Solution of Homework #2 (CE391F) (Advanced Traffic Theory)

By: Abduallah Mohamed, EID: aam5433

To: Dr. Christian Claudel

Matlab code: https://goo.gl/LPnnbp , also it's printed

I-Legendre Fenchel transform: **We consider the Greenshields fundamental diagram, defined as follows (v and $\mathbf{k_m}$ are parameters):**

$$\psi(\mathbf{k}) = \frac{\mathbf{v}}{\mathbf{k_m}}\mathbf{k}(\mathbf{k_m} - \mathbf{k})$$

1) **Compute the Legendre-Fenchel transform associated with this fundamental diagram (Hint: the Legendre-Fenchel transform is defined on $[-\mathbf{v}, \mathbf{v}]$, so you only need to compute the expression of the transform in this range).**

Introduction[1]: The Legendre transform is a transformation from a convex differentiable function $f(\mathbf{x})$ to a function that depends on the family of tangents $\mathbf{u}=\nabla_{\mathbf{x}} f(\mathbf{x})$, The Legendre transform is useful on its own, but it is limited to convex and differentiable functions. If any of these properties fail, the transform cannot be used, but generalize the Legendre transform to this type of functions will lead to the Legendre-Fenchel transform:

$$f^*(\mathbf{u}) = \sup_x \left(\mathbf{u}^T\mathbf{x} - f(\mathbf{x})\right)$$

Solution:

$$\phi^*(k) = \sup_{k\in[-v,v]} \left(uk + \frac{v}{k_m}k(k_m - k)\right) \tag{1}$$

let

$$f(k) = uk + \frac{v}{k_m}k(k_m - k) \tag{2}$$

In order to find the supermum of equation we need to differinte it with respect to k and equal it to zero

$$f^{'}(k) = u + v - 2\frac{vk}{k_m} = 0 \tag{3}$$

Then the value of k that satisifies the supermum satisfies is

$$k = \frac{k_m(u+v)}{2v} \tag{4}$$

pluggint the value of k back into equation yields

$$\phi^*(k) = \begin{cases} u\frac{k_m(u+v)}{2v} + \frac{v}{k_m}\frac{k_m(u+v)}{2v}(k_m - \frac{k_m(u+v)}{2v}), & \text{if } -v < u < v \\ +\infty, & \text{otherwise} \end{cases} \tag{5}$$

where:

k is the density

v, k_m are parameters

II-Computational methods: We consider the Greenshields fundamental diagram previously defined, and assume that only the initial condition is defined. We further assume that the initial condition encodes the following density:

k(0,x)=0.01 for 0<x<500

k(0,x)=0.04 for 500<x<600

k(0,x)=0.025 for 600<x<950

k(0,x)=0.02 for 950<x<1000

(distances are in meters, times are in second) The Greenshields diagram has the following parameters: v=20 m/s, $k_m = 0.2\ veh/m$

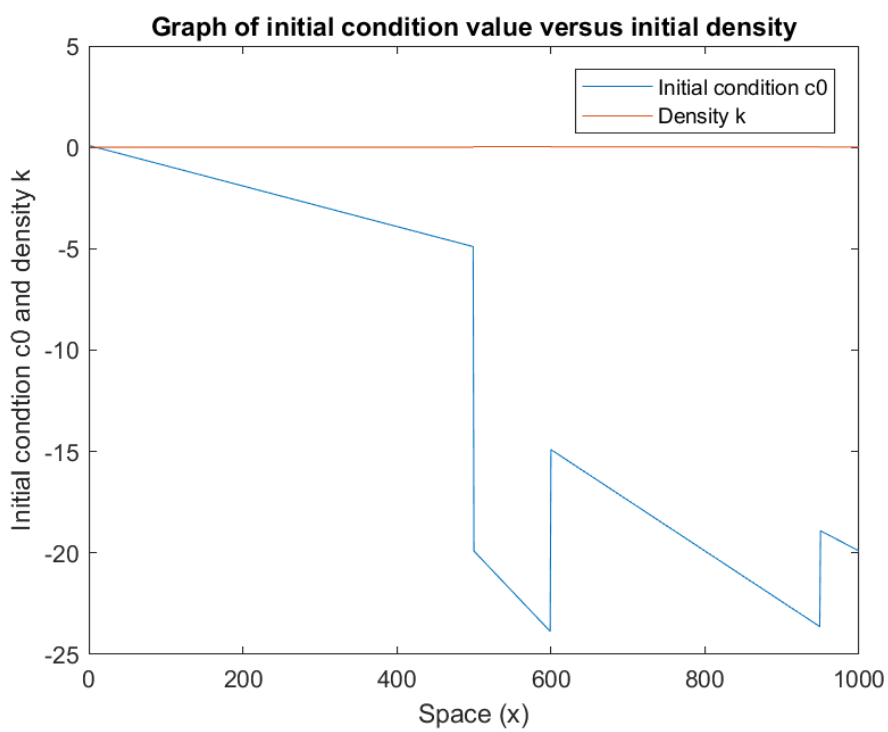
We recall that each initial condition block is defined as

$$\mathbf{c_i(t,x) = k_i\ x\ + d_i\ if\ a_i < x < b_i\ and\ t = 0}$$

$$c_i(t,x) = +\infty\ otherwise$$

1) Compute the initial density function $c(0,x)$, using the definition of the Moskowitz function.

- For code go to Matlab code `%%SECTION#1%%`

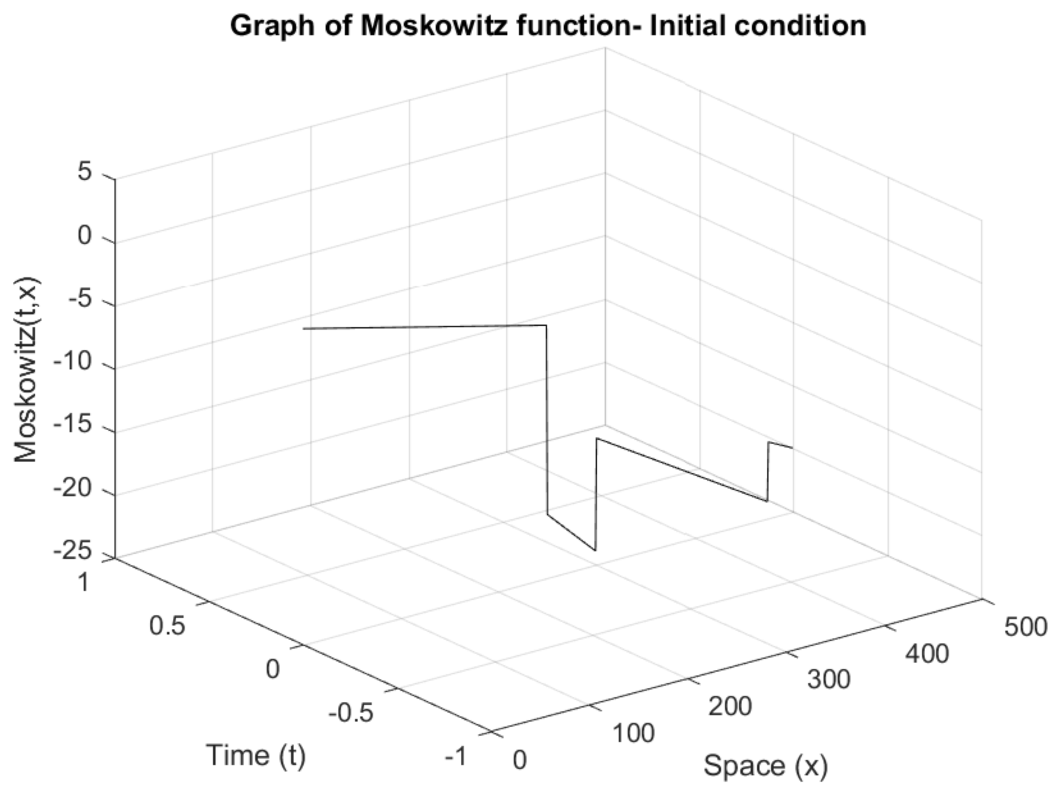


2) Recall the expression of the Lax-Hopf formula for this particular case (initial condition)

$$M_c(t,x) = \inf_{u \in Dom(\phi^*)} c(0, x+tu) + t\phi^*(u)$$

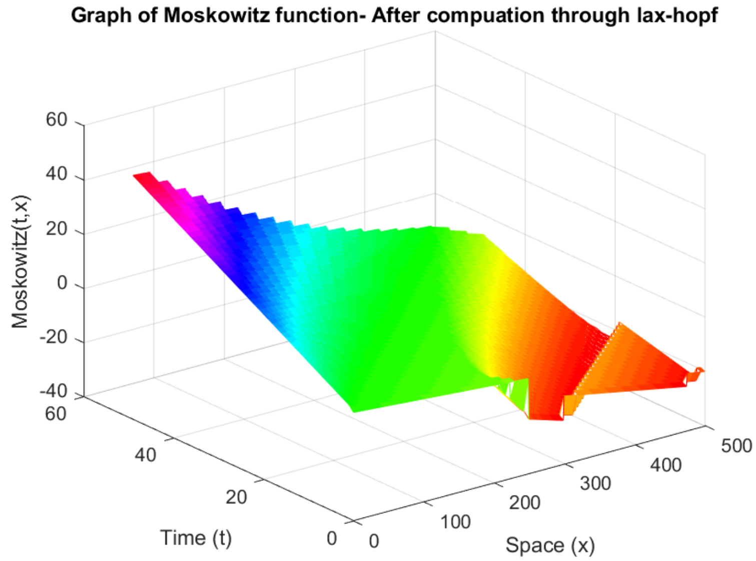
3) Write a program that stores the value of c(0,x) in the first line of a table of size 50 rows by 5000 columns (columns represent space, and rows represent time)

- For code go to Matlab code %%SECTION#3%%



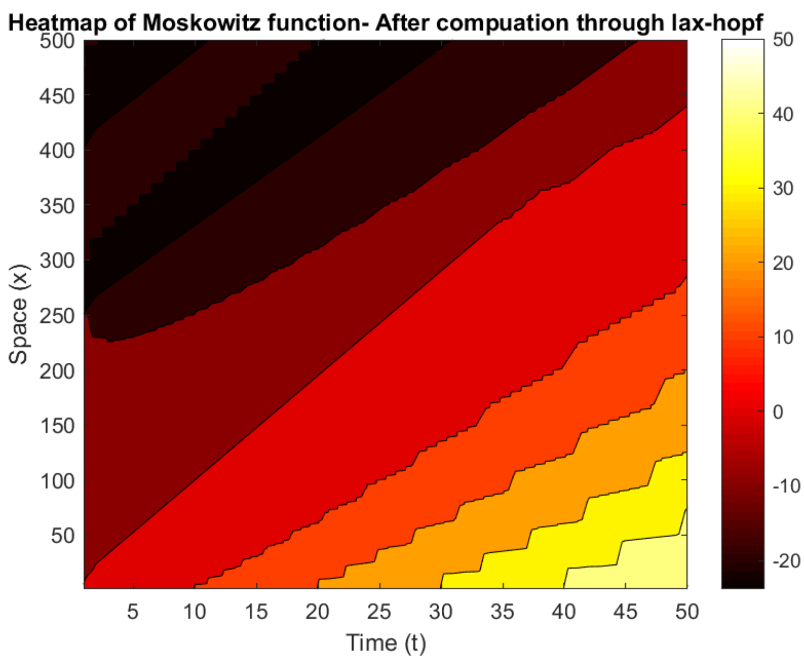
4) Now, write a program that enables the computation of each element of the table (i,j), representing the value of M(t,x), where $t = i\Delta t$ and $x = j\,\Delta x$, with $\Delta t = 1\ s$ and $\Delta x = 2\ m$

- For code go to Matlab code %%SECTION#4%%



5) Plot the corresponding table as a colormap

- For code go to Matlab code %%SECTION#5%%



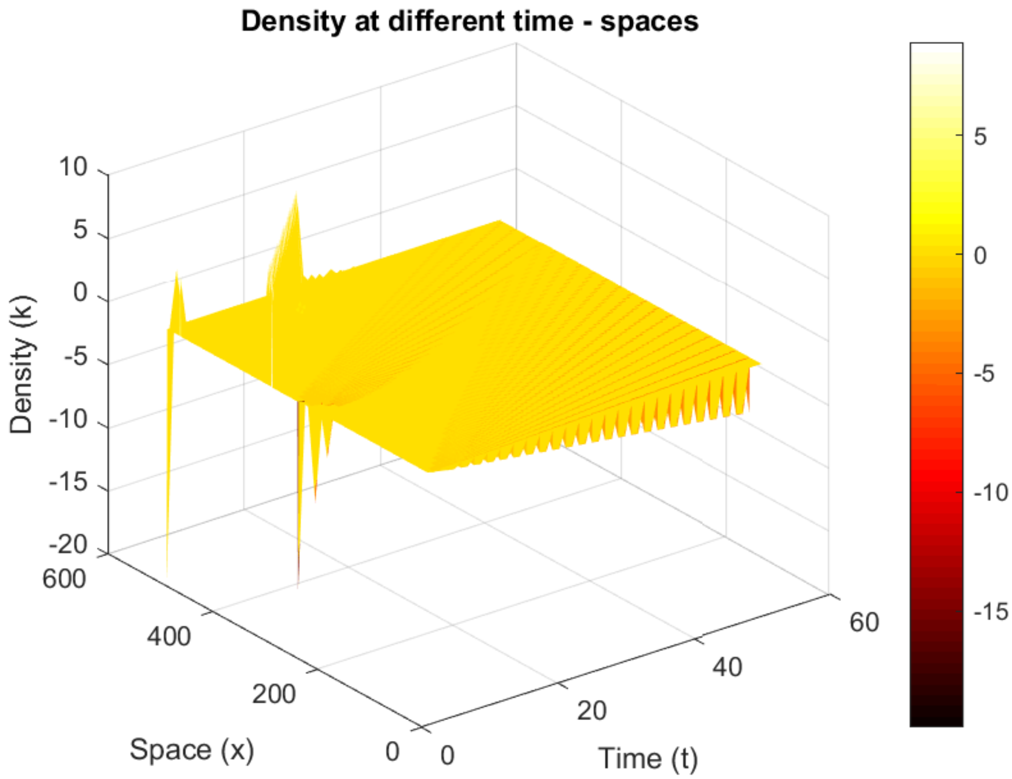
6) Now, use the relation between Moskowitz function and density to compute the approximate density at each grid point (you can can compute the approximate density at a grid point i by computing how many cars are between grid points i and $i + 1$.

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

- Where [2] ρ is the density
- For code go to Matlab code %%SECTION#6%%

7) Plot the corresponding density function

- For code go to Matlab code %%SECTION#7%%, **Also I'm not sure that this is correct**



Matlab code

```
%Author: Abduallah Mohamed
%Email: abduallah.mohamed@utexas.edu
%Description: Compuational method for Moskowitz function based on
%Greenshields fundamental Diagram

% Variables which you can change
v = 20 ; %Greenshields fundamental diagram parameter
km = 0.2 ; %Greenshields fundamental diagram parameter
di = 0.1; %Arbitrary variable because of the inetgration of a picewise
constant function
ti = 0 ; %Dessity time variable
time_dimension = 50; %Stores the time steps for Moskowitz grid
space_dimension = 500; %Stores the space steps for Moskowitz grid

%Variables which you should pay attention when changing it,
%becuase mainly they effect internal functions implementaion
initial_space_lower_limit = 1;
initial_space_upper_limit = 1000;
delta_x = 2;
delta_t = 1;

%%SECTION#1%%
%1) Compute the initial density function c(0,x), using the definition of the
Moskowitz function.
ci_0 =[]; %Vector to hold initial condition function values
ki_0 = []; %Vector to hold initial density per x

for x =initial_space_lower_limit:initial_space_upper_limit
    k = DensityMapper(x);
    ki_0(end+1) = k;
    ci_0(end +1) = InitialDenisty(DensityMapper(x),x,di,ti);
end

x = initial_space_lower_limit:initial_space_upper_limit;
figure;
plot(x,ci_0,x,ki_0);
title('Graph of initial condition value versus initial density');
xlabel('Space (x)');
ylabel('Initial condtion c0 and density k');
legend('Initial condition c0','Density k');
saveas(gcf,'DenVsInitC.png')

%%SECTION#2%%

%2) Recall the expression of the Lax-Hopf formula for this particular case
(initial condition)
%inf c (0,x+tu)+ t* lf(u)
```

```
%%SECTION#3%%

%3) Write a program that stores the value of c(0,x) in the first line of a
table of size 50
%rows by 500 columns (columns represent space, and rows represent time)

Moskowitz = inf(time_dimension,space_dimension);
for j= initial_space_lower_limit:initial_space_upper_limit
    if rem(j,delta_x) == 0 % To account for the delta x grid point
        %Because the domain of x is 1000, delta x is 2 and the grid is 500
        %point, so I made an assumption of mapping due to this ambiguity
        Moskowitz(1,j/delta_x) = ci_0(j);
    end
end

figure;
time_axis = 0:time_dimension-1;
space_axis = 0:space_dimension-1;
%surf(Moskowitz);
surf(space_axis,time_axis,Moskowitz);
colormap(hsv);
title('Graph of Moskowitz function- Initial condition');
xlabel('Space (x)');
ylabel('Time (t)');
zlabel('Moskowitz(t,x)');
saveas(gcf,'MoskowitzInitial.png')

%%SECTION#4%%

%4) Now, write a program that enables the computation of each element of the
table (i,j),
%representing the value of M(t,x), where t=i?t and x=j ?x, with ?t=1 s and
?x=2 m

for i= 2:time_dimension % we already computed at t = 0 whis is i =1
    for j= 1:space_dimension
        t= delta_t * i ;
        x= delta_x * j;
        Moskowitz(i,j) = MoskowitzFunction(t,x,ci_0,v,km);
    end
end

figure;
time_axis = 0:time_dimension-1;
space_axis = 0:space_dimension-1;
%surf(Moskowitz);
mesh(space_axis,time_axis,Moskowitz);
colormap(hsv);
title('Graph of Moskowitz function- After compuation through lax-hopf');
xlabel('Space (x)');
ylabel('Time (t)');
zlabel('Moskowitz(t,x)');
saveas(gcf,'MoskowitzAfter.png')
```

```
%%SECTION#5%%

%5)Plot the corresponding table as a colormap

MoskoReverse = zeros(space_dimension,time_dimension);%Moskowitz structre is
space * time
for i =1:time_dimension
    for j= 1:space_dimension
        MoskoReverse(j,i) = Moskowitz(i,j);
    end
end

figure;
contourf(MoskoReverse);
colormap(hot);
colorbar();
title('Heatmap of Moskowitz function- After compuation through lax-hopf');
xlabel('Time (t)');
ylabel('Space (x)');
saveas(gcf,'MoskowitzHeat.png')

%%SECTION#6%%

%6) Now, use the relation between Moskowitz function and density to
%compute the approximate density at each grid point (you can can compute
%the approximate density at a grid point i by computing how many cars are
between grid points i and i+1.

k_grid = zeros(space_dimension,time_dimension);

for j= initial_space_lower_limit:initial_space_upper_limit
    if rem(j,delta_x) == 0 % To account for the delta x grid point
        %Because the domain of x is 1000, delta x is 2 and the grid is 500
        %point, so I made an assumption of mapping due to this ambiguity
        k_grid(j/delta_x,1) = ci_0(j);
    end
end

for i= 2:space_dimension-1 % -1 cuz We don't have value at x =
space_dimension +1
    for j= 1:time_dimension
      %delta_x here is set by default , so no need to divide
      k_grid(i,j) =  ((MoskoReverse(i+1,j) - MoskoReverse(i,j))); % change in
space and time is fixed
    end
end

%%SECTION#7%%

%7) Plot the corresponding density function
```

```
figure;
h = surf(k_grid);
set(h, 'edgecolor','none');
colormap(hot);
colorbar();
title('Density at different time - spaces');
xlabel('Time (t)');
ylabel('Space (x)');
zlabel('Density (k)');
saveas(gcf,'Density.png')



%Functions

function ci = InitialDenisty(ki,x,di,ti)
%This function used to compute initial condition  from intial density
%ki = initial density, x = space value, di = arbitrary variable, ti = time
%value which is zero by default, ci = initial condition value
ti = 0;
ci = -1*ki*x+di;
end

function k_0_x = DensityMapper(x)
%This function is a helper function to map different constant densities
%corresponding to different space values
%x is a space value, k_0_x = intial  density value at point x
if (x>=0)&&(x<500)
    k_0_x = 0.01;
elseif (x>=500)&&(x<600)
    k_0_x = 0.04;
elseif (x>=600)&&(x<950)
    k_0_x = 0.025;
elseif (x>=950)&&(x<=1000) %x<= 1000, needed here
    k_0_x = 0.02;
end

end

function lf = LegndreFenchel(u,km,v)
%This function is the legendre-fenchel transform for the greenshields
%u is the slope, km & v are greenshields constants, lf is the value of the
%transform
k = (km*(u+v))/(2*v);
lf = u*k+((v/km)*k)*(km-k);
end

function mf =  MoskowitzFunction(t,x,c,v,km)
%This function calculates the Moskwitz based on lax-hopf transform
%t is time, x is space, c is initial values vector , v & km are
%greenshields constants
```

```
%inf c (0,x+tu)+ t* lf(u)
candidates = [] ;

%The domain of u is [-v,v]
%the value of legnders-fenchel outside this domain is positive infinity
for u= -1*v:v
    ind = x+t*u;
    %For our case ind is between -38 & 2000, and the domain is 0,1000,
    %thus c(0,t) outside the domain will be positive infinity
    if ind <1 || ind > 1000
        candidates(end+1) = inf;
        continue;
    end
    candidates(end+1)= c(ind)+t*LegndreFenchel(u,km,v);


end

mf = min(candidates) ;

end
```

References

[1] Legendre and Legendre-Fenchel transform
http://www.onmyphd.com/?p=legendre.fenchel.transform

[2] Alexandre M. BAYEN, Christian CLAUDEL, Patrick SAINT-PIERRE, “Computation of solutions to the Moskowitz Hamilton-Jacobi-Bellman equation under viability constraints," 46th IEEE Conference on Decision and Control, 2007