.65	0.65	0.770006	0.984994	1	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.6275	0.6275
	0.3	0.65	0.65	0.65	0.65	0.700403	0.932322	0.999775	1	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.643756	0.611244	0.611244
Signal turned red	0.4	0.65	0.65	0.65	0.667662	0.847144	0.995195	1	1	0.65	0.65	1	0	0.65	0.65	0.65	0.65	0.65	0.648166	0.635466	0.598869	0.598869
	0.5	0.65	0.65	0.65561	0.76006	0.971853	0.999977	1	1	0.65	0.8775	1	0	0.4225	0.65	0.65	0.65	0.649453	0.644564	6.27E-01	0.589094	0.589094
	0.6	0.651715	0.651715	0.699027	0.915074	0.999185	1	1	1	0.770006	0.984994	1	0	0.195	0.65	0.65	0.649836	0.648016	0.639766	6.19E-01	0.581156	0.581156
	0.7	0.668309	0.668309	0.831702	0.991973	0.999999	1	1	1	0.932322	0.999775	1	0	0.038025	0.579475	0.649951	0.649294	0.645641	0.634327	0.611217	0.57457	0.57457
	0.8	0.750007	0.750007	0.963713	0.999935	1	1	1	1	0.995195	1	1	0	0.001446	0.388539	0.649754	0.648217	0.642474	0.628653	0.604408	5.69E-01	5.69E-01
	0.9	0.902533	0.902533	0.998618	1	1	1	1	1	0.999977	1	1	0	2.09E-06	0.16241	0.649296	0.646547	0.638727	0.623002	0.598269	5.64E-01	5.64E-01
	1	0.98912	0.98912	0.999998	1	1	1	1	1	1	1	1	0	4.37E-12	0.026379	0.556805	0.644316	0.634611	0.61753	0.59274	0.560119	0.560119
	1.1	0.99988	0.99988	1	1	1	1	1	1	1	1	1	0	0	0.000696	0.353315	0.641609	0.630304	0.612317	0.587754	0.556505	0.556505
	1.2	1	1	1	1	1	1	1	1	1	1	1	0	0	4.84E-07	0.125527	0.637072	0.62594	0.607403	0.583246	0.553312	0.553312
	1.3	1	1	1	1	1	1	1	1	1	1	1	0	0	2.34E-13	0.015758	0.512703	0.621615	0.602797	0.579158	0.55047	0.55047
	1.4	1	1	1	1	1	1	1	1	1	1	1	0	0	0	2.48E-04	0.293002	0.617392	0.598496	0.575439	0.547923	0.547923
Signal turned green	1.5	1	1	1	1	1	1	1	1	1	1	0.75	0.25	0	0	6.17E-08	0.086098	0.584245	0.594486	0.572045	0.545626	0.545626
	1.6	1	1	1	1	1	1	1	1	1	0.8125	0.6875	0.3125	0.1875	0	3.77E-15	7.41E-03	0.421858	0.590748	0.568936	0.543544	0.543544
	1.7	1	1	1	1	1	1	1	1	0.847656	0.75	0.652344	0.347656	0.25	0.152344	0	5.50E-05	0.187451	0.587265	0.56608	0.541648	0.541648
	1.8	1	1	1	1	1	1	1	1	0.789291	0.710709	0.629135	0.370865	0.289291	0.210709	0.129135	3.02E-09	0.035193	0.493945	0.563448	0.539914	0.539914
	1.9	1	1	1	1	1	1	1	1	0.75	0.682986	0.612459	0.387541	0.317014	0.25	0.182986	0.112459	0.001239	0.281925	0.561015	0.538321	0.538321
	2	1	1	1	1	1	1	1	1	0.720984	0.662149	0.599812	0.400188	0.337851	0.279016	0.220984	0.162149	0.099814	0.080719	0.514927	0.536852	0.536852
	2.1	1	1	1	1	1	1	1	1	0.698443	0.645819	0.58985	0.41015	0.354181	0.301557	0.25	0.198443	0.14582	0.096366	0.340489	0.535494	0.535494
	2.2	1	1	1	1	1	1	1	1	0.680326	0.632629	0.581777	0.418223	0.367371	0.319674	0.273121	0.226879	0.180326	0.133843	0.203013	0.51005	0.51005
	2.3	1	1	1	1	1	1	1	1	0.665399	0.621726	0.575089	0.424911	0.378274	0.334601	0.292077	0.25	0.207923	0.165723	0.157143	0.421849	0.421849
	2.4	1	1	1	1	1	1	1	1	0.65286	0.612547	0.569451	0.430549	0.387453	0.34714	0.307952	0.269268	0.230732	0.192155	0.162953	0.310406	0.310406
	2.5	1	1	1	1	1	1	1	1	0.64216	0.604704	0.564627	0.435373	0.395296	0.35784	0.321468	0.285623	0.25	0.214418	0.181785	0.232751	0.232751
	2.6	1	1	1	1	1	1	1	1	0.632914	0.597918	0.560451	0.439549	0.402082	0.367086	0.333132	0.299707	0.266542	0.233475	0.201489	0.202912	0.202912
	2.7	1	1	1	1	1	1	1	1	0.624836	0.591984	0.556796	0.443204	0.408016	0.375164	0.343311	0.311979	0.280928	0.250008	0.219562	0.202064	0.202064
	2.8	1	1	1	1	1	1	1	1	0.617713	0.586749	0.553571	0.446429	0.413251	0.382287	0.352279	0.32278	0.293569	0.264511	0.235712	0.212185	0.212185
	2.9	1	1	1	1	1	1	1	0.539578	0.611382	0.582093	0.550701	0.449299	0.417907	0.388618	0.360244	0.332365	0.304776	0.277352	0.250105	0.225174	0.225174

Figure 4: Density data with speed breaker and signal

Traffic flow with speed breaker after signal:

Code: **traffic_flow_model_spdbrk_right.m**

```
% Traffic flow model with speed breaker after signal
clear;
clc;

rhoMax = 1;      % Maximum rho
rhoMin = 0;      % Minimum rho
h = 0.1;        % space step size
k = 0.1;        % time step size
x = -1:h:1;      % Road - (-1 to 1)
x_divs = length(x);          % No. of space steps

total_time= 3;               % Total time under consideration
t_steps = total_time / k;    % No. of time steps

% Initial conditions
rho0 = 0.65 * ones(1, x_divs);

% rho0 = np.linspace(0.1, 0.5, num_divs)
%  rho0 = np.zeros(num_divs)
%  rho0[x <= 0] = 1
%  rho0[x > 0] = 0

u = zeros(t_steps, x_divs);      % Array to store u

% Initial and boundary conditions for u
u(1,:) =  1 - 2 .* rho0/rhoMax ;
u(:,1) = u(1,1)*ones(t_steps,1);

% Signal
x_signal = 0.2;      % Position of signal
x_signal_idx = floor((x_signal - x(1))/h);  % Index of signal position in x array

signal_times = [0.5, 1.5, 2.5];               % Signal timings (in min.)
signal_time_idx = signal_times/k;            % Index of signal time in time array

% Speed breaker
x_spd_brk = 0.7;                                 % Position of speed breaker
x_spd_brk_idx = floor((x_spd_brk - x(1))/h);     % Index of speed breaker position in
    x array
v_spd_brk = 0.15;                                % Velocity at speed breaker

f = @(u) (u.^2)/2;  % Flux function
df = @(u) u;        % Derivative of Flux

% Apply boundary condition (1)
for i = 1:2:length(signal_time_idx)-1
   u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx) = -1;
   u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx + 1) = 1;
end
u(:, x_spd_brk_idx+1) = 1;

% Numerical calculation using Godunov method
for t = 1:t_steps-1
```

```
    % Apply boundary condition (2)
    for i = 1:2:length(signal_time_idx)-1
       u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx) = -1;
       u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx + 1) = 1;
    end
    u(:, x_spd_brk_idx+1) = 1;

    for i = 2:x_divs - 1
        u(t+1,i) = u(t,i) - (k/h)*(f(u_star(t,i, u,f, df))-f(u_star(t,i-1, u,f, df))
            );
        if i == x_spd_brk_idx
            u(t + 1, i) = v_spd_brk * u(t + 1, i);
        end
    end
end
u(:,1) = u(:,2);
u(:,end) = u(:,end-1);

% Get rho value from u
rho_tab =  (rhoMax / 2).*(1 - u);


% Plotting at different cases
figure;
% First subplot - when green light (initial cond.)
subplot(2, 2, 1);
plot(x, rho_tab(1,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('t = 0, Uniform traffic flow (Green signal)');
% Display grid
grid on;
hold on;
% Add a vertical line at signal position
xline(x_signal, 'g--', 'LineWidth', 2); % Signal position
xline(x_spd_brk, 'm--', 'LineWidth', 1); % Speed breaker position
hold off;

% Second subplot - when red light
subplot(2, 2, 2);
plot(x, rho_tab(10,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('During Red signal');
grid on;
hold on;
xline(x_signal, 'r--', 'LineWidth', 2); % Add a vertical line at signal position
xline(x_spd_brk, 'm--', 'LineWidth', 1); % Speed breaker position
hold off;

% Third subplot - when red light (just before green light)
subplot(2, 2, 3);
plot(x, rho_tab(15,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('Just before Green signal');
```

```
grid on;
hold on;
xline(x_signal, 'r--', 'LineWidth', 2); % Add a vertical line at signal position
xline(x_spd_brk, 'm--', 'LineWidth', 1); % Speed breaker position
hold off;

% forth subplot
subplot(2, 2, 4);
plot(x, rho_tab(25,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('Traffic flow 1 min after signal turned green');
grid on;
hold on;
xline(x_signal, 'g--', 'LineWidth', 2); % Add a vertical line at signal position
xline(x_spd_brk, 'm--', 'LineWidth', 1); % Speed breaker position
hold off;


% Function for u_star calculation used in Godunov scheme
function u_star_out = u_star(t, i, u, f, df)
    gradF = df(u(t, i));         % Gradient of flux for given u(n,i)
    gradF1 = df(u(t, i + 1));    % Gradient of flux for given u(n,i+1)
    % Apply different conditions...
    if gradF >= 0 && gradF1 >= 0
        u_star_out = u(t, i);
    elseif gradF < 0 && gradF1 < 0
        u_star_out = u(t, i + 1);
    elseif gradF >= 0 && gradF1 < 0
        s = (f(u(t, i + 1))-f(u(t, i)))/(u(t, i + 1)- u(t, i)) ;
        if s >= 0
            u_star_out = u(t, i);
        else
            u_star_out = u(t, i + 1);
        end
    else
        u_star_out = 0;
    end
end
```

Output:

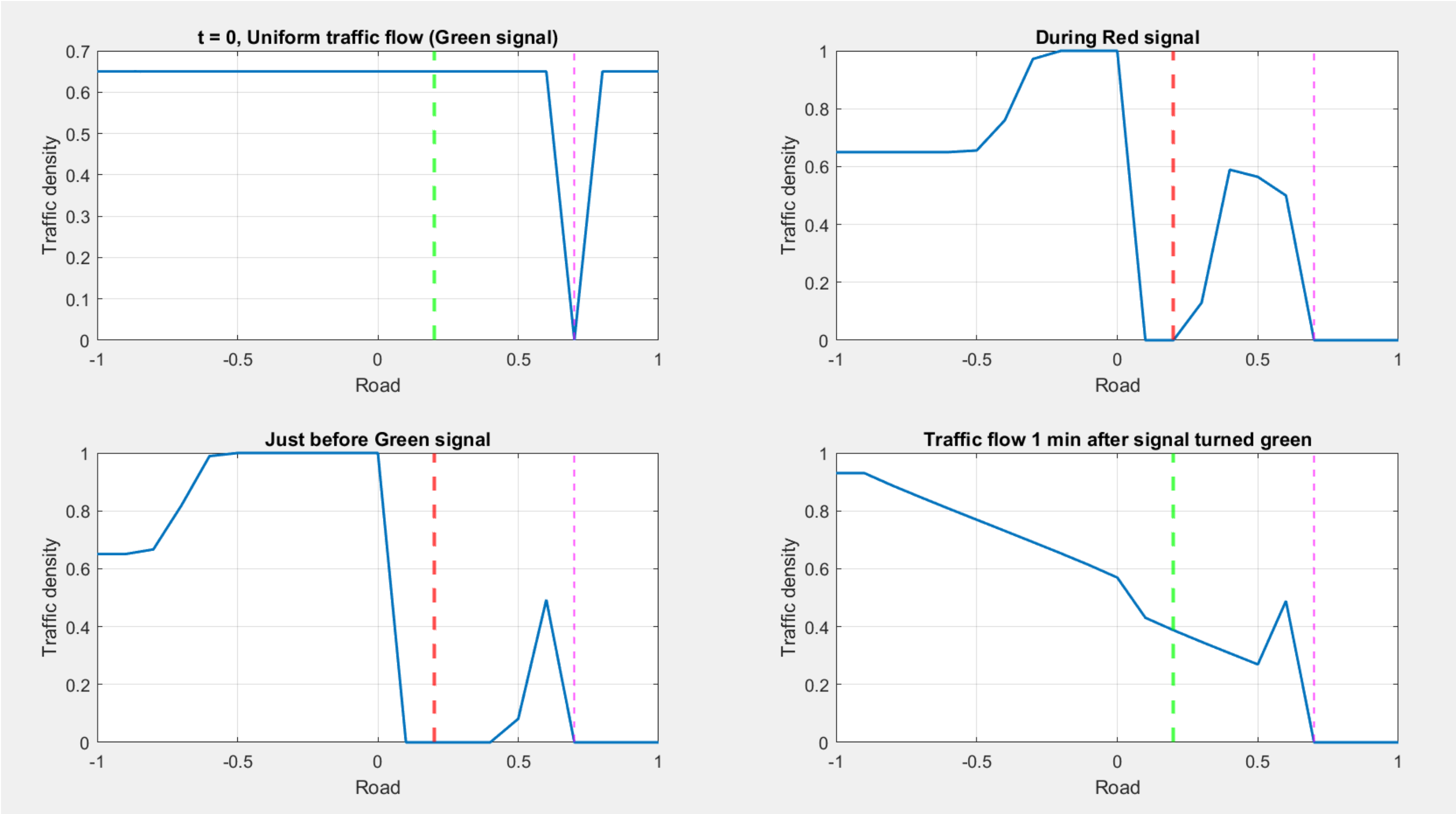


Figure 5: Density plot for different traffic signal conditions with Speed breaker before signal

													Traffic signal position						Speed breaker pos			
	t/x	-1	-0.9	-0.8	-0.7	-0.6	-0.5	-0.4	-0.3	-0.2	-0.1	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
	0	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0	0.65	0.65	0.65
	0.1	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.519125	0	0.4225	0.65	0.65
	0.2	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.627866	0.502814	0	0.195	0.6275	0.6275
	0.3	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.64385	0.611524	0.500421	0	0.038025	0.534475	0.534475
Signal turned red	0.4	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	1	0	0.65	0.648193	0.635595	0.599087	0.500063	0	0.001446	0.321054	0.321054
	0.5	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.8775	1	0	0.421961	0.644618	0.627027	0.589268	0.500009	0	2.09E-06	0.10452	0.10452
	0.6	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.770006	0.984994	1	0	0.192875	0.639839	0.61886	0.5813	0.500001	0	4.37E-12	0.010926	0.010926
	0.7	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.700403	0.932322	0.999775	1	0	0.037201	0.559641	0.611342	0.57469	0.5	0	0	0.000119	0.000119
	0.8	0.65	0.65	0.65	0.65	0.65	0.65	0.667662	0.847144	0.995195	1	1	0	0.001384	0.357855	0.604523	0.569111	0.5	0	0	1.43E-08	1.43E-08
	0.9	0.65	0.65	0.65	0.65	0.65	0.65561	0.76006	0.971853	0.999977	1	1	0	1.92E-06	0.129442	0.589095	0.564335	0.5	0	0	2.22E-16	2.22E-16
	1	0.65	0.65	0.65	0.65	0.651715	0.699027	0.915074	0.999185	1	1	1	0	3.67E-12	0.016757	0.455921	0.560196	0.5	0	0	0	0
	1.1	0.65	0.65	0.65	0.650517	0.668309	0.831702	0.991973	0.999999	1	1	1	0	0	0.000281	0.226021	0.556572	0.5	0	0	0	0
	1.2	0.65	0.65	0.650155	0.65619	0.750007	0.963713	0.999935	1	1	1	1	0	0	7.89E-08	0.051366	0.481508	0.5	0	0	0	0
	1.3	0.650047	0.650047	0.652004	0.694298	0.902533	0.998618	1	1	1	1	1	0	0	6.22E-15	0.002639	0.280577	0.499949	0	0	0	0
	1.4	0.650638	0.650638	0.666651	0.818579	0.98912	0.999998	1	1	1	1	1	0	0	0	6.96E-06	0.081355	0.49277	0	0	0	0
Signal turned green	1.5	0.655719	0.655719	0.740371	0.956325	0.99988	1	1	1	1	1	0.75	0.25	0	0	4.85E-11	0.006626	0.472634	0	0	0	0
	1.6	0.689248	0.689248	0.890825	0.997972	1	1	1	1	1	0.8125	0.6875	0.3125	0.1875	0	0.00E+00	4.39E-05	0.459495	0	0	0	0
	1.7	0.806178	0.806178	0.986057	0.999996	1	1	1	1	0.847656	0.75	0.652344	0.347656	0.25	0.152344	0	1.93E-09	0.456677	0	0	0	0
	1.8	0.948685	0.948685	0.999801	1	1	1	1	0.870865	0.789291	0.710709	0.629135	0.370865	0.289291	0.210709	0.129135	0.00E+00	0.456283	0	0	0	0
	1.9	0.997168	0.997168	1	1	1	1	0.887541	0.817014	0.75	0.682986	0.612459	0.387541	0.317014	0.25	0.182986	0.112459	0.456229	0	0	0	0
	2	0.999992	0.999992	1	1	1	0.900188	0.837851	0.779016	0.720984	0.662149	0.599812	0.400188	0.337851	0.279016	0.220984	0.162149	0.471194	0	0	0	0
	2.1	1	1	1	1	0.91015	0.854181	0.801557	0.75	0.698443	0.645819	0.58985	0.41015	0.354181	0.301557	0.25	0.198443	0.478682	0	0	0	0
	2.2	1	1	1	0.918223	0.867371	0.819674	0.773121	0.726879	0.680326	0.632629	0.581777	0.418223	0.367371	0.319674	0.273121	0.226879	0.48323	0	0	0	0
	2.3	1	1	0.924911	0.878274	0.834601	0.792077	0.75	0.707923	0.665399	0.621726	0.575089	0.424911	0.378274	0.334601	0.292077	0.25	0.486337	0	0	0	0
	2.4	0.930549	0.930549	0.887453	0.84714	0.807952	0.769268	0.730732	0.692048	0.65286	0.612547	0.569451	0.430549	0.387453	0.34714	0.307952	0.269268	0.488604	0	0	0	0
	2.5	0.895296	0.895296	0.85784	0.821468	0.785623	0.75	0.714377	0.678532	0.64216	0.604704	0.564627	0.435373	0.395296	0.35784	0.321468	0.285623	0.490324	0	0	0	0
	2.6	0.867086	0.867086	0.833132	0.799707	0.766542	0.733458	0.700293	0.666868	0.632914	0.597918	0.560451	0.439549	0.402082	0.367086	0.333132	0.299707	0.491669	0	0	0	0
	2.7	0.843311	0.843311	0.811979	0.780928	0.75	0.719072	0.688021	0.656689	0.624836	0.591984	0.556796	0.443204	0.408016	0.375164	0.343311	0.311979	0.492743	0	0	0	0
	2.8	0.82278	0.82278	0.793569	0.764507	0.735493	0.706431	0.67722	0.647721	0.617713	0.586749	0.553571	0.446429	0.413251	0.382287	0.352279	0.32278	0.493617	0	0	0	0
	2.9	0.804776	0.804776	0.77735	0.75	0.72265	0.695224	0.667635	0.639756	0.611382	0.582093	0.550701	0.449299	0.417907	0.388618	0.360244	0.332365	0.494338	0.249959	0	0	0

Figure 6: Enter Caption

2. Implement Gudanov Method to numerically solve the same PDE equation when the speed density relation follows the car following model, that is the traffic microscopic model. Generate the same tables as in Question no 1 for each case. This creates a bridge between traffic macroscopic and microscopic model.

```matlab
% Traffic flow model for speed-density relation follows the car following model.
clear;
clc;

rhoMax = 1;      % Maximum rho
rhoMin = 0;      % Minimum rho

h = 0.1;        % space step size
k = 0.1;        % time step size
x = -1:h:1;      % Road - (-1 to 1)
x_divs = length(x);          % No. of space steps

total_time= 3;               % Total time under consideration
t_steps = total_time / k;    % No. of time steps

% Initial conditions
rho0 = linspace(0.1, 0.5, x_divs);

u = zeros(t_steps, x_divs);      % Array to store u

% Initial and boundary conditions for u
u(1,:) = rho0;  % 1 - 2 .* rho0/rhoMax ;
u(:,1) = u(1,1)*ones(t_steps,1);

% Numerical calculation using Godunov method
for t = 1:t_steps-1
    % Apply boundary condition (2)
    % for i = 1:2:length(signal_time_idx)-1
    %    u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx) = -1;
    %    u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx + 1) = 1;
    % end

    for i = 2:x_divs - 1
        u(t+1,i) = u(t,i) - (k/h)*(f(u_star(t,i, u,@f, @df))-f(u_star(t,i-1, u,@f, @df)));
    end
end
u(:,1) = u(:,2);
u(:,end) = u(:,end-1);

% Get rho value from u
rho_tab =  u;


% Plotting at different cases
figure;
% First subplot
subplot(2, 2, 1);
plot(x, rho_tab(1,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('t = 0');
```

```
% Display grid
grid on;


% Second subplot
subplot(2, 2, 2);
plot(x, rho_tab(10,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('t1');
grid on;

% Third subplot
subplot(2, 2, 3);
plot(x, rho_tab(15,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('t2');
grid on;

% forth subplot
subplot(2, 2, 4);
plot(x, rho_tab(25,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('t3');
grid on;

% Function for u_star calculation used in Godunov scheme
function u_star_out = u_star(n, i, u, f, df)
    gradF = df(u(n, i));         % Gradient of flux for given u(n,i)
    gradF1 = df(u(n, i + 1));    % Gradient of flux for given u(n,i+1)
    % Apply different conditions...
    if gradF >= 0 && gradF1 >= 0
        u_star_out = u(n, i);
    elseif gradF < 0 && gradF1 < 0
        u_star_out = u(n, i + 1);
    elseif gradF >= 0 && gradF1 < 0
        s = (f(u(n, i + 1))-f(u(n, i)))/(u(n, i + 1)- u(n, i)) ;
        if s >= 0
            u_star_out = u(n, i);
        else
            u_star_out = u(n, i + 1);
        end
    else
        u_star_out = 0;
    end
end

% Flux function for car following model
function result = f(rho)
    rhoCrit = 0.45; % Critical rho
    rhoJam = 1;     % Jam rho
    vMax = 0.95;    % Max velocity

    if rho <= rhoCrit
```

```
            result = rho * vMax;
        elseif rhoCrit < rho && rho < rhoJam
            result = vMax * (1 - rho / rhoJam);
        elseif rho >= rhoJam
            result = 0;
        end
end

% Derivative of flux
function result = df(rho)
    rhoCrit = 0.45; % Critical rho
    rhoJam = 1;     % Jam rho
    vMax = 0.95;    % Max velocity

    if rho <= rhoCrit
        result = vMax;
    elseif rhoCrit < rho && rho < rhoJam
        result = -vMax / rhoJam;
    elseif rho >= rhoJam
        result = 0;
    end
end
```

Output:

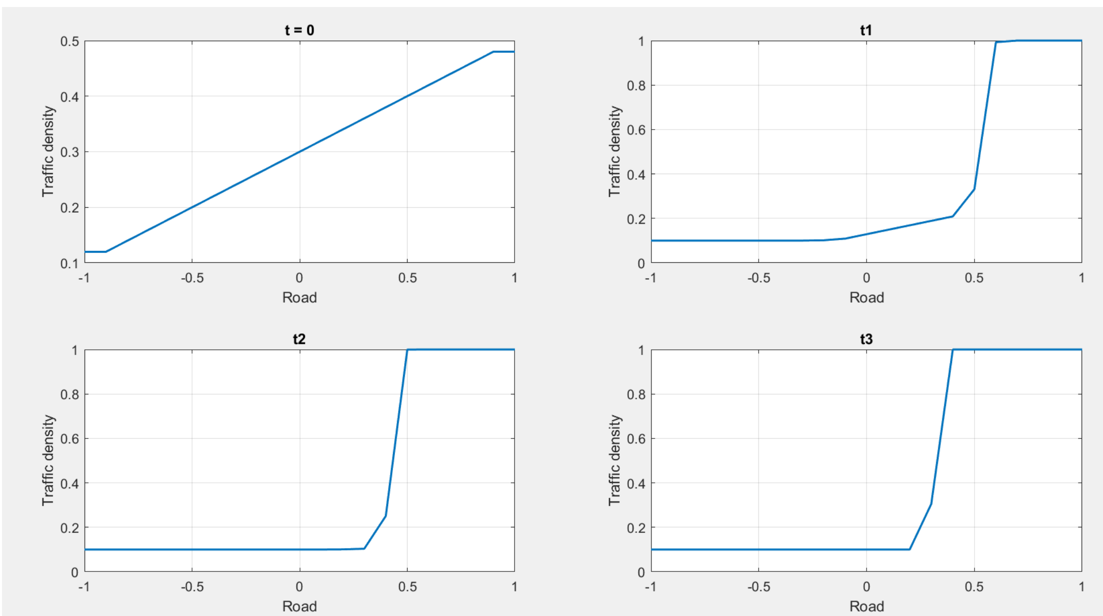


Figure 7: Density plots for different t for car following model

Car following model with signal

```
% Traffic flow model for speed-density relation follows the car following model with signal
clear;
clc;

rhoMax = 1;     % Maximum rho
rhoMin = 0;     % Minimum rho

h = 0.1;        % space step size
k = 0.1;        % time step size
x = -1:h:1;     % Road - (-1 to 1)
x_divs = length(x);         % No. of space steps

total_time= 3;              % Total time under consideration
t_steps = total_time / k;   % No. of time steps

% Initial conditions
rho0 = linspace(0.1, 0.5, x_divs);

u = zeros(t_steps, x_divs);     % Array to store u

% Initial and boundary conditions for u
u(1,:) = rho0;
u(:,1) = u(1,1)*ones(t_steps,1);

% Signal
x_signal = 0.2;     % Position of signal
x_signal_idx = floor((x_signal - x(1))/h);  % Index of signal position in x array

signal_times = [0.5, 1.5, 2.5];             % Signal timings (in min.)
signal_time_idx = signal_times/k;           % Index of signal time in time array

% Apply boundary condition (1)
for i = 1:2:length(signal_time_idx)-1
   u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx) = -1;
   u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx + 1) = 1;
end

% Numerical calculation using Godunov method
for t = 1:t_steps-1
    % Apply boundary condition (2)
    for i = 1:2:length(signal_time_idx)-1
       u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx) = -1;
       u(signal_time_idx(i):signal_time_idx(i+1), x_signal_idx + 1) = 1;
    end

    for i = 2:x_divs - 1
        u(t+1,i) = u(t,i) - (k/h)*(f(u_star(t,i, u,@f, @df))-f(u_star(t,i-1, u,@f, @df)));
    end
end
u(:,1) = u(:,2);
u(:,end) = u(:,end-1);

% Get rho value from u
rho_tab =  u;
```

```
% Plotting at different cases
figure;
% First subplot - when green light (initial cond.)
subplot(2, 2, 1);
plot(x, rho_tab(1,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('t = 0, Uniform traffic flow (Green signal)');
% Display grid
grid on;
hold on;
% Add a vertical line at signal position
xline(x_signal, 'g--', 'LineWidth', 2); % Signal position
hold off;

% Second subplot - when red light
subplot(2, 2, 2);
plot(x, rho_tab(10,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('During Red signal');
grid on;
hold on;
% Add a vertical line at signal position
xline(x_signal, 'r--', 'LineWidth', 2);
hold off;

% Third subplot - when red light (just before green light)
subplot(2, 2, 3);
plot(x, rho_tab(15,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('Just before Green signal');
grid on;
hold on;
% Add a vertical line at signal position
xline(x_signal, 'r--', 'LineWidth', 2);
hold off;

% forth subplot
subplot(2, 2, 4);
plot(x, rho_tab(25,:), 'LineWidth', 1.5);
% Add labels and title
xlabel('Road');
ylabel('Traffic density');
title('Traffic flow 1 min after signal turned green');
grid on;
hold on;
% Add a vertical line at signal position
xline(x_signal, 'g--', 'LineWidth', 2);
hold off;
```

```matlab
% Function for u_star calculation used in Godunov scheme
function u_star_out = u_star(n, i, u, f, df)
    gradF = df(u(n, i));         % Gradient of flux for given u(n,i)
    gradF1 = df(u(n, i + 1));    % Gradient of flux for given u(n,i+1)
    % Apply different conditions...
    if gradF >= 0 && gradF1 >= 0
        u_star_out = u(n, i);
    elseif gradF < 0 && gradF1 < 0
        u_star_out = u(n, i + 1);
    elseif gradF >= 0 && gradF1 < 0
        s = (f(u(n, i + 1))-f(u(n, i)))/(u(n, i + 1)- u(n, i)) ;
        if s >= 0
            u_star_out = u(n, i);
        else
            u_star_out = u(n, i + 1);
        end
    else
        u_star_out = 0;
    end
end

% Flux function for car following model
function result = f(rho)
    rhoCrit = 0.45; % Critical rho
    rhoJam = 1;     % Jam rho
    vMax = 0.95;    % Max velocity

    if rho <= rhoCrit
        result = rho * vMax;
    elseif rhoCrit < rho && rho < rhoJam
        result = vMax * (1 - rho / rhoJam);
    elseif rho >= rhoJam
        result = 0;
    end
end

% Derivative of flux
function result = df(rho)
    rhoCrit = 0.45; % Critical rho
    rhoJam = 1;     % Jam rho
    vMax = 0.95;    % Max velocity

    if rho <= rhoCrit
        result = vMax;
    elseif rhoCrit < rho && rho < rhoJam
        result = -vMax / rhoJam;
    elseif rho >= rhoJam
        result = 0;
    end
end
```

Output:

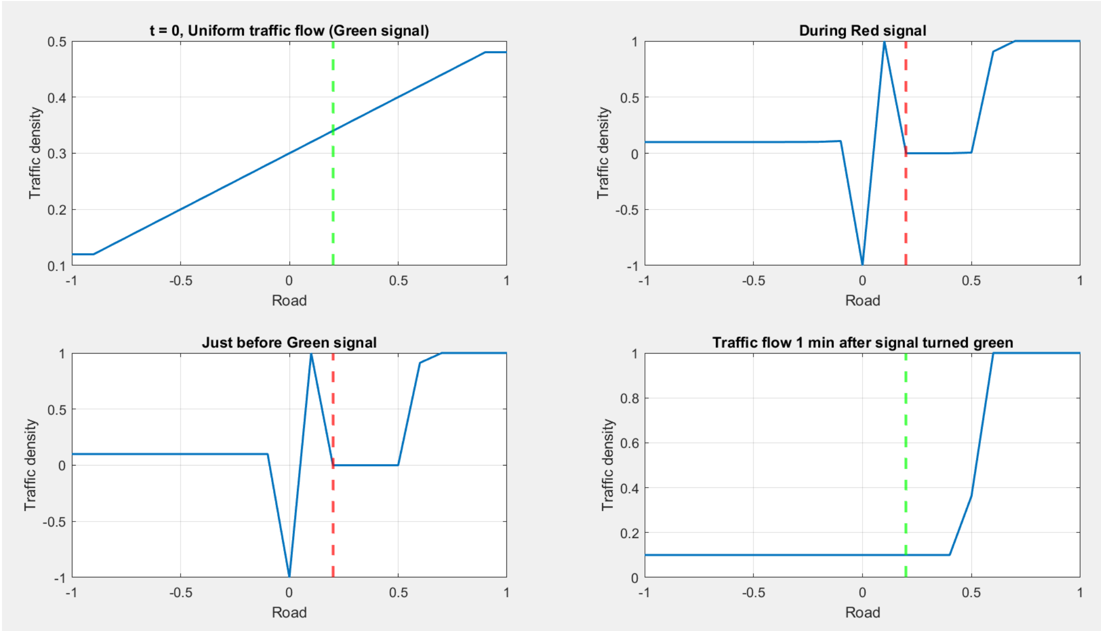


Figure 8: Density plots for different t for car following model with traffic signal

3. How will you formulate the traffic flow modelling problem as a black box model?

To formulate the traffic flow model to a black box model, we focus on observing the inputs and collecting the outputs of the system instead of looking at the complex formulations. Some of the input variables for the black box model are:

(a) Traffic Density - vehicles per unit length
(b) Traffic Speed - avg. speed of vehicle on the road
(c) Traffic Flow - no. of vehicles passing through a point per unit time.
(d) Traffic Composition - types of vehicles on the road.
(e) External variables: Road type, Time of day, Weather, etc.

We collect data for these inputs and train the black-box model to predict the traffic flow of the road and conditions like traffic jams etc.

****