num_veh = 5

Define Global Parameters (Q_z, $Q_{z'}$, H, ...).

Get Preceding Vehicle Trajectory

Generate Vehicle Objects in platoon

Generate trajectories for vehicles in platoon

for veh in (1→5)

    Initialize_Variables() (All pos/vel/ace parameters needed should be initialized)

for k in (control steps 0→n-1)

    for veh in (1→5)

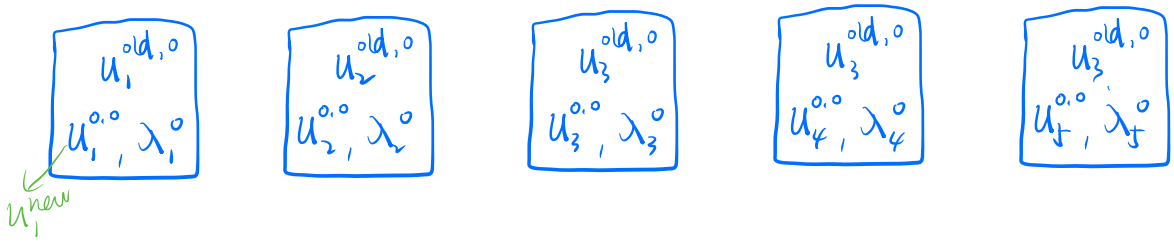
        Update_VP_buffer() (All vehicles share its vel/pos to the buffers one-by-one)

    for veh in (1→5):

        Initialize_Controller() (All vehs calculate everything it needs for controller, just left u_i and x_i to be iterated)

        Define a small enough initial u_i^{old} ($< a_{min} - |a_{max} - a_{min}|$)

        Update_u_traj (u_i^{new})



sum_flag_λ = 0;

while sum_flag_λ ≠ 5:

    sum_flag_u = 0;

    while sum_flag_u ≠ 5:

for veh in (1→5):

Σ_Buffersharing() (All vehicle share its latest u_i^{new} value to the corresponding place of the buffer one-by-one)

for veh in (1→5)

if $\|u_1^{new} - u_1^{old}\| < \varepsilon_1$

$u_1^{0,1} = u_1^{0,0}$

$u_1^{old} = u_1^{new}$

$u_1^{raw} = u_1^{0,1}$

update_u_traj(u_1^{new})

if $\|u_4^{new} - u_4^{old}\| \geq \varepsilon_1$

compute $u_4^{0,1}$

$u_4^{old} = u_4^{new}$

$u_4^{raw} = u_4^{0,1}$

(if sum_flag_u ≠ 5)

(if sum_flag_u = 5)

for veh in (1→5):

Σ_Buffersharing() (All vehicle share its latest u value to the corresponding place of the buffer one-by-one)

for veh in (1→5):

compute λ_1' (λ_1^{new}) λ_2' λ_3' λ_4' λ_5'

update_λ_traj()

$\|\lambda_1^{new} - \lambda_1^{old}\|$ flag_λ_1 flag_λ_2

(if sum_flag_λ ≠ 5)

(if sum_flag_λ = 5)

```
for veh in (1->5):
    I_Buffersharing() ⇒ control step u⃗

for veh in (1->5):
    Compute_Next_State()   (Compute v_i(k+1), p_i(k+1))
    Update_Ego_State()

Update_Ego_Traj_Df()
```