Dynamic System Study

Mini-Project ISC4

Pierre Lague

Abstract

This project aims to familiarize students with the study of dynamic systems inspired from real-world problems (autonomous car insertion on a highway, consideration of sensor measurements uncertainties etc.). The project was implemented on the Octave GUI (substitute for MatLab). The answers to the questions will be divided in two parts :

- The theoretical part, finding and understanding the strategy
- The practical part, implementing the strategy on Octave GUI

Code annotation on this report are in english for consistency, code annotations on the real scripts are in french.

Theoretical part: determination of the open loop strategy

The first part aims to determine and understand the open loop strategy of the system (starting decision and starting acceleration).

(1) Choosing the appropriate two vehicles to insert the car between

Show that if $x_i(t) - x_{i+1}(t) > D_{\text{sav}} + D_0 + D_{\text{sar}} + D_i$, then the vehicles n°i and n°i + 1 are sufficiently distant.

... (see annexed draft)

(2) Compute D_{xx}, the distance margin

During an insertion, our vehicle has to respect the mandatory security distance but the distance between the vehicle in front of us, respectively behind us, is not always exactly the security distance. There is a margin called D_{xx}.

$$\mathrm{D}_{xx} = \frac{x_i - x_{i+1} - D_i - D_0 - D_{sav} - D_{sar}}{2}$$

This margin is in other words the error margin. If it wasn't used, our vehicle would insert itself exactly onto or within the security distances, going against the policy of our system.

(3) Compute T_a, the E.T.A. to insertion point

For each pair of vehicles where there is enough space, the question is whether the system can reach the insertion point $x_a(t) = x_{i+1}(t) + D_{sar} + D_0 + D_{xx}$ in time. We need to compute the time at which $x_a(t)$ will reach this abcissa $-\frac{3L_0}{4}$ (i.e. $x_a(t_0 + T_a) = -\frac{3L_0}{4}$).

$$\mathrm{T}_a = \frac{D_i}{v_f} = \frac{-\frac{3}{4}L_0 - x_a(t)}{v_f} = \frac{-\frac{3}{4}L_0 - (x_{i+1}(t) + D_{sar} + D_0 + D_{xx})}{v_f}$$

Thus, if $T_a > T_f$, our car will have enough time to reach the insertion point.

The project statement then gives us the relation between t, T_a and T_f to compute t_0, the time at which our system needs to start accelerating.

(4) Implementing the strategy In the "acceleration_BO.m" file, we copy and past the content of the "acceleration.m" file and make the following changes:

- Define D_i, length of the queue.
- Define D_{xx} the distance margin.
- Define T_a the E.T.A to the insertion point.

Open Loop Startegy "acceleration_BO.m"

```
#this is an extract of the script. The rest is linked with the projects archive.
if (N_file > 1)
        for i=1:N_file-1
            Deltax = File(i).pos_est - File(i+1).pos_est  ;% Distance inter-évhicule
            Di=File(i).long;
            if (Deltax> D0+Dsav+Dsar+Di)            % If there is enough place,
             ↪ verify that we can reach the insertion point.
                Dxx = (File(i).pos_est-File(i+1).pos_est-Di-D0-Dsar-Dsav)/2;    %
                 ↪ Distance margin
                Ta = (- 3*L0/4 - (File(i+1).pos_est  + D0 +Dsav+ Dxx))/vf_est ; %ETA
                 ↪ to insertion
                if Ta > Tf   % there is enough time
                    t0=t+Ta-Tf;% compute the starting time of the acceleration
                    acc_BO=vf_est/Tf;
                    demar=1;
                    N_suivi=i;
                    mess=sprintf('%s%s%s%s','On s''èinsre entre les
                     ↪ évhicules',Couleurs(File(N_suivi).icouleur,:), 'et',
                     ↪ Couleurs(File(N_suivi+1).icouleur,:));
                    disp(mess);
                    disp(['Distance entre les évhicules i et i+1 =
                     ↪ ',num2str(File(i).pos_est -File(i+1).pos_est
                     ↪ -File(i).long)]);
                    break;
                end
            else
            end

        end
    end
```

(5) Comments Before trying out the new strategy, we tested the demo code and observed that not only the security distances were not always respected, the insertion wasn't always possible because the system did not take its time to start accelerating. After implementing the new strategy, we tested it with different parameters (5000 cars per hour instead of default 3000). We observed that the system took it's time to find an appropriate space to start. And that the insertion was successful most of the time.

Practical Part : Consideration of sensor measurement uncertainties

In order to witness sensor measurement uncertainties leading to an unsuccessful insertion, we executed 10 insertions (with the "launch_simufile.m" script, diary called "q2_diary.txt") with the estimated parameters. Out of 10 experiments, 4 led to an emergency braking (this can be avoided by decreasing the rate of cars per hour). 4 led to a failure to respect safety distances and 2 led to successful insertions. This shows that the sensor measurement uncertainties in the open loop strategy can lead to unsuccessful insertion, hence a non reliable system. The necessity for a closed loop strategy is now undeniable.

Practical Part : Implementing the closed loop strategy

The subject gives us all the main formulas to compute the closed loop acceleration :

$$a(t) = \lambda\left(\dot{x}_i(t) - \dot{x}(t)\right) + \mu\left(x_i(t) - L_i - x(t)\right)$$

With the following :

- $\lambda = \frac{6}{T_{5\%}}$
- $\dot{x}_i(t) = v_f$
- $\dot{x}(t)$: vit (variable defined l.99 in the "simufile.m" script)
- $\mu = \left(\frac{3}{T_{5\%}}\right)^2$
- $L_i = D_i + D_{xx} + D_{sav}$
- $x(t)$: pos (variable defined l.99 in the "simufile.m" script)

This allows us to implement the following main loop :

Closed Loop Startegy "acceleration_BF.m"

```
if (N_file > 1)
        for i=1:N_file-1
            Deltax = File(i).pos_est - File(i+1).pos_est
            Di=File(i).long;
            Xt = -3*L0/4;
            if (Deltax> D0+Dsav+Dsar+Di)
                Dxx = (File(i).pos_est-File(i+1).pos_est-Di-D0-Dsar-Dsav)/2;
                Ta = (- 3*L0/4 - (File(i+1).pos_est  + D0 +Dsav+ Dxx))/vf_est ;
                if Ta > Tf
                    t0=t+Ta-Tf;

                    %formulas found in the subject
                    Li = Di + Dxx + Dsav
                    T5 = L0/2*vf_est;
                    mu = (3/T5)^2
                    lambda = (6/T5)
                    demar=1;
                    N_suivi=i;
                    acc_BF = lambda*(vf_est-vit)+mu*(File(N_suivi).pos_est-Li-pos);

                    mess=sprintf('%s%s%s%s','On s''èinsre entre les
                     ↪ évhicules',Couleurs(File(N_suivi).icouleur,:), 'et',
                     ↪ Couleurs(File(N_suivi+1).icouleur,:));
                    disp(mess);
                    disp(['Distance entre les évhicules i et i+1 =
                     ↪ ',num2str(File(i).pos_est -File(i+1).pos_est
                     ↪ -File(i).long)]);
                    break;
                end
            else
            end
        end
    end
```

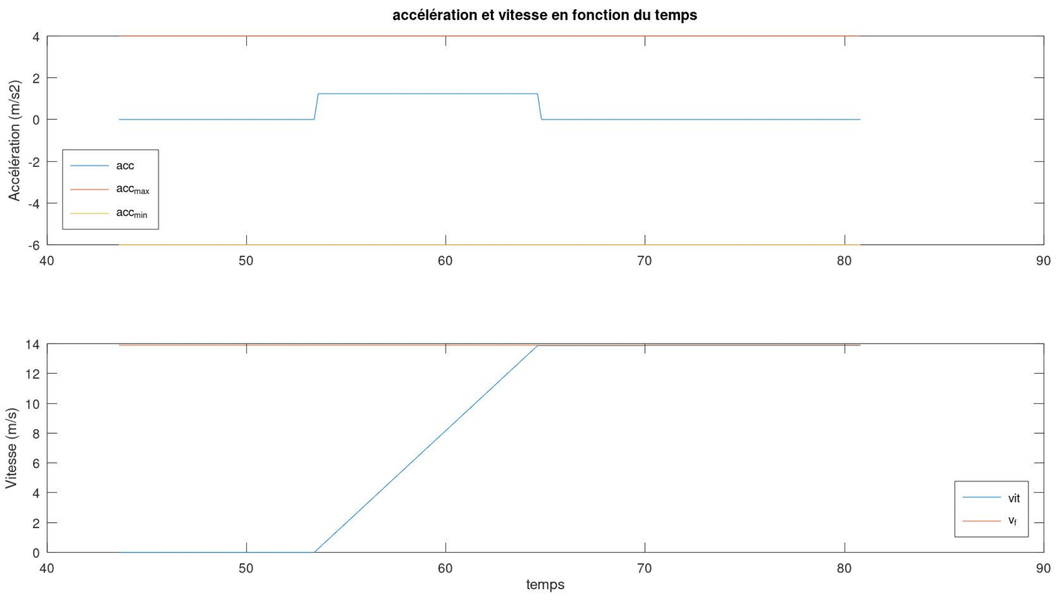


Figure 1: Acceleration and speed of the system. Rate : 5000 cars per hour.

Complementary Work

After a few tests, at a rate of 5000 cars, we've seen that the maximum speed at which the car is able to insert itself successfully is 70 km/h. At a rate of 3000 cars, the maximum speed at which the car is able to insert itself successfully is 130 km/h (legal limit in France). In this case however we gave to exceed to acceleration limitation of the system.

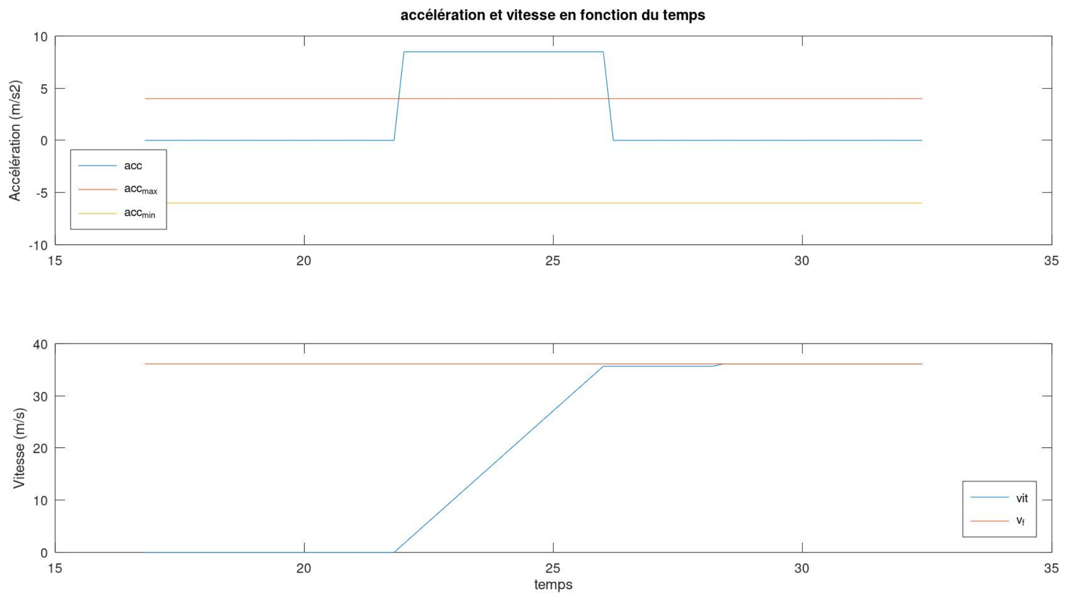
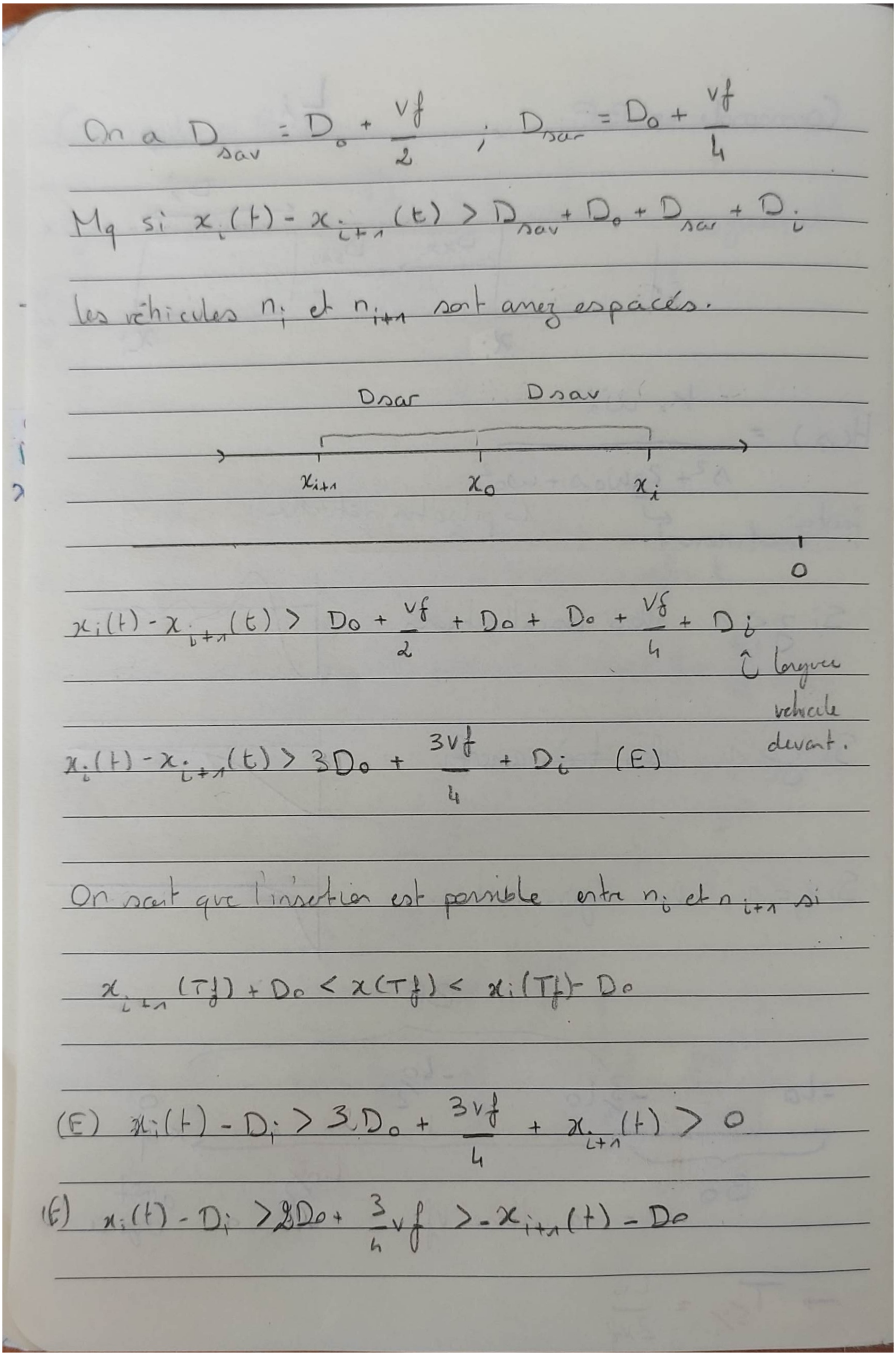


Figure 2: Acceleration and speed of the system. Rate : 3000 cars per hour. Vf = 130 km/h

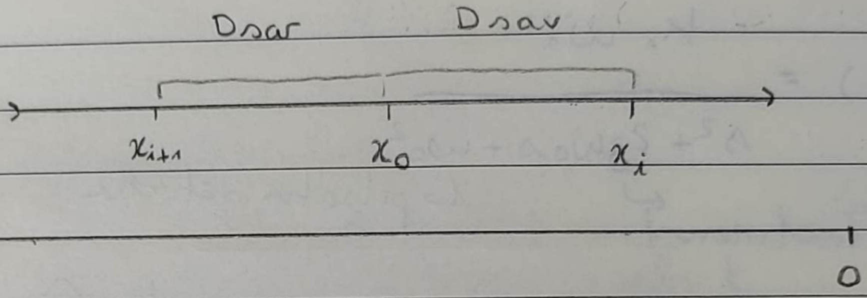
Annexes



On a $D_{sav} = D_0 + \frac{vf}{2}$; $D_{sar} = D_0 + \frac{vf}{4}$

Mq si $x_i(t) - x_{i+1}(t) > D_{sav} + D_0 + D_{sar} + D_i$

les véhicules n_i et n_{i+1} sont assez espacés.



$x_i(t) - x_{i+1}(t) > D_0 + \frac{vf}{2} + D_0 + D_0 + \frac{vf}{4} + D_i$

$\hat{}$ longueur véhicule devant.

$x_i(t) - x_{i+1}(t) > 3D_0 + \frac{3vf}{4} + D_i$ (E)

On sait que l'insertion est possible entre n_i et n_{i+1} si

$x_{i+1}(Tf) + D_0 < x(Tf) < x_i(Tf) - D_0$

(E) $x_i(t) - D_i > 3.D_0 + \frac{3vf}{4} + x_{i+1}(t) > 0$

(E) $x_i(t) - D_i > 2D_0 + \frac{3}{4}vf > -x_{i+1}(t) - D_0$

Figure 3: Draft for Q1