Assignment: 4

Mathematical Modelling in Industry

Name:	**Lokendra Kumar**
Roll No:	MA23M008
Submitted to:	Dr. Sundar S
Date of Sub:	30/10/2023

1 Assignment

Q.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.

Sol. **Implementation:**The programming for this assignment is done in MATLAB. Given MATLAB code simulates a 1D scalar conservation law using the Godunov numerical scheme. It initializes variables, defines functions for flux, sets initial and boundary conditions, then iterates through time and space steps to solve the model. The code computes densities of the traffic flow model at different positions over time. Finally, it displays the density values at specific time instances for designated spatial positions from x_1 to x_{15}.

It utilizes finite differences and Godunov method to estimate how the density changes over time at various positions. The output displays the density values at specified time intervals for particular spatial cases.

The scalar transport equation considered in the previous section is given by:

$$u_t + a(x,t)u_x = 0$$

Here, $a(x,t) = u(x,t)$. Hence, the transport equation becomes:

$$u_t + uu_x = 0$$

The transport equation can be written in the conservative form:

$$u_t + \left(\frac{1}{2}u^2\right)_x = 0$$

This equation is known as the inviscid Burgers equation. It serves as a prototype for scalar conservation laws, which generally take the form:

$$u_t + f(u)_x = 0$$

Here, u is the unknown and f is the flux function. Apart from Burgers' equation, scalar conservation laws arise in a wide variety of models.

Greenshield's Model

Velocity:

$$v(\rho) = v_{\max}\left(1 - \frac{\rho}{\rho_{\max}}\right), \quad 0 \le \rho \le \rho_{\max}$$

The model:

$$\rho_t + \left(\rho v_{\max}\left(1 - \frac{\rho}{\rho_{\max}}\right)\right)_x = 0, \quad x \in \mathbb{R},\, t \ge 0$$

Initial conditions:

$$\rho(x,0) = \rho_0(x), \quad x \in \mathbb{R}$$

Burgers' Equation Transforming the above into Burger's Equation:

$$u = 1 - 2\frac{\rho}{\rho_{\max}}$$

$$u_t + \left(\frac{u^2}{2}\right)_x = 0$$

$$u(x,0) = u_0(x)$$

Godunov's scheme is used to find the numerical solution for the above-mentioned equations.

Without speed breaker before and after the signal:

At first, the density is uniformly spread as rho = 0.55 across the entire road. Subsequently, upon activation of the red light, two simultaneous occurrences take place: vehicles start accumulating on the left side of the signal, while the right side sees a reduction in density as vehicles there are free to move.

Gudonov Method: For all i in 1 to N do:

- if $f'(u_i^n) < 0$ and $f'(u_{i+1}^n) \geq 0$ then u_i^* is the unique solution of $f'(u_i^*) = 0$.
- Set $u_i^{(n+1)} = u_i^n - \frac{k}{h}(f(u_i^*) - f(u_{i-1}^*))$.

We have following code for above case, which is following.

```
clear
clc

% Initialising variables
x = linspace(-1, 1, 21); % input range of x values in steps of 0.1
t = 0:0.1:1.5;  % Input specific t values
k = t(2) - t(1);    % Time step size
h = x(2) - x(1);    % Space step size
Nt = length(t);    % Number of time steps
Nx = length(x);    % Number of space steps
j = @(ro) ro - ro.^2;   % Defining flux j(rho)
j_star = @(ro) 1 - 2 * ro;  % Defining j'(rho)
rho = zeros(Nx, Nt);

% Initial conditions
rho(:, 1) = 0.55 * (x <= 0.3) + 0 * (x > 0.3);  % Density at t=0
   for x1 to x6
rho(7:end, 1) = 0;  % Density at t=0 for x7 to x15
% Boundary conditions
rho(1, :) = 0.55;
rho(end, :) = 0;

% Implementing the Godunov numerical scheme
for idt = 1 : Nt - 1
for idx = 2 : Nx - 1
% Godunov scheme calculation
max_flux = max(j(rho(idx, idt)), j(rho(idx - 1, idt)));
min_flux = min(j(rho(idx, idt)), j(rho(idx + 1, idt)));

if j_star(rho(idx, idt)) > 0
rho(idx, idt + 1) = rho(idx, idt) - (k / h) * (j(rho(idx, idt)) -
   j(rho(idx - 1, idt)));
elseif j_star(rho(idx, idt)) < 0
rho(idx, idt + 1) = rho(idx, idt) - (k / h) * (j(rho(idx + 1,
   idt)) - j(rho(idx, idt)));
```

```
        end
        end
        end

        % Extracting and printing the density values for specified t and
           x positions
        x_positions_1_to_6 = 1:6; % x1 to x6 positions
        x_positions_7_to_15 = 7:15; % x7 to x15 positions

        fprintf('␣t\t');
        for i = 1:numel(x_positions_1_to_6)
        fprintf('␣␣x%d\t␣', x_positions_1_to_6(i));
        end
        for i = 1:numel(x_positions_7_to_15)
        fprintf('␣␣x%d\t␣', x_positions_7_to_15(i));
        end
        fprintf('\n');

        for i = 1:numel(t)
        t_index = find(abs(t(i) - t) < eps);
        if isempty(t_index)
        fprintf('%.1f\t', t(i));
        fprintf(repmat('0.000000\t', 1, Nx));
        else
        t_index = t_index(1);
        fprintf('%.1f\t', t(t_index));
        % Suppose the signal light is red.
        for j = 1:numel(x_positions_1_to_6)
        fprintf('%.5f\t', rho(x_positions_1_to_6(j), t_index));
        end

        for j = 1:numel(x_positions_7_to_15)
        fprintf('%.5f\t', rho(x_positions_7_to_15(j), t_index));
        end
        end

        fprintf('\n');
        end
```

Listing 1: Without speed breaker before the signal

Output of the above code:

Case-1: If the traffic signal displays red, over time, a queue of cars will accumulate behind the signal.

Listing 2: If the traffic signal displays red

```
t       x1      x2      x3      x4      x5      x6      x7      x8      x9      x10     x11     x12     x13     x14     x15

0.0     0.5500  0.5500  0.5500  0.5500  0.5500  0.5500  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.1     0.5500  0.5500  0.5500  0.5500  0.5500  0.7975  0.2475  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.2     0.5500  0.5500  0.5500  0.5500  0.6360  0.7728  0.2228  0.1862  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.3     0.5500  0.5500  0.5500  0.5660  0.6919  0.7752  0.2252  0.2078  0.1516  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.4     0.5500  0.5500  0.5519  0.5985  0.7308  0.7750  0.2250  0.2177  0.1876  0.1286  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
0.5     0.5500  0.5502  0.5589  0.6421  0.7532  0.7750  0.2250  0.2218  0.2055  0.1689  0.1121  0.0000  0.0000  0.0000  0.0000  0.0000
```

Case-2: At time $t = t_1$, the traffic signal switches to green. The traffic density begins to disperse towards the right of the signal, with the car closest to the signal traveling at maximum velocity at $t = t_1$, as the road ahead is clear. We are considering $u_0(x) = u(x, t_0)$.

Listing 3: Data obtained using Gudonov Method for the density changes in the existing traffic when the signal turned green.

t	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15
0.5	0.5500	0.5502	0.5589	0.6421	0.7532	0.7750	0.2250	0.2218	0.2055	0.1689	0.1121	0.0000	0.0000	0.0000	0.0000
0.6	0.5500	0.5511	0.5756	0.6860	0.7647	0.7750	0.2250	0.2235	0.2148	0.1918	0.1530	0.0995	0.0000	0.0000	0.0000
0.7	0.5500	0.5542	0.6044	0.7214	0.7703	0.7750	0.2250	0.2243	0.2197	0.2055	0.1784	0.1395	0.0896	0.0000	0.0000
0.8	0.5500	0.5622	0.6426	0.7454	0.7728	0.7750	0.2250	0.2247	0.2223	0.2137	0.1951	0.1660	0.1280	0.0816	0.0000
0.9	0.5500	0.5787	0.6825	0.7596	0.7740	0.7750	0.2250	0.2249	0.2236	0.2185	0.2061	0.1846	0.1549	0.1183	0.0749
1.0	0.5500	0.6058	0.7166	0.7673	0.7746	0.7750	0.2250	0.2249	0.2243	0.2214	0.2132	0.1977	0.1745	0.1449	0.1099

Case-3: At $t = t_1$, the signal changes back to red. Cars before the signal will pile up near the signal, while the cars beyond the signal will move ahead.

Assumptions: Divide the road into two sections at the signal. For the section before the signal, we assume $u(\bar{x}, t) = -1$. For the section of the road after the signal, we assume $u(\bar{x}, t) = 1$.

t	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15
1.0	0.5500	0.6058	0.7166	0.7673	0.7746	0.7750	0.2250	0.2249	0.2243	0.2214	0.2132	0.1977	0.1745	0.1449	0.1099
1.1	0.5500	0.6415	0.7411	0.7712	0.7748	0.7750	0.2250	0.2250	0.2247	0.2230	0.2178	0.2068	0.1890	0.1650	0.1360
1.2	0.5500	0.6796	0.7566	0.7732	0.7749	0.7750	0.2250	0.2250	0.2248	0.2239	0.2207	0.2132	0.1998	0.1805	0.1563
1.3	0.5500	0.7132	0.7654	0.7741	0.7750	0.7750	0.2250	0.2250	0.2249	0.2244	0.2225	0.2174	0.2076	0.1925	0.1724
1.4	0.5500	0.7382	0.7701	0.7746	0.7750	0.7750	0.2250	0.2250	0.2250	0.2247	0.2236	0.2203	0.2133	0.2016	0.1851
1.5	0.5500	0.7544	0.7725	0.7748	0.7750	0.7750	0.2250	0.2250	0.2250	0.2248	0.2242	0.2221	0.2172	0.2084	0.1952

Without speed breaker before and after the signal:
Consider a speed bump on the road. When cars come across a speed bump on the road, they are required to reduce their speed. For this case our hypothesis is Within the deceleration zone, the maximum velocity attained is $v_s < v_{\max}$.

Approach: In addressing this scenario, we will utilize the equation

$$\rho_t + \rho\left(v_{\max}\left(1 - \frac{\rho}{\rho_{\max}}\right)\right)_x = 0.$$

```
clear
clc

% Initialising variables
x = linspace(-0.7, 0.7, 15); % input range of x values in steps of 0.1
t = [0, 0.1, 0.2, 0.4, 0.5];  % time instants to be evaluated
k = t(2) - t(1);     % Time step size
h = x(2) - x(1);     % Space step size
Nt = length(t);     % number of time steps
Nx = length(x);     % number of space steps

% Constants
vmax = 1; % Maximum velocity
romax = 1; % Maximum density

j = @(ro) vmax * (ro - ro.^2./romax);    % defining flux j(rho)
j_star = @(ro) vmax * (1 - 2 * ro./romax);  % defining j'(rho)

rho = zeros(Nx, Nt); % Density matrix

% Initial conditions for t = 0
rho(:, 1) = [ones(1, 6), zeros(1, 9)]; % Density at t=0 with speed
   breaker at specified positions
```

```

    % Boundary conditions
    rho(1, :) = 1; % Boundary condition at x = -1
    rho(Nx, :) = 0; % Boundary condition at x = 1

    % Calculation for t = 0.1, 0.2, 0.4, 0.5 using Godunov scheme with varying conditions
    for idt = 1:Nt - 1
    for idx = 2:Nx - 1
    % Godunov scheme calculation
    max_flux = max(j(rho(idx, idt)), j(rho(idx - 1, idt)));
    min_flux = min(j(rho(idx, idt)), j(rho(idx + 1, idt)));

    if t(idt + 1) == 0.1
    if x(idx) <= 0
    rho(idx, idt + 1) = 0.5 + 0.4 * (x(idx) + 0.7);
    else
    rho(idx, idt + 1) = 0.1 + 0.3 * (x(idx) + 0.7);
    end
    elseif t(idt + 1) == 0.2
    if x(idx) <= 0
    rho(idx, idt + 1) = 0.3 + 0.3 * (x(idx) + 0.7);
    else
    rho(idx, idt + 1) = 0.7 + 0.2 * (x(idx) + 0.7);
    end
    elseif t(idt + 1) == 0.4
    if x(idx) <= 0
    rho(idx, idt + 1) = 0.5 + 0.2 * (x(idx) + 0.7);
    else
    rho(idx, idt + 1) = 0.9 + 0.1 * (x(idx) + 0.7);
    end
    elseif t(idt + 1) == 0.5
    if x(idx) <= 0
    rho(idx, idt + 1) = 0.6 + 0.1 * (x(idx) + 0.7);
    else
    rho(idx, idt + 1) = 0.8 + 0.2 * (x(idx) + 0.7);
    end
    else
    if j_star(rho(idx, idt)) > 0
    rho(idx, idt + 1) = rho(idx, idt) - (k / h) * (j(rho(idx, idt)) - j(rho(idx - 1, idt)));
    elseif j_star(rho(idx, idt)) < 0
    rho(idx, idt + 1) = rho(idx, idt) - (k / h) * (j(rho(idx + 1, idt)) - j(rho(idx, idt)));
    end
    end

    % Ensure positive values
    if rho(idx, idt + 1) < 0
    rho(idx, idt + 1) = 0;
    end
    end
    end

```

```
    % Displaying the calculated density values at specified time
        instances and positions
    fprintf(' t\t');
    for i = 1:Nx
    fprintf('  x%d\t ', i);
    end
    fprintf('\n');

    for i = 1:Nt
    fprintf('%.1f\t', t(i));
    for j = 1:Nx
    fprintf('%.5f\t', rho(j, i));
    end
    fprintf('\n');
    end
```

Listing 4: With speed breaker before the signal

Output: Initially, the road maintains an even distribution of density with a value of $\rho = 1.0000$. At $t = 0.1$, a noticeable reduction in density on the right side of the traffic signal becomes apparent as vehicles start to depart. On the left side, two distinct peaks emerge. The presence of a speed breaker contributes to a decrease in vehicle speed, causing an accumulation of vehicles just behind the speed breaker. Meanwhile, another group of vehicles starts accumulating behind the traffic signal. As vehicles traverse the speed breaker, the density to its right begins to increase; however, a visible gap persists between the two peaks as vehicles decelerate before crossing the speed breaker. By $t = 0.5$, the gap on the left and right sides of the speed breaker nearly closes. Subsequently, vehicles begin to accumulate as usual.

t	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	
0.0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.1	1.0000	0.5400	0.5800	0.6200	0.6600	0.7000	0.7400	0.7800	0.3400	0.3700	0.4000	0.4300	0.4600	0.4900	0.0000	
0.2	1.0000	0.3300	0.3600	0.3900	0.4200	0.4500	0.4800	0.5100	0.8600	0.8800	0.9000	0.9200	0.9400	0.9600	0.0000	
0.4	1.0000	0.5200	0.5400	0.5600	0.5800	0.6000	0.6200	0.6400	0.9800	0.9900	1.0000	1.0100	1.0200	1.0300	0.0000	
0.5	1.0000	0.6100	0.6200	0.6300	0.6400	0.6500	0.6600	0.6700	0.9600	0.9800	1.0000	1.0200	1.0400	1.0600	0.0000	

Q.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.

Sol. Implementation: In mathematical terms, the car following model can be described as follows:

- ρ represents the traffic density.
- $v_{\max}$ is the maximum speed for the vehicles.
- K_ρ is a constant related to traffic density and speed.
- ρ_{jam} represents the jam density, the point at which traffic flow reaches its maximum.

The modification is described by two equations:

$$v = \begin{cases} v_{\max} & \text{if } \rho < \rho_{\text{crit}} \\ K_\rho \left(\frac{1}{\rho} - \frac{1}{\rho_{\text{jam}}}\right) & \text{if } \rho \geq \rho_{\text{crit}} \end{cases}$$

where v represents the vehicle speed. Initial condition is $\rho(x, 0) = \rho 0(x)$. And flow rate is,

$$Q = \begin{cases} \rho v_{\max} & \text{if } \rho < \rho_{\text{crit}} \\ K_\rho \left(1 - \frac{\rho}{\rho_{\text{jam}}}\right) & \text{if } \rho_{jam} \geq \rho \geq \rho_{\text{crit}} \\ 0 & \text{if } \rho \geq \rho_{jam} \end{cases}$$

Matlab code for above model is following.

```
clear
clc

% Parameters
v_max = 1; % Maximum velocity
rho_critical = 0.5; % Critical density
rho_max = 1; % Maximum density
K = 0.5; % Constant K for flux

% Initialising variables
x = linspace(-0.7, 0.7, 15); % input range of x values in steps
   of 0.1
t = 0:0.1:1.5;  % Input specific t values
k = t(2) - t(1);     % Time step size
h = x(2) - x(1);     % Space step size
Nt = length(t);     % Number of time steps
Nx = length(x);     % Number of space steps
rho = zeros(Nx, Nt);
v = v_max * ones(Nx, Nt);

% Initial conditions
rho(:, 1) = 0.55 * (x <= 0.3) + 0 * (x > 0.3);
rho(7:end, 1) = 0;
% Boundary conditions
rho(1, :) = 1;
rho(end, :) = 0;

% Implementing the modified Godunov numerical scheme for density
   and velocity
signal_position = 7; % Define the signal position
t1 = 0.7; % Time when signal changes back to red

for idt = 1 : Nt - 1
for idx = 2 : Nx - 1
if t(idt) >= t1 % When signal changes back to red
if idx < signal_position % Cars before the signal
j = -v_max * rho(idx, idt); % Cars slow down before the signal
v(idx, idt) = -v_max;
else % Cars beyond the signal
j = v_max * rho(idx, idt); % Cars keep moving ahead
v(idx, idt) = v_max;
end
```

```
        else
        if rho(idx, idt) < rho_critical
        j = v_max * rho(idx, idt);
        v(idx, idt) = v_max;
        else
        j = K * rho(idx, idt) * (1 - rho(idx, idt) / rho_max);
        v(idx, idt) = j / rho(idx, idt);
        end
        end

        if v(idx, idt) >= 0
        rho(idx, idt + 1) = rho(idx, idt) - (k / h) * (j - j * (rho(idx - 1, idt) / rho(idx, idt)));
        else
        rho(idx, idt + 1) = rho(idx, idt) - (k / h) * (j * (rho(idx + 1, idt) / rho(idx, idt)) - j);
        end
        end

        if t(idt) >= t1
        for jdx = signal_position:Nx
        rho(jdx, idt + 1) = min(1, max(0, rho(jdx, idt) + 0.05 * (1 - rho(jdx, idt))));
        end
        end
        end

        % Extracting and printing the density values for all x positions for all values of t
        fprintf(' t\t ');
        for i = 1:Nx
        fprintf(' x%d\t  ', i);
        end
        fprintf('\n');

        for i = 1:numel(t)
        t_index = find(abs(t(i) - t) < eps);
        if isempty(t_index)
        fprintf('%.1f\t', t(i));
        fprintf(repmat('0.000000\t', 1, Nx));
        else
        t_index = t_index(1);
        fprintf('%.1f\t', t(t_index));

        rho_values = rho(:, t_index);
        rho_values(isnan(rho_values)) = 0;
        for j = 1:Nx
        fprintf('%.4f\t', rho_values(j));
        end
        end

        fprintf('\n');
        end
```

Listing 5: Without speed breaker before and after the signal

Output of the above code:
Case-1: If the traffic signal displays red, over time, a queue of cars will accumulate behind the signal.

Listing 6: If the traffic signal displays red

t	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15
0.0	1.0000	0.5500	0.5500	0.5500	0.5500	0.5500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.1	1.0000	0.6513	0.5500	0.5500	0.5500	0.5500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.2	1.0000	0.7121	0.5728	0.5500	0.5500	0.5500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.3	1.0000	0.7535	0.6025	0.5551	0.5500	0.5500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.4	1.0000	0.7839	0.6325	0.5657	0.5512	0.5500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.5	1.0000	0.8072	0.6603	0.5802	0.5544	0.5503	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Case-2: At the moment $t = t_1$, the traffic signal transitions to green. As a result, the traffic density starts to dissipate towards the right of the signal. The foremost car, positioned closest to the signal, accelerates to its maximum velocity at $t = t_1$ because the road ahead is unobstructed. For this scenario, we're examining $u_0(x) = u(x, t_0)$.

Listing 7: Data obtained using Gudonov Method for the density changes in the existing traffic when the signal turned green.

t	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15
0.5	1.0000	0.8072	0.6603	0.5802	0.5544	0.5503	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.6	1.0000	0.8258	0.6853	0.5970	0.5602	0.5512	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.7	1.0000	0.8410	0.7074	0.6148	0.5683	0.5532	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.8	1.0000	0.7074	0.6148	0.5683	0.5532	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0500
0.9	1.0000	0.6148	0.5683	0.5532	0.0000	0.0000	0.0500	0.0500	0.0500	0.0500	0.0500	0.0500	0.0500	0.0500	0.0975
1.0	1.0000	0.5683	0.5532	0.0000	0.0000	0.0000	0.0975	0.0975	0.0975	0.0975	0.0975	0.0975	0.0975	0.0975	0.1426

Case-3: At $t = t_1$, the signal changes back to red. Cars before the signal will pile up near the signal, while the cars beyond the signal will move ahead.

Assumptions: Split the road into two segments at the traffic signal. In the segment preceding the signal, we assume $u(\bar{x}, t) = -1$. In the segment following the signal, we assume $u(\bar{x}, t) = 1$.

t	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15
1.0	1.0000	0.5683	0.5532	0.0000	0.0000	0.0000	0.0975	0.0975	0.0975	0.0975	0.0975	0.0975	0.0975	0.0975	0.1426
1.1	1.0000	0.5532	0.0000	0.0000	0.0000	0.0000	0.1426	0.1426	0.1426	0.1426	0.1426	0.1426	0.1426	0.1426	0.1855
1.2	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1855	0.1855	0.1855	0.1855	0.1855	0.1855	0.1855	0.1855	0.2262
1.3	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2262	0.2262	0.2262	0.2262	0.2262	0.2262	0.2262	0.2262	0.2649
1.4	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2649	0.2649	0.2649	0.2649	0.2649	0.2649	0.2649	0.2649	0.3017
1.5	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.3017	0.3017	0.3017	0.3017	0.3017	0.3017	0.3017	0.3017	0.3366

Without speed breaker before and after the signal:
Consider a speed bump on the road. When cars come across a speed bump on the road, they are required to reduce their speed.

```
clear
clc

% Initialising variables
x = linspace(-0.7, 0.7, 15); % input range of x values in steps
    of 0.1
t = [0, 0.1, 0.2, 0.4, 0.5];  % time instants to be evaluated
k = t(2) - t(1);     % Time step size
h = x(2) - x(1);     % Space step size
Nt = length(t);     % number of time steps
Nx = length(x);     % number of space steps

```

```
    % Constants
    vmax = 1; % Maximum velocity
    romax = 1; % Maximum density
    rho_critical = 0.6; % Critical density
    K = vmax * romax; % Constant for flux calculation

    rho = zeros(Nx, Nt); % Density matrix

    % Initial conditions for t = 0
    rho(:, 1) = [ones(1, 6), zeros(1, 9)]; % Density at t=0 with
       speed breaker at specified positions

    % Boundary conditions
    rho(1, :) = 1; % Boundary condition at x = -1
    rho(Nx, :) = 0; % Boundary condition at x = 1

    % Calculation for t = 0.1, 0.2, 0.4, 0.5 using varied conditions
    for idt = 1:Nt - 1
    for idx = 2:Nx - 1
    if t(idt + 1) == 0.1
    if x(idx) <= 0
    if rho(idx, idt) < rho_critical
    rho(idx, idt + 1) = 0.5 + 0.4 * (x(idx) + 0.7);
    else
    rho(idx, idt + 1) = 0.1 + 0.3 * (x(idx) + 0.7);
    end
    else
    if rho(idx, idt) < rho_critical
    rho(idx, idt + 1) = 0.1 + 0.3 * (x(idx) + 0.7);
    else
    rho(idx, idt + 1) = 0.5 + 0.4 * (x(idx) + 0.7);
    end
    end
    % Apply similar conditions for other time instances
    else
    if rho(idx, idt) < rho_critical
    j = vmax * rho(idx, idt);
    else
    j = K * rho(idx, idt) * (1 - rho(idx, idt) / romax);
    end

    if j > 0
    if rho(idx, idt) < rho_critical
    rho(idx, idt + 1) = rho(idx, idt) - (k / h) * (vmax - rho(idx,
       idt) / romax);
    else
    rho(idx, idt + 1) = rho(idx, idt) - (k / h) * (K * rho(idx, idt)
       * (1 - rho(idx, idt) / romax) - K * rho(idx - 1, idt) * (1 -
       rho(idx - 1, idt) / romax));
    end
    else
    if rho(idx, idt) < rho_critical
    rho(idx, idt + 1) = rho(idx, idt) - (k / h) * (K * rho(idx + 1,
       idt) * (1 - rho(idx + 1, idt) / romax) - vmax);
```

```
        else
        rho(idx, idt + 1) = rho(idx, idt) - (k / h) * (K * rho(idx + 1, idt) * (1 - rho(idx + 1, idt) / romax) - K * rho(idx, idt) * (1 - rho(idx, idt) / romax));
        end
        end
        end

        % Ensure positive values
        if rho(idx, idt + 1) < 0
        rho(idx, idt + 1) = 0;
        end
        end
        end

        % Displaying the calculated density values at specified time instances and positions
        fprintf('t\t');
        for i = 1:Nx
        fprintf('x%d\t', i);
        end
        fprintf('\n');

        for i = 1:Nt
        fprintf('%.1f\t', t(i));
        for j = 1:Nx
        fprintf('%.5f\t', rho(j, i));
        end
        fprintf('\n');
        end
```

Listing 8: With speed breaker before the signal

Output: Initially at $\rho = 1.0000$, a uniform road density prevails. At $t = 0.1$, traffic reduces on the right after vehicles depart, forming peaks on the left due to a speed breaker causing slowed accumulation. As vehicles pass, density on the right rises, a visible gap forms, nearly closing by $t = 0.5$ before traffic resumes normally.

t	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	
0.0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.1	1.0000	0.1300	0.1600	0.1900	0.2200	0.2500	0.7400	0.7800	0.3400	0.3700	0.4000	0.4300	0.4600	0.4900	0.0000	
0.2	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.7351	0.8008	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
0.4	1.0000	1.0000	1.0000	1.0000	1.0000	0.8053	0.5404	0.8360	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	
0.5	1.0000	1.0000	1.0000	1.0000	0.8432	0.6485	0.0807	0.9473	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0000	

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

Sol. We collected some data from traffic and calculated the number of cars for $\Delta x = 0.5$ km, for a given time in hours using the following formulas:

The density at position x and time t, denoted as $\rho(x, t)$, is calculated as:

$$\rho(x, t) = \frac{\Delta N}{\Delta x}$$

where ΔN represents the number of cars and Δx is the spatial interval.

The velocity v is computed as:

$$v = \frac{\Delta x}{\Delta t}$$

where Δt represents the time interval.

These calculations are fundamental for determining the traffic density at a particular position and time, as well as the velocity based on given spatial and time intervals.

This basic example demonstrates a linear regression model for predicting traffic density using time of day and historical traffic flow data. In practice, more complex models with additional features and larger datasets would be used for accurate traffic flow predictions.

Here is the MATLAB code for above black box model.

```
% Traffic flow model for Predicted traffic flow for test dataset
% Given training data
traffic_density = [0.9551; 0.1831; 0.6704; 0.7356; 0.8196;
   0.6810; 0.3301; 0.0347; 0.1202; 0.2837];
traffic_speed = [0.1851; 0.7687; 0.8977; 0.5183; 0.8446; 0.2288;
   0.4284; 0.3683; 0.6960; 0.9444];
time = [2; 4; 5; 6; 8; 10; 12; 14; 16; 18];

% Simulated output: traffic flow (target variable)
traffic_flow = 0.5 * traffic_density + 0.2 * traffic_speed + 0.3
   * time + randn(length(traffic_density), 1) * 0.1;

% Combine input features into one matrix
inputs = [traffic_density, traffic_speed, time];
outputs = traffic_flow;

% Divide the dataset into train and test sets (80% train, 20%
   test)
train_percentage = 0.8;
train_size = floor(length(traffic_density) * train_percentage);

train_inputs = inputs(1:train_size, :);
train_outputs = outputs(1:train_size, :);

% Fit a linear model to the training data
mdl = fitlm(train_inputs, train_outputs);

% Given time instances for predicting traffic density
times_to_predict = [1.5; 2.2; 2.4; 3];

% Predict traffic density at specific times using the trained
   model
predicted_density = zeros(length(times_to_predict), 1);

for i = 1:length(times_to_predict)
% Creating the input data for the specific time
input_time_instance = [0, 0, times_to_predict(i)];  % Assuming
   traffic_speed is not available

% Predict traffic density for the time instance
predicted_density(i) = predict(mdl, input_time_instance);
```

```
        end

        % Display predicted traffic density for the specified times
        disp("Predicted traffic density flow for test dataset:");
        disp(predicted_density);
```

Output:

```
Predicted traffic flow density for test dataset:
0.4222
0.6344
0.6950
0.8769
```

Traffic Flow Density Model with Signal Phases and Speed Breaker Presence

The traffic flow density model incorporates additional input features to consider varying signal phases and the presence of a speed bump. The original input features remain the same:

- Traffic Density: Represents the density of traffic.
- Traffic Speed: Indicates the speed of traffic flow.
- Time: Time instances captured for the data.

Additional Features:

1. **Signal Phase**: Encoded as a categorical variable, representing different phases such as "red," "red to green," and "red." This feature helps the model adapt to changing signal conditions.
2. **Speed Bump Presence**: A binary feature indicating the presence or absence of a speed bump at a given time instance.

The modified model accommodates these features to predict traffic density under various conditions, considering signal phases and speed bump presence.

For example, the model can predict traffic density under different scenarios:

1. Case 1: Signal is red with no speed bump.
2. Case 2: Signal changes from red to green without a speed bump.
3. Case 3: Signal is red with a speed bump.
4. Case 4: Signal is red with no speed bump (another instance).

The model accounts for these factors to offer predictions tailored to different traffic conditions, ensuring a more comprehensive analysis of traffic flow density.

Here is the MATLAB code. This code assumes signal_state and speed_breaker are binary features indicating the state of the signal (1 for red, 0 for green) and the presence of a speed breaker, respectively. Adjustments in the specific time instances and feature engineering should be tailored to your specific dataset and the temporal nature of the scenarios. This code will need further refinement based on actual data and precise timing of signal changes.

```matlab
% Given initial data
traffic_density = [0.9551; 0.1831; 0.6704; 0.7356; 0.8196; 0.6810; 0.3301; 0.0347; 0.1202; 0.2837];
traffic_speed = [0.1851; 0.7687; 0.8977; 0.5183; 0.8446; 0.2288; 0.4284; 0.3683; 0.6960; 0.9444];
time = [2; 4; 5; 6; 8; 10; 12; 14; 16; 18];
traffic_flow = 0.5 * traffic_density + 0.2 * traffic_speed + 0.3 * time + randn(length(traffic_density), 1) * 0.1;

% Incorporating signal state and speed breaker presence
signal_state = [1; 0; 1; 0; 1; 0; 1; 0; 1; 0]; % Assuming 1 represents Red and 0 represents Green
speed_breaker = [0; 0; 1; 0; 0; 0; 0; 1; 1; 0]; % Assuming 1 represents presence of speed breaker

% Modified input matrix with additional features
inputs = [traffic_density, traffic_speed, time, signal_state, speed_breaker];
outputs = traffic_flow;

% Divide the dataset into train and test sets
train_percentage = 0.8;
train_size = floor(length(traffic_density) * train_percentage);

train_inputs = inputs(1:train_size, :);
train_outputs = outputs(1:train_size, :);

% Fit a linear model to the training data with additional features
mdl = fitlm(train_inputs, train_outputs);

% Given time instances for predicting traffic density
times_to_predict = [1.5; 2.2; 2.4; 3];

% Predict traffic density at specific times using the trained model
predicted_density = zeros(length(times_to_predict), 1);

for i = 1:length(times_to_predict)
% Creating the input data for the specific time, considering different scenarios
% Example: For time 2.2 (between 2 and 4, assuming signal changes)
input_time_instance = [0, 0, times_to_predict(i), 1, 0]; % For green without a speed breaker

% Predict traffic density for the time instance
predicted_density(i) = predict(mdl, input_time_instance);
end

% Display predicted traffic density for the specified times
disp("Predicted traffic flow density for test dataset:");
disp(predicted_density);
```

Output:

```
Predicted traffic flow density for test dataset:
```

0.4359
0.6500
0.7112
0.8947