



UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

ELABORATO PER L'ESAME DI: SISTEMI DI TRASPORTO INTELLIGENTI

A cura di:

Dario Fiore
M62002987

ESERCITAZIONE RAMP METERING

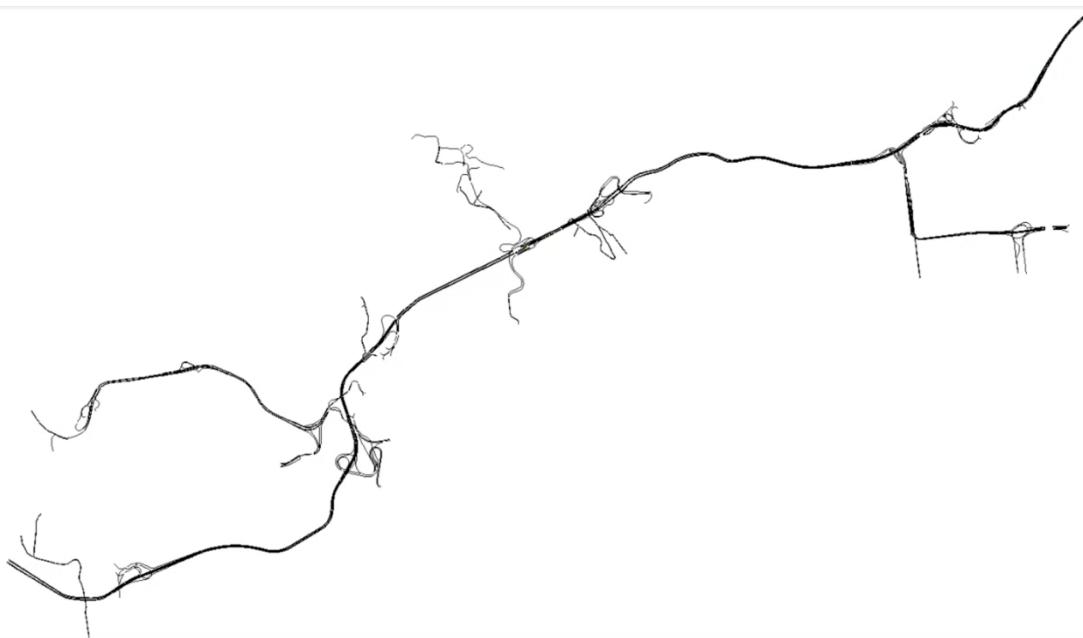
Il ramp metering è una gestione degli accessi in ambito extraurbano che permette sia una miglior condizione di deflusso (principalmente sulla main lane) che una gestione migliore degli ingressi dalle rampe.

A seconda di alcune condizioni di traffico che noi andiamo a ritrovare sulla corsia principale, noi possiamo prendere delle decisioni atte a migliorarne il deflusso; ma solitamente si dovrebbe ricercare un ottimo di sistema, quindi non penalizzare solo quelli sulla rampa.

Perché se chiudo totalmente gli ingressi sulla rampa, i veicoli sulla main lane raggiungeranno si una condizione di free flow, ma quelli sulla rampa non vanno bene per niente, motivo per cui si cerca di raggiungere sempre un ottimo di sistema.

Però, nella maggior parte dei casi, visto che gli accessi dai varchi sono sempre minori rispetto ai flussi della corsia principale, si predilige migliorare le condizioni di deflusso sulla corsia principale.

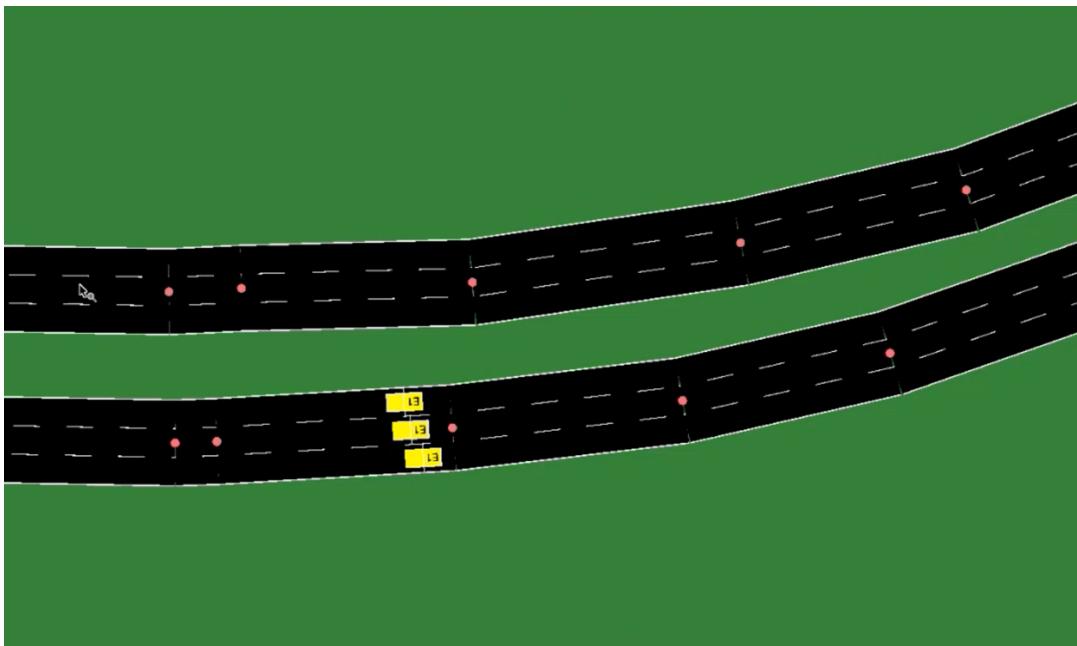
Nel nostro caso lo scenario è un pezzo della tangenziale di Napoli, tecnicamente da Fuorigrotta che va avanti fino a oltre Capodichino. Questo scenario è stato creato banalmente andando a scaricare da Open Street Map la porzione di mappa della tangenziale e rimuovendo tutte le strade che si ritrovavano all'interno del quadrato (andando a fare una cancellazione per tipologie di deflusso).



Gli scenari che tecnicamente siamo andati a studiare sono quattro, e sono a informazioni crescenti.

Nello **scenario 0** consideriamo il flusso solo sulla corsia principale, senza altri flussi nella nostra rete, quindi noi andando a fare una simulazione del flusso sulla corsia principale, possiamo osservare le condizioni di deflusso di quell'arteria.

Nella realtà, abbiamo visto che per poter monitorare realmente una sezione stradale, quello che bisogna fare è inserire dei detector, quindi dei punti della mia sezione in cui posso misurare le condizioni di deflusso (i parametri di flusso velocità e densità).



NB: In questo caso ricordiamo che non è densità ma è occupancy, dall'occupancy si può ricavare la densità perché uno è riferito al numero di passaggi al km mentre l'altro è riferito al numero di passaggi nell'unità di tempo.

Quello bisogna fare in estrema sintesi è un *run* di questa simulazione e leggerci le condizioni di deflusso monitorate per poter andare a fare un plot che ci possa far vedere i valori della velocità nel tempo. Cioè andremo a diagrammare, tramite uno specifico diagramma di matlab, una *mesh*, in cui sull'asse delle ascisse c'è il tempo e sull'asse delle ordinate la velocità per ogni sezione.

Nello **scenario 1**, inseriamo un flusso da una delle rampe della tangenziale, ai fini esercitativi sceglieremo una sola rampa perché noi vogliamo vedere il comportamento di un tronco di main lane + rampa. Quindi andiamo a studiare i dati ricavati dai detector a seguito dell'interazione tra veicoli sulla rampa che si immettono nella main lane e veicoli sulla main lane, producendo un diagramma mesh.

Nello **scenario 2** abbiamo uno scenario di ramp metering che nella realtà non esiste, ma lo abbiamo studiato per mostrare anche come è la differenza tra un semaforo a ciclo fisso e un semaforo controllato. Quindi nello scenario 2 c'è un semaforo a ciclo fisso, in cui ogni *delta t* tempo chiude gli accessi e poi li riapre.

Lo **scenario 3** è lo scenario in cui c'è un semaforo sulla rampa che però è controllato dalla logica di controllo. La logica di controllo, con riferimento a una specifica sezione della corsia principale, andrà a monitorare le condizioni di deflusso. Ci sarà una condizione, che siamo

noi progettisti a dover definire, oltre la quale andrà a chiudere la rampa, quindi a mettere il semaforo a rosso.

Messo il semaforo a rosso, la simulazione continuerà ad andare avanti, noi continueremo a monitorare le condizioni di deflusso sulla corsia principale e avremo una seconda condizione, in corrispondenza della quale potremo ritenere il deflusso tornato normale e potremo riaprire la rampa.

Ma questo controllo non basta, perché se le condizioni sulla corsia principale, nel mentre che il semaforo sulla rampa è settato a rosso, non migliorano velocemente, potrebbe accadere che la rampa inizia a congestionarsi totalmente, rigurgita la coda nel tratto urbano di accesso alla rampa.

Quindi, in definitiva, abbiamo bisogno di monitorare contemporaneamente due sezioni: la corsia principale e la corsia della rampa.

Soltanmente la corsia della rampa viene monitorata come lunghezza massima di coda che noi vogliamo ottenere.

C'è da stressare il fatto che fenomeni di ramp metering, o comunque servizi I.T.S., servono a migliorare le condizioni di deflusso se e solo se la domanda (quindi il flusso veicolare che arriva) ha un fenomeno di picco ma poi ha una decrescita. Perché se abbiamo lunghi momenti di congestione, sopra la capacità della nostra struttura, non esiste servizio I.T.S. che possa far scomparire questo fenomeno. Quindi un aspetto importante è la domanda che andiamo a considerare.

SCENARIO 0-1-2

Noi quando andiamo ad aggiungere un detector di tipo E1 sulla nostra rete e salviamo, si apre un file xml nella nostra cartella in cui ci sono le informazioni del detector (su che corsia si trova, a che ascissa curvilinea di quel pezzo di arco si trova, la frequenza di aggregazione ex. ogni 60 sec, etc..).

```
<additionals>
    <e1Detector id="15.2_0" lane="29436874#1_0" pos="25.48" freq="60.00" file="output_detector\15.2_0.txt" friendlyPos="0"/>
    <e1Detector id="15.2_1" lane="29436874#1_1" pos="25.41" freq="60.00" file="output_detector\15.2_1.txt" friendlyPos="0"/>
    <e1Detector id="15.2_2" lane="29436874#1_2" pos="25.31" freq="60.00" file="output_detector\15.2_2.txt" friendlyPos="0"/>
    <e1Detector id="15.2_rampa" lane="413953552_0" pos="2.02" freq="60.00" file="output_detector\15.2_rampa.txt" friendlyPos="0"/>
    <e1Detector id="15.3_0" lane="29436874#1_0" pos="125.50" freq="60.00" file="output_detector\15.3_0.txt" friendlyPos="0"/>
    <e1Detector id="15.3_1" lane="29436874#1_1" pos="125.27" freq="60.00" file="output_detector\15.3_1.txt" friendlyPos="0"/>
    <e1Detector id="15.3_2" lane="29436874#1_2" pos="125.09" freq="60.00" file="output_detector\15.3_2.txt" friendlyPos="0"/>
    <e1Detector id="15.3_rampa" lane="413953552_0" pos="96.92" freq="60.00" file="output_detector\15.3_rampa.txt" friendlyPos="0"/>
    <e1Detector id="15.4_0" lane="29436874#1_0" pos="225.15" freq="60.00" file="output_detector\15.4_0.txt" friendlyPos="1"/>
    <e1Detector id="15.4_1" lane="29436874#1_1" pos="225.13" freq="60.00" file="output_detector\15.4_1.txt" friendlyPos="1"/>
    <e1Detector id="15.4_2" lane="29436874#1_2" pos="225.20" freq="60.00" file="output_detector\15.4_2.txt" friendlyPos="1"/>
    <e1Detector id="15.4_rampa" lane="413953552_0" pos="197.63" freq="60.00" file="output_detector\15.4_rampa.txt" friendlyPos="0"/>
    <e1Detector id="15.55_0" lane="414445839-AddedOnRampEdge_1" pos="37.87" freq="60.00" file="output_detector\15.55_0.txt" friendlyPos="1"/>
    <e1Detector id="15.55_1" lane="414445839-AddedOnRampEdge_2" pos="38.05" freq="60.00" file="output_detector\15.55_1.txt" friendlyPos="1"/>
    <e1Detector id="15.55_2" lane="414445839-AddedOnRampEdge_3" pos="37.95" freq="60.00" file="output_detector\15.55_2.txt" friendlyPos="1"/>
```

Quindi sostanzialmente abbiamo un file generato, questo file ha un nome, generalmente pari al nome del detector.txt.

Ora se facciamo girare la simulazione senza fare nessun controllo, succede che questi file di testo vengono riempiti con le informazioni rilevate dai detector.



```
</configuration>
-->

<detector xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/det_el_file.xsd">
    <interval begin="0.00" end="60.00" id="15_2_0" nVehContrib="0" flow="0.00" occupancy="0.00" speed="-1.00" harmonicMeanSpeed="-1.00" length="-1.00" nVehEntered="0"/>
    <interval begin="60.00" end="120.00" id="15_2_0" nVehContrib="4" flow="240.00" occupancy="1.39" speed="20.55" harmonicMeanSpeed="20.55" length="4.30" nVehEntered="4"/>
    <interval begin="120.00" end="180.00" id="15_2_0" nVehContrib="12" flow="720.00" occupancy="4.16" speed="20.69" harmonicMeanSpeed="20.66" length="4.30" nVehEntered="12"/>
    <interval begin="180.00" end="240.00" id="15_2_0" nVehContrib="17" flow="1020.00" occupancy="5.78" speed="21.10" harmonicMeanSpeed="21.07" length="4.30" nVehEntered="17"/>
    <interval begin="240.00" end="300.00" id="15_2_0" nVehContrib="20" flow="1200.00" occupancy="6.96" speed="20.66" harmonicMeanSpeed="20.60" length="4.30" nVehEntered="20"/>
    <interval begin="300.00" end="360.00" id="15_2_0" nVehContrib="28" flow="1680.00" occupancy="10.59" speed="19.81" harmonicMeanSpeed="19.73" length="4.47" nVehEntered="28"/>
    <interval begin="360.00" end="420.00" id="15_2_0" nVehContrib="35" flow="2100.00" occupancy="12.74" speed="19.96" harmonicMeanSpeed="19.95" length="4.36" nVehEntered="35"/>
    <interval begin="420.00" end="480.00" id="15_2_0" nVehContrib="34" flow="2040.00" occupancy="13.10" speed="18.90" harmonicMeanSpeed="18.85" length="4.36" nVehEntered="34"/>
    <interval begin="480.00" end="540.00" id="15_2_0" nVehContrib="34" flow="2040.00" occupancy="13.81" speed="18.01" harmonicMeanSpeed="17.86" length="4.36" nVehEntered="34"/>
    <interval begin="540.00" end="600.00" id="15_2_0" nVehContrib="24" flow="1440.00" occupancy="23.17" speed="10.05" harmonicMeanSpeed="7.42" length="4.30" nVehEntered="24"/>
    <interval begin="600.00" end="660.00" id="15_2_0" nVehContrib="26" flow="1560.00" occupancy="26.41" speed="10.62" harmonicMeanSpeed="7.30" length="4.38" nVehEntered="26"/>
    <interval begin="660.00" end="720.00" id="15_2_0" nVehContrib="18" flow="1080.00" occupancy="43.02" speed="4.11" />

```

Questo però è un file facilmente leggibile tramite uno script matlab, perché se vediamo questo file è formattato in maniera tale da avere una sua ricorsività.

Se vediamo ha un campo che inizia sempre con un campo “interval” e finisce con questo carattere ”/>”.

NB: questo file è di uno specifico detector, se ne ho tanti di detector, ho tanti file come questo.

Facendo riferimento a una specifica riga, possiamo osservare che c’è qualcosa che si presenta in maniera ricorsiva, ossia che ogni valore di mio interesse si trova tra due apici, questa caratteristica mi permette di dare un metodo a matlab, che gli permette di andare a individuare, tramite un ciclo for, in che posizione si trovano quelle scritte viola che mi servono. Quindi il mio script matlab cosa deve andare a fare? Deve andare a porre una doppia condizione di find:

- La prima che va a trovare nell’intero file dove si trova il campo “interval” e il campo “/>”;
- La seconda è, all’interno del pezzo di stringa individuato tra i due campi di cui sopra, trovare dove si trovano i caratteri dei doppi apici e leggere l’informazione all’interno.

Ovviamente questa operazione va reiterata per ogni riga del file txt tramite un ciclo for.

The screenshot shows the MATLAB environment. On the left, the 'Current Folder' browser displays a list of files: 15.2_0.txt, 15.2_1.txt, 15.2_2.txt, 15.2_rampa.txt, 15.3_0.txt, 15.3_1.txt, 15.3_2.txt, 15.3_rampa.txt, 15.4_0.txt, 15.4_1.txt, 15.4_2.txt, 15.4_rampa.txt, 15.5_0.txt, 15.5_1.txt, 15.5_2.txt, 15.5_rampa.txt, 15.6_0.txt, 15.6_1.txt, 15.6_2.txt, 15.6_rampa.txt, and 15.7_0.txt. On the right, the 'Editor' window contains the script 'Lettura_InductionLoop_postProcessed.m' with the following code:

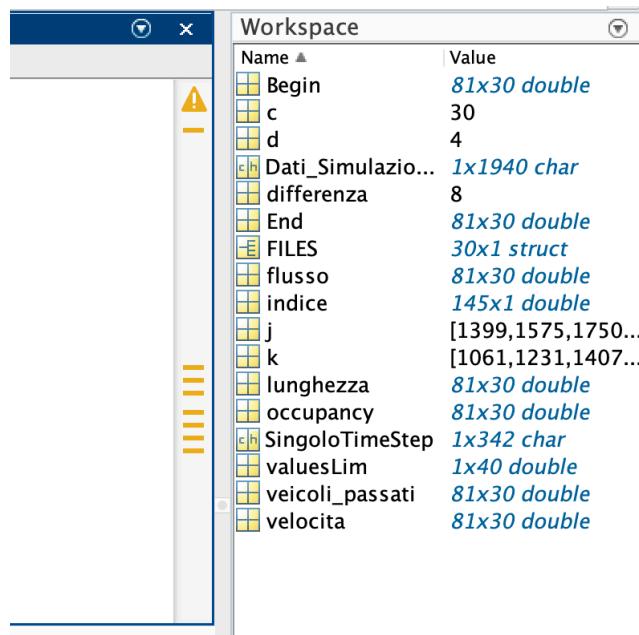
```

1 clear all
2 % close all
3
4 FILES = dir('*.txt');
5
6
7
8 for c = 1:length(FILES)
9     Dati_Simulazione = fileread (FILES(c).name);
10    k = strfind(Dati_Simulazione,'<interval');
11    j = strfind(Dati_Simulazione,'/>');
12    differenza = length(j)+1-length(k);
13    j (1:differenza) = [];
14    for d=1:length(j)
15        SingoloTimeStep=Dati_Simulazione(k(d):j(d));
16        valuesLim=find(SingoloTimeStep=='''');
17
18        Begin(d,c)=str2double(SingoloTimeStep(valuesLim(1)+1:valuesLim(2)-1));
19        End(d,c)=str2double(SingoloTimeStep(valuesLim(3)+1:valuesLim(4)-1));

```

A sinistra possiamo vedere la current folder, che è la cartella in cui ci sono tutti i file di output dei detector.

Una volta che la current folder è selezionata correttamente, si fa RUN e si ottengono delle variabili all'interno del nostro workspace:



Lo script cosa fa? Genera un campo Files utilizzando il comando dir che va a leggere tutte le informazioni presenti nella current folder. Con la riga 4 io sto dicendo al matlab di leggermi tutti i file della current folder che hanno come suffisso .txt.

NB: l'asterisco sta a dire non fare caso a ciò che è scritto prima, prendimi tutti i file che finiscono con .txt.

Ora vediamo un po' gli output di questo script...Questo script ci ha creato una struttura in cui ogni riga è un intervallo di simulazione, ogni colonna è in riferimento a uno specifico detector.

In questo caso l'ordine dei detector ce lo da l'ordine presente del campo files, quindi per vedere a che sezione fa riferimento una colonna dobbiamo andare a vedere i nomi del campo files.

Ma andiamo a vedere questi nomi come sono stati generati...quando abbiamo generato i nomi degli induction loop siamo stati previdenti, cioè noi abbiamo messo dei suffissi uguali per tutti per le sezioni. Se andiamo a vedere i nomi di una singola sezione hanno tutti lo stesso suffisso iniziale, ossia ad esempio 15.2_, poi a seconda della corsia avremo 15.2_0, 15.2_1, 15.2_2, perché sumo va da 0 a n, utilizza la dicitura python based dove il primo indice è 0. Quindi da questo possiamo evincere che tutti i file appartenente ad un'unica sezione sono consecutivi tra di loro, quindi io posso analizzare le colonne delle variabili che ha creato il mio script (velocità, flusso, occupancy etc..) a blocchi di 3, dove un singolo blocco di 3 mi identifica la sezione:

FILES						
27x1 struct with 6 fields						
Fields	name	folder	date	bytes	isdir	datenum
1	'15.2_0.txt'	'C:\Users\L...	'02-dic-202...	15857	0	7.3959e+05
2	'15.2_1.txt'	'C:\Users\L...	'02-dic-202...	15846	0	7.3959e+05
3	'15.2_2.txt'	'C:\Users\L...	'02-dic-202...	15798	0	7.3959e+05
4	'15.2_rampa.txt'	'C:\Users\L...	'02-dic-202...	15805	0	7.3959e+05
5	'15.3_0.txt'	'C:\Users\L...	'02-dic-202...	15858	0	7.3959e+05
6	'15.3_1.txt'	'C:\Users\L...	'02-dic-202...	15850	0	7.3959e+05
7	'15.3_2.txt'	'C:\Users\L...	'02-dic-202...	15804	0	7.3959e+05
8	'15.3_rampa.txt'	'C:\Users\L...	'02-dic-202...	15805	0	7.3959e+05
9	'15.4_0.txt'	'C:\Users\L...	'02-dic-202...	15857	0	7.3959e+05
10	'15.4_1.txt'	'C:\Users\L...	'02-dic-202...	15854	0	7.3959e+05
11	'15.4_2.txt'	'C:\Users\L...	'02-dic-202...	15802	0	7.3959e+05
12	'15.4_rampa.txt'	'C:\Users\L...	'02-dic-202...	15805	0	7.3959e+05
13	'15.55_0.txt'	'C:\Users\L...	'02-dic-202...	15937	0	7.3959e+05
14	'15.55_1.txt'	'C:\Users\L...	'02-dic-202...	15934	0	7.3959e+05
15	'15.55_2.txt'	'C:\Users\L...	'02-dic-202...	15880	0	7.3959e+05
16	'15.55 rampa.txt'	'C:\Users\L...	'02-dic-202...	15887	0	7.3959e+05

Se io voglio poter aggregare questi dati in un'aggregazione di sezione, cosa posso fare?

Con riferimento al campo velocità, devo prendermi questi tre valori e costruirmi, per ogni intervallo, un valore di SEZIONE, quindi per ogni riga prendo le colonne a gruppi di 3 e faccio il valor medio, e così via per ogni riga.

FILES													
velocita													
81x27 double													
1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222
2	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222
3	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222
4	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222	22.2222
5	23.0600	22.2222	22.2222	22.2222	23.2100	22.2222	22.2222	22.2222	23.3600	22.2222	22.2222	22.2222	23.3900
6	20.4300	20.4000	20.7900	22.2222	21.0800	20.7200	21.5200	22.2222	21.3700	21.6800	22.2222	22.2222	21.9400
7	18.1800	19.0500	20.5400	22.2222	18.9400	19.4200	20.2200	22.2222	18.2600	19.5200	20.5700	22.2222	18.3600
8	18.5500	20.7200	21.8100	22.2222	18.9200	20.7100	21.8000	22.2222	18.5500	20.4100	21.5200	22.2222	18.2000
9	17.5000	19.2300	20.6200	22.2222	17.0400	19.3900	20.8700	22.2222	18.7300	19.9900	20.9000	22.2222	18.3500
10	18.4200	19.5700	20.8800	22.2222	17.9600	19.0600	20.5000	22.2222	18.3200	18.9100	20.2600	22.2222	18.0600
11	16.3600	18.2900	19.9100	22.2222	18.4000	19.0800	20.5900	22.2222	19.0800	19.9500	20.5800	22.2222	18.4600
12	18.3900	20.6800	21.6800	22.2222	18.2000	20.2800	20.8500	22.2222	18.3500	20.3800	21.5600	22.2222	18.4800
13	17.9600	19.8300	20.0500	22.2222	17.6200	20.4100	22.0100	22.2222	17.6200	20.5600	22.1500	22.2222	17.3300
14	17.7000	19.3800	21.4500	22.2222	17.6500	19.2800	21.3300	22.2222	17.8200	19.1000	21.1300	22.2222	17.8300
15	18.9900	19.1700	21.1500	22.2222	19.1400	19.1000	21.4000	22.2222	18.2800	19.2900	21.2500	22.2222	17.7500
16	18.5100	19.7900	22.0500	22.2222	18.7700	20.4300	22.1100	22.2222	18.8500	19.9300	22.2600	22.2222	18.4400

Per fare questo, innanzitutto se io in un intervallo temporale non rивelo nessun veicolo che passa, allora matlab rileverà una velocità pari a -1, per cui innanzitutto sostituiamo quei valori di velocità con *NaN* (not a number), dopodiché vado a calcolare i valori medi di quelle sezioni, per ogni sezione, con il comando *mean*:

```

Editor - /Users/dario/Desktop/ramp_metering_scenari_2/lettura_InductionLoop_postProcessed.m
Editor - /Users/dario/Desktop/ramp_metering_scenari_2/lettura_inductionloop_RM_post_processed.m

20 end
21
22
23 %% nelle prossime due righe vado ad eliminare, solo per l'array delle velocità i dati riferiti
24 velocita(velocita == -1) = NaN;
25 velocita(:,4:4:24)=[];
26
27 %% In questo caso vado a calcolarmi il valore medio di ogni sezione
28 Sez_1 = (nanmean(velocita(:,1:3),2))';
29 Sez_2 = (nanmean(velocita(:,4:6),2))';
30 Sez_3 = (nanmean(velocita(:,7:9),2))';
31 Sez_4 = (nanmean(velocita(:,10:12),2))';
32 Sez_5 = (nanmean(velocita(:,13:15),2))';
33 Sez_6 = (nanmean(velocita(:,16:18),2))';
34 Sez_7 = (nanmean(velocita(:,19:21),2))';
35

```

Ora passiamo al comando di *imagesc*, che è un comando di matlab che consiste in un metodo di plot che è un'immagine: sull'asse delle ascisse noi abbiamo il tempo, sull'asse delle ordinate abbiamo le varie sezioni partendo da quella a monte (dopo l'incrocio della rampa) a quella a valle (prima l'incrocio della rampa), e noi diagrammeremo per ogni sezione il valore della velocità.

Quindi dobbiamo costruirci una matrice in cui sulle righe ha l'aspetto del tempo, sulle colonne, ogni colonna è il valore in quell'istante temporale della velocità in quella sezione. Quindi nello script abbiamo visto che vado a calcolarmi la velocità media per ogni sezione, dopodiché vado a metterle nell'ordine che interessano a me, cioè dalla sezione più a monte a quella più a valle.

```

%% Qui inserisco le nuove variabili singole in un unico array in modo da esser diagrammate
MeshSpeeds =[Sez_7;Sez_6;Sez_5;Sez_4;Sez_3;Sez_2;Sez_1];
%% nella riga 43 non considero i primi 10 minuti di simulazione considerandoli di warm up

```

Una volta creato questo array, noi dobbiamo comunque non considerare i primi 10 minuti di simulazione, perché noi sappiamo che partiamo da una situazione di rete scarica, quindi la rete si deve caricare. Una volta che la rete è caricata, va in una condizione di esercizio. Quindi noi non dobbiamo considerare questa fase di *warm up* per cui non andiamo a considerare le prime 10 righe di questa variabile (perché noi ricordiamo che ogni riga corrisponde a 1 minuto di simulazione).

Quindi andiamo a crearci una variabile *Mesh_2* che non considera le prime dieci righe:

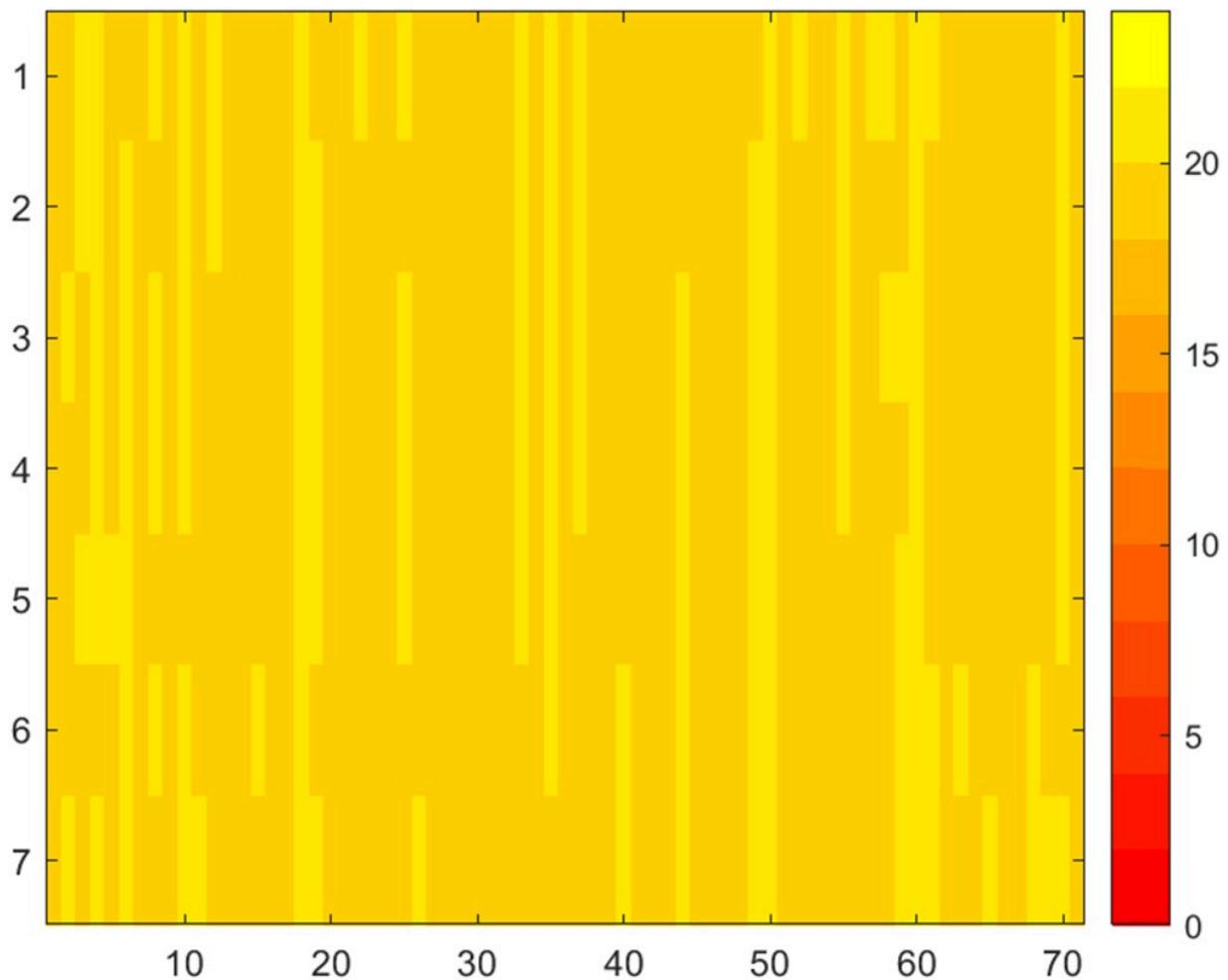
```

38 %% nella riga 43 non considero i primi 10 minuti di simulazione considerandoli di warm up
39 Mesh_2 = MeshSpeeds(:,10:end-1);
40 figure()

```

Di seguito osserviamo il plot che risulta dall'analisi post processing dei dati, come possiamo osservare sulle ordinate a destra c'è una scala della velocità che va da 0 a 20 e passa metri al

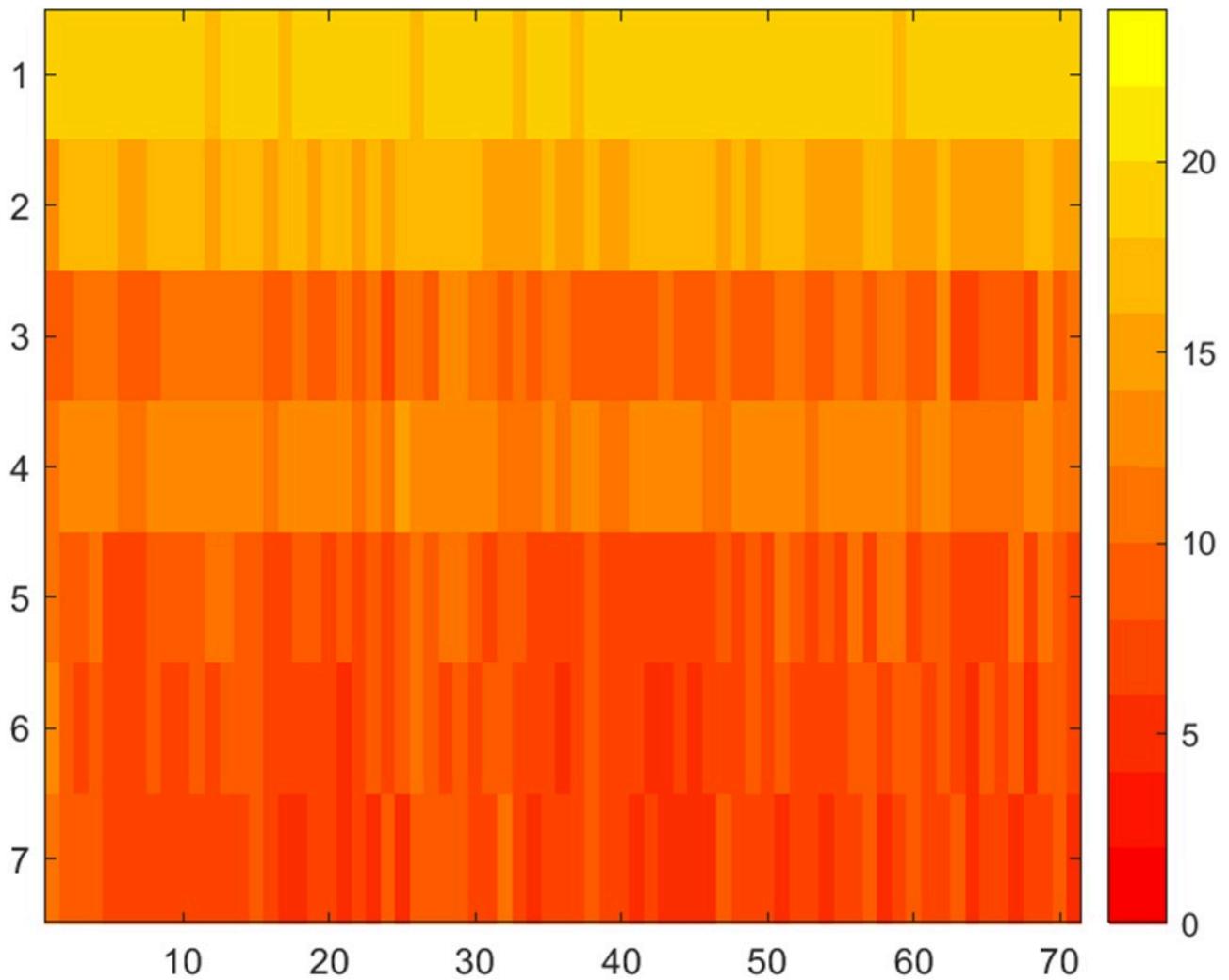
secondo, che bisogna moltiplicare per 3.6 per avere i km/h, sulle ascisse abbiamo il tempo e sulle ordinate a sinistra abbiamo le sezioni in ordine da quella più a valle a quella più a monte. Siccome in questo scenario abbiamo solo il flusso sulla main lane e non abbiamo accessi dalle rampe, sostanzialmente la condizione di deflusso è di free flow per tutto il tempo di simulazione, infatti possiamo vedere colori tra il giallo e l'arancio chiarissimo che si aggirano tra valori di 20-25 m/s cioè tra 70-90 km/h.



SCENARIO 1

Nello scenario 1 si considera il flusso sia sulla corsia principale che sulla rampa, in assenza di semaforo, quindi in tutto il tempo di simulazione possiamo osservare l'interazione tra veicoli sulla rampa che cercano di immettersi sulla corsia principale e veicoli sulla corsia principale. Noi sappiamo che la sezione 1 è la sezione dopo la rampa, infatti le condizioni di deflusso dopo un restringimento, dopo un bottleneck, sono condizioni quasi di free flow, dico quasi perché di fatti non hanno più conflitti ma dopo il bottleneck devono accelerare, infatti nel diagramma abbiamo colori sull'arancio chiaro, cioè valori di velocità intorno ai 50 km/h, diciamo che condizioni di free flow sono intorno a 75-80km/h, ma non vedo quei valori perché loro stanno appena accelerando, siamo troppo vicini al bottleneck.

Mentre invece, le zone prima del bottleneck, le code rigurgitano verso dietro, abbiamo una backward propagation, perché il fenomeno di coda è andare avanti nel tempo, indietro nello spazio.

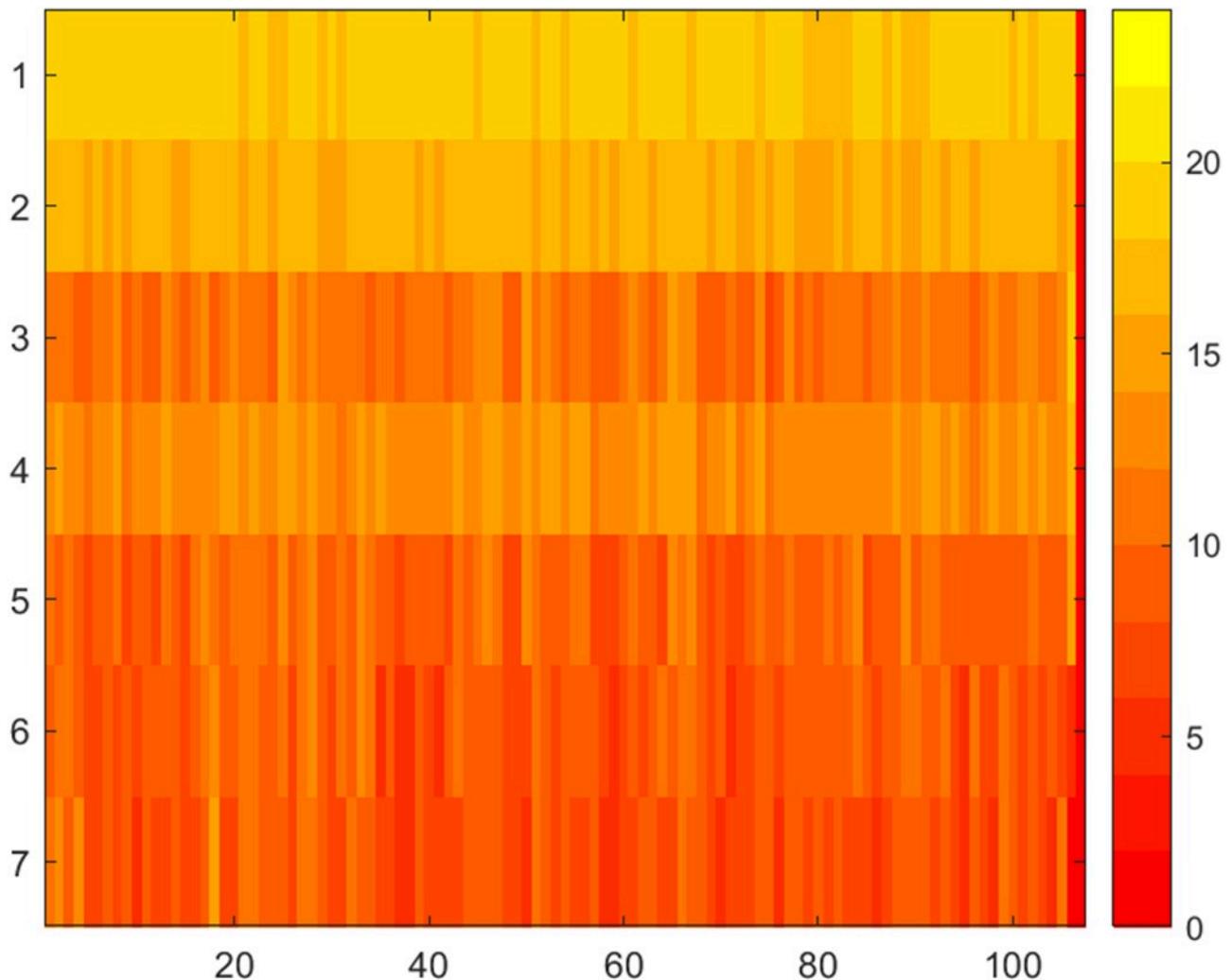


SCENARIO 2

Nello scenario 2 si inserisce un semaforo a ciclo fisso, quindi senza alcuna logica di controllo, in una circostanza del genere potremmo osservare leggeri miglioramenti ma questa cosa non è sicura perché il semaforo potrebbe passare a rosso mentre sulla main lane ci sono condizioni di deflusso inesistenti oppure mentre sulla rampa non c'è alcun veicolo, e viceversa.

Infatti possiamo osservare un fenomeno che è interpretabile come la cosiddetta shock-wave, cioè vediamo degli spazi rossi e poi vediamo delle colonne, non colonne verticali ma leggermente oblique, di colori più chiari, perché di fatti si blocca tutto sulla corsia principale (zona rossa), in quel momento potrebbe chiudersi la rampa, quindi incomincia a defluire il traffico sulla corsia principale e osservo zone più chiare, dopodiché una volta che la rampa si riapre, man mano si arriva nuovamente a una situazione più congestionata e così via...

Questo fenomeno si traduce in delle diagonali di colore più chiaro che nella realtà possono essere interpretate come delle shock wave. Ovviamente in questo caso il mio grado di libertà è il ciclo semaforico, io progettista potrei settare il tempo di verde al minimo e avvantaggiare il flusso sulla main lane a rischio del fatto che la coda sulla rampa potrebbe rigurgitare sul tratto urbano.



SCENARIO 3 DI CONTROLLO

Il controllo non è altro che, ogni delta t di tempo che io aggrego (perché passo a un discreto dal continuo), devo capire se devo chiudere la mia rampa.

Questo intervallo di aggregazione va spesso di pari passo con la tecnologia di monitoraggio. Le tecnologie di monitoraggio sono in alcuni casi anche in continuo, ma le informazioni vengono considerate aggregate e inviate a una centrale operativa in maniera aggregata, soprattutto le informazioni di sezione. Generalmente i periodi vanno nell'ordine dei minuti, il più classico è 5 minuti.

Quindi nel nostro script cosa facciamo, scegliamo una variabile, solitamente si può utilizzare la densità, noi non abbiamo direttamente la densità, abbiamo l'occupancy, quindi utilizziamo l'occupancy, in particolare utilizziamo l'occupancy aggregata, quella di sezione.

Quindi prendo il valore di occupancy aggregata di quello specifico intervallo, definisco un valore di soglia, ed essenzialmente metto il semaforo a rosso se l'occupancy aggregata supera quel valore soglia.

Dopodiché attivo una variabile di stato che chiamo *ramp_metering* che mi dice se quell'istante di simulazione (che va avanti secondo per secondo) il ramp metering è attivo o non attivo. Ma perché devo definire questa variabile di stato? Tecnicamente io finora gli ho detto: ogni 5 minuti essenzialmente metto se è attivo o non attivo il ramp metering, e poi nello stesso intervallo di aggregazione, cioè sempre ogni 5 minuti dovrei fare il controllo su dove si trova la coda sulla rampa. Però solitamente, soprattutto se i tempi di aggregazione sono di 5 minuti, chiudere una rampa nei momenti di picco per 5 minuti è troppo! Rischiamo veramente che si blocchi tutto in accesso alla rampa.

Quindi noi abbiamo bisogno di un secondo intervallo di aggregazione del tempo, che mi permetta di verificare le condizioni della rampa, e deve essere un intervallo di aggregazione molto più piccolo rispetto a quello della corsia principale.

Quindi in teoria questa variabile di stato *ramp_metering* mi permette di capire se la coda è rigurgitata troppo perché c'è il semaforo messo a rosso o perché c'è troppa fila.

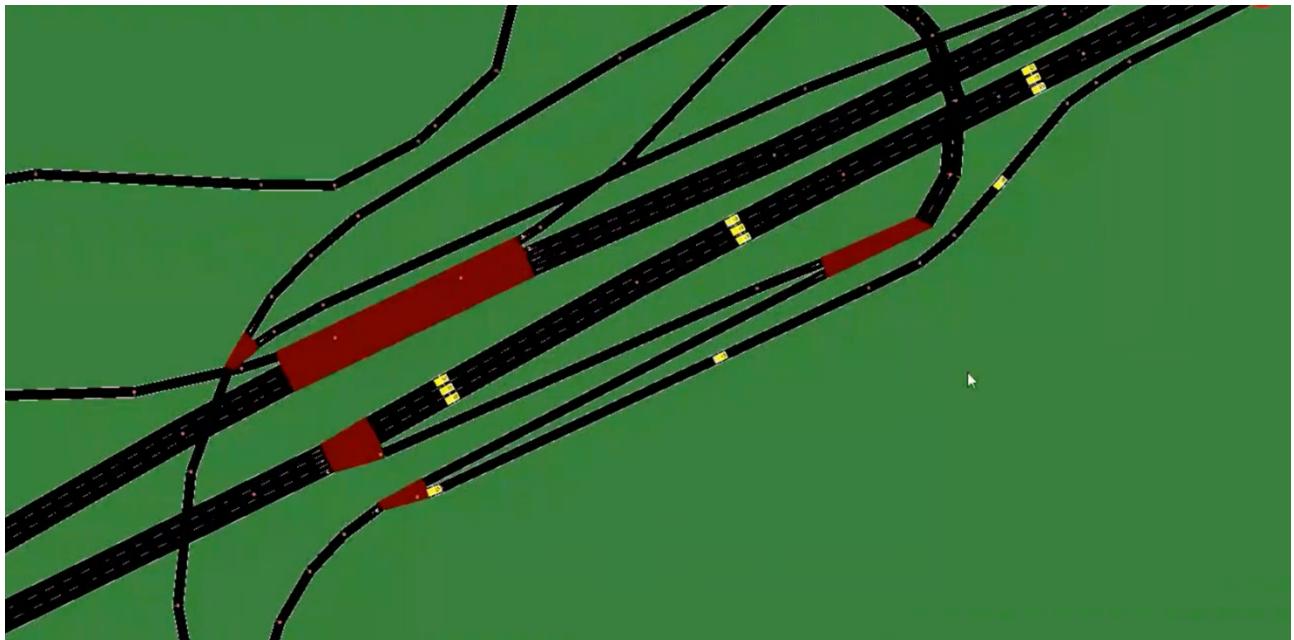
```
if nanmean(OCCUPANCY_AGG(contatore_intervalli_tre_minuti-1,:))>=30
    for d = 1:length(lista_semafori)
        traci.trafficlights.setRedYellowGreenState(lista_semafori{d}, 'r');
    end
    disp('GatingActivated')
    ramp_metering = 1;
    contatore = contatore +1;
else
```

Quindi io ho bisogno di un altro tempo di aggregazione, che io ho chiamato *tempo_valutazione_coda* che potrei settare a 1 minuto.

Ora il controllo sulla coda lo vado a fare quando il ramp metering è attivo, per cui metto un if *ramp_metering == 1*, perciò mi sono definito quella variabile prima.

Se la variabile di stato è pari a 1 allora si che io vado a valutare il valore di occupancy del detector che a me serve.

Ma io ho tanti detector sulla rampa, ma quello che a me interessa vedere è quello in prossimità dell'accesso alla rampa, poi se volessi essere più cautelativo potrei scegliere quelli che si trovano già nella rampa, ma questo sta esclusivamente nella scelta del progettista.



A me in questo caso l'unico detector che interessa è il 15.2_rampa; quindi, io vado a valutare il valore di occupancy rilevato da quel detector.

Anche qui vado a definire un valore di soglia, che fisso a 80/90%, se fissassi 100% significherebbe che il veicolo sarebbe stato fermo lì per 1 minuto, ma quindi si sarebbe creata della coda dietro di lui in quel minuto, quindi devo considerare un valore circa 90%.

Ovviamente se l'occupancy supera quel valore metto il semaforo a verde, e in questo caso dobbiamo considerare un transitorio, cioè non è che la coda si dissipa istantaneamente, ma in quel transitorio si potrebbe generare ulteriore coda. Quindi in questo caso è molto importante giocare con i valori soglia di occupancy ma anche sul detector che sceglieremo sulla rampa.

```

Editor - /Users/dario/Desktop/ramp_metering_scenari_2/script_RM.m *
Lettura_InductionLoop_postProcessed.m x lettura_inductionloop_RM_post_processed.m x script_RM.m * + x
49
50 %Controllare curva dopo il rosso semaforico per assicurarci che non
51 %crei problemi ad altri archi
52 if i == TLEvaluation_time*s
53     if ramp_metering == 1
54         for r = 1:length(rampa)
55             occupancy_evaluation = traci.inductionloop.getLastStepOccupancy(rampa{r});
56             if occupancy_evaluation >=80 %PERCENTUALE
57                 traci.trafficlights.setRedYellowGreenState(lista_semafori{r}, 'G');%nome semaforo
58                 ramp_metering=0
59             end
60         end
61         s = s+1;
62     else
63         s = s+1;
64     end
65 end
66 end
67

```

Facendo la cosiddetta *sumo gui* e andando a fare *run*, inizio a vedere che il semaforo è sempre verde.

Iniziano ad interfacciarsi i veicoli che vengono dalla rampa con i veicoli che si trovano sulla main lane, inizia a crearsi traffico, con un fenomeno di stop and go, fino a quando l'occupancy non supera la soglia prestabilita sulla sezione scelta della main lane, in corrispondenza di quell'evento il semaforo si mette a rosso sulla rampa.

Inizia a formarsi la coda e il semaforo rimane a rosso, fino a che la coda non raggiunge il detector in fondo alla rampa che rileva un'occupancy superiore al 90% e quindi setta il semaforo a verde, ma noi sappiamo che c'è un transitorio; quindi, osserviamo che la coda continua ad allungarsi fino a rigurgitare in una realtà diversa.

Quindi questo esperimento ci ha fatto comprendere che la sezione scelta sulla rampa, il detector scelto, non è quello corretto, perché è troppo prossimo a un punto che io non voglio che si blocchi, quindi dovrò scegliere un altro detector, ad esempio il 15.3_rampa, quindi nello script dovrei semplicemente andare a cambiare quel nome.

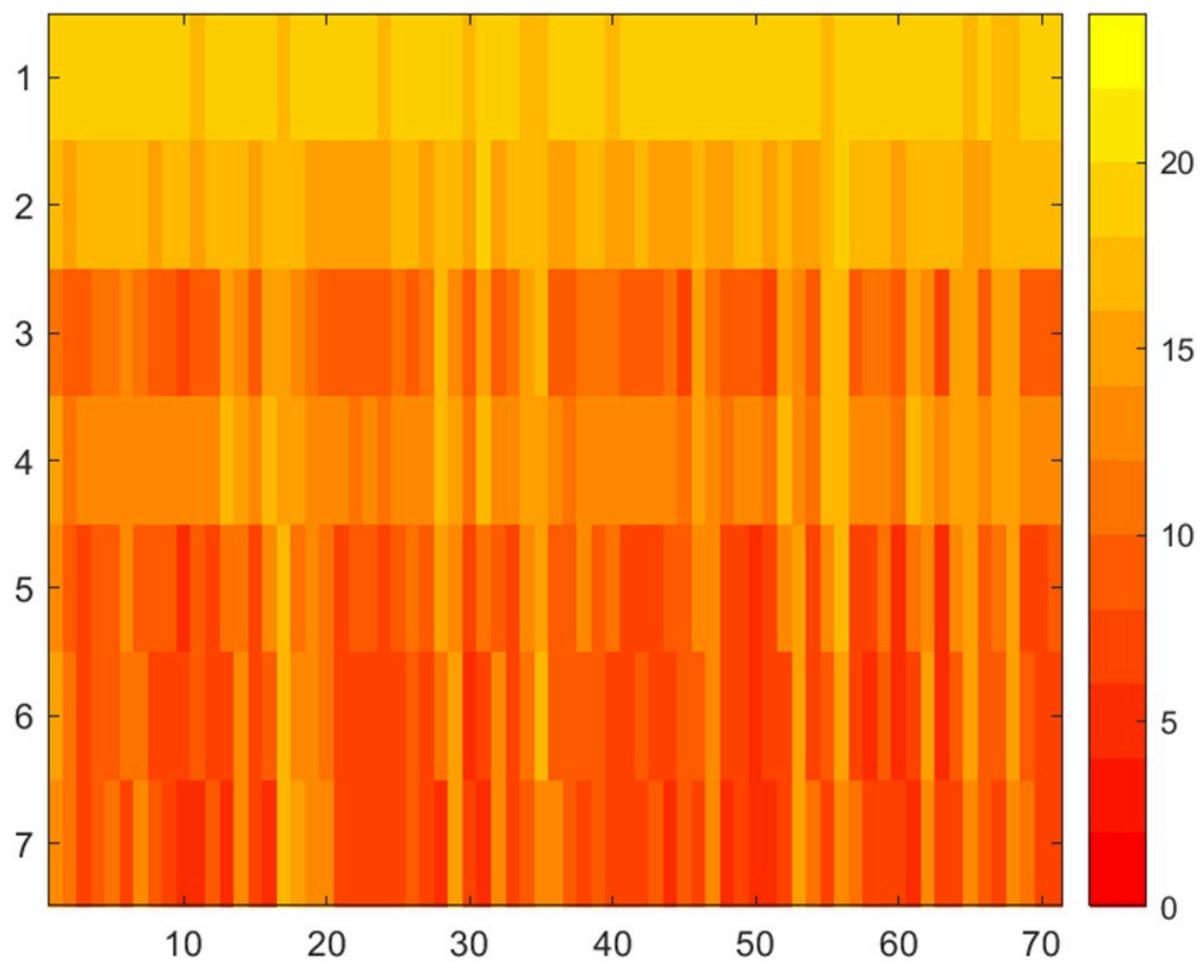
In questo scenario la costruzione del diagramma la facciamo sempre in post processing, cioè la simulazione su sumo con la logica di controllo di ramp metering mi riempie i file di testo, dopodiché faccio un'analisi post processing che mi genera i diagrammi con imagesc.

Noi sappiamo che la sezione 1 è la sezione dopo la rampa, infatti le condizioni di deflusso dopo un restringimento, dopo un bottleneck, sono condizioni quasi di free flow, dico quasi perché di fatti non hanno più conflitti ma dopo il bottleneck devono accelerare, infatti nel diagramma abbiamo colori sull'arancio chiaro, cioè valori di velocità intorno ai 50 km/h, diciamo che condizioni di free flow sono intorno a 75-80km/h, ma non vedo quei valori perché loro stanno appena accelerando, sto troppo vicino al bottleneck.

Mentre invece, le zone prima del bottleneck, le code rigurgitano verso dietro, perché il fenomeno di coda è: andare avanti nel tempo, indietro nello spazio.

Per lo scenario sul ramp metering vediamo un fenomeno che è interpretabile come la cosiddetta shock-wave, cioè vediamo degli spazi rossi e poi vediamo delle colonne, non colonne verticali ma leggermente oblique, di colori più chiari, perché di fatti si blocca tutto sulla corsia principale (zona rossa), chiudo la rampa, quindi incomincia a defluire il traffico sulla corsia principale e osservo zone più chiare, dopodiché una volta che si è tornati in condizioni di free flow riapro la rampa e man mano si arriva nuovamente a una situazione più congestionata e così via, questo fenomeno si traduce in delle diagonali di colore più chiaro che nella realtà possono essere interpretate come delle shock wave.

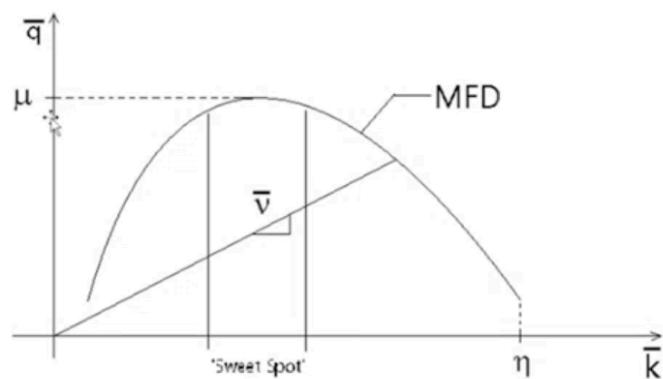
Anche in questo caso, il mio grado di libertà è dove scelgo il detector sulla rampa e in corrispondenza di quali valori di occupancy setto a verde piuttosto che a rosso il semaforo sulla rampa. Se sulla main lane scelgo come upper bound di occupancy un valore troppo alto, allora sto avvantaggiando i veicoli sulla rampa, perché il semaforo sarà settato a verde per molto tempo, per cui vedrò una situazione molto più congestionata. Se sulla rampa prendo un detector che si trova molto in prossimità del punto sul quale voglio evitare che arrivi la coda, a causa di un transitorio, la coda potrebbe arrivarcì, e così via...



MACROSCOPIC FUNDAMENTAL DIAGRAM

Lo scopo ultimo del macroscopic fundamental diagram non è altro che creare una relazione tra due variabili: l'accumulation e la production (che possono essere viste come delle proxy rispettivamente della densità e del flusso circolante nell'area).

Nella realtà, costruendo una curva di questo tipo, si possono applicare delle politiche di controllo di un'area di una rete, affinché quella specifica area rimanga nelle sue condizioni operative, cioè rimanga all'interno della cosiddetta sweet spot, cioè a cavallo dei valori massimi di production, quindi di flussi circolanti nella nostra area.



Quindi quello che andremmo a fare è inserire dei punti di monitoraggio sulla nostra rete, prendere le informazioni a livello di arco di densità e flusso, calcolare i valori medi dell'intera area, e sulla base di questi poter prendere essenzialmente delle decisioni.

Quindi, a differenza dell'esercitazione precedente, abbandoniamo il classico ambiente extraurbano, per andare a lavorare su un ambiente urbano. L'ambiente urbano che andiamo a considerare è un pezzo del centro storico di Nola e la domanda che abbiamo considerato è una domanda che vede quattro centroidi: i flussi hanno un solo centroide di uscita e hanno tre archi di ingresso.

Inoltre, così come nel ramp metering la chiusura la andavamo a simulare con un semaforo, anche in questo caso stiamo simulando il medesimo comportamento.

NB: in questo caso, dato che l'arco è bidimensionale, quello che noi dobbiamo andare a bloccare è solamente l'arco di ingresso, mentre l'arco di uscita non deve essere bloccato, quindi l'unica corsia bloccata è quella in ingresso.

Di base i semafori sono settati a verde e tramite la nostra logica di controllo sceglieremo se settarli a rosso o mantenerli a verde.

I punti gialli sono gli archi che noi stiamo andando a monitorare e che rappresentano una zona di traffico all'interno della quale io voglio mantenere le condizioni di deflusso all'interno della sweet spot.

Le operazioni che andremo a vedere sono:

- Calibrazione o definizione della curva MFD della nostra area, che è specifica per ogni area;
- Implementazione della logica di controllo;



Nell'esercitazione precedente abbiamo già utilizzato questo script che andava a leggersi in post processing i file di output dei detector per calcolare le variabili della nostra simulazione (il flusso, la velocità etc..).

```

Editor - /Users/dario/Desktop/esercitazione_MFD_3/MFD_per_Lezione/lettura_induction_loop_mfd.m
script_RM.m  lettura_inductionloop_RM_post_processed.m  lettura_induction_loop_mfd.m +  x
1 clear all
2 % close all
3
4 FILES = dir('*.txt');
5
6 zona_1 = [1 2 3];
7
8 for c = 1:length(FILES)
9     Dati_Simulazione = fileread (FILES(c).name);
10    k = strfind(Dati_Simulazione,'<interval');
11    j = strfind(Dati_Simulazione,'/>');
12    differenza = length(j)-length(k);
13    j (1:differenza) = [];
14    for d=1:length(j)
15        SingoloTimeStep=Dati_Simulazione(k(d):j(d));
16        valuesLim=find(SingoloTimeStep==''''');
17
18        Begin(d,c)=str2double(SingoloTimeStep(valuesLim(1)+1:valuesLim(2)-1));
19        End(d,c)=str2double(SingoloTimeStep(valuesLim(3)+1:valuesLim(4)-1));

```

Abbiamo visto anche essenzialmente come cambiare i valori di -1 della velocità con NaN, cioè per ogni valore di -1 interpretare il fatto che non fosse passato nessun veicolo.

```

28 indice = find(velocita == -1);
29 velocita(indice) = NaN;
30 indice = find(lunghezza == -1);
31 lunghezza(indice) = NaN;
32 lunghezza_archi = repmat (100,1,length(FILES));
33

```

Cosa abbiamo aggiunto a questo script? Noi vediamo che dalla formulazione del MFD di Daganzo e Geroliminis, i valori di sezione di flusso e di occupancy vengono moltiplicati per la lunghezza dell'arco a cui appartengono:

FORMULE DAGANZO/GEROLIMINIS

$$q^w = \frac{\sum_{i=1}^n q_i \cdot l_i}{\sum_{i=1}^n l_i} \quad k^u = \frac{\sum_{i=1}^n k_i \cdot l_i}{\sum_{i=1}^n l_i}$$

Quindi di base noi dovremmo andare a conoscere la lunghezza degli archi delle sezioni che stiamo andando a monitorare. Nella realtà è prassi comune, anche per facilitare l'elaborazione dei dati in fase di simulazione, ipotizzare che tutti gli archi abbiano la stessa lunghezza.

In questo caso, in questo script abbiamo creato un array, tramite l'operatore *repmat*, che non fa altro che ripetere il valore '100' in un vettore riga tante volte quanti sono i file '.txt' generati

dalla simulazione, ossia tante volte quanti sono i detector. Quindi in definitiva io otterrò un vettore riga di tutti 100, con 100 lunghezza dell'arco, tanti quante sono le sezioni monitorate.

32

```
lunghezza_archi = repmat (100,1,length(FILES));
```

Ora, quello che dobbiamo andare a fare, è il calcolo delle variabili q (flusso) e o (occupancy), ma noi sappiamo che in ambiente microsimulato l'occupancy e la densità sono legate da una costante k che amplifica il valore che noi leggiamo. Quindi noi l'andamento possiamo vederlo anche solo andandoci a calcolare i valori di occupancy e poi successivamente farli diventare densità con quella formula semplice:

```

37
38 %definizione qw
39 for g= 1:size(flusso,1)
40     qw (g,:) = ((flusso(g,:)).*lunghezza_archi);
41     qu (g,:) = ((flusso(g,:)));
42 end
43 for g= 1:size(flusso,1)
44     ow (g,:) = ((occupancy(g,:)).*lunghezza_archi);
45     ou (g,:) = ((occupancy(g,:)));
46 end
47 kw_m = sum(qw,2)/sum(lunghezza_archi);
48 qu_m = sum(qu,2)/sum(lunghezza_archi);
49 ow_m = sum(ow,2)/sum(lunghezza_archi);
50 ou_m = sum(ou,2)/sum(lunghezza_archi);
51
52 kw_m = ow_m*1000/5/100;
53 ku_m = ou_m*1000/5/100;
54 vu_m = qu_m./ku_m;

```

Quindi in definitiva questa parte di script è dedicata al calcolo del numeratore delle formule di Daganzo.

```

%definizione qw
for g= 1:size(flusso,1)
    qw (g,:) = ((flusso(g,:)).*lunghezza_archi);
    qu (g,:) = ((flusso(g,:)));
end

```

$$q^w = \frac{\sum_{i=1}^n q_i \cdot l_i}{\sum_{i=1}^n l_i}$$

```

for g= 1:size(flusso,1)
    ow (g,:) = ((occupancy(g,:)).*lunghezza_archi);
    ou (g,:) = ((occupancy(g,:)));
end

kw_m = ow_m*1000/5/100;
ku_m = ou_m*1000/5/100;
vu_m = qu_m./ku_m;

```

$$k^u = \frac{\sum_{i=1}^n k_i \cdot l_i}{\sum_{i=1}^n l_i}$$

Una volta che ho ottenuto i numeratori, posso andarli a dividere elemento per elemento per la sommatoria della lunghezza degli archi, che non è altro che uno scalare.

A questo punto ho q, ho k, posso andare a diagrammarli per iniziare ad ottenere un certo andamento della nostra area.

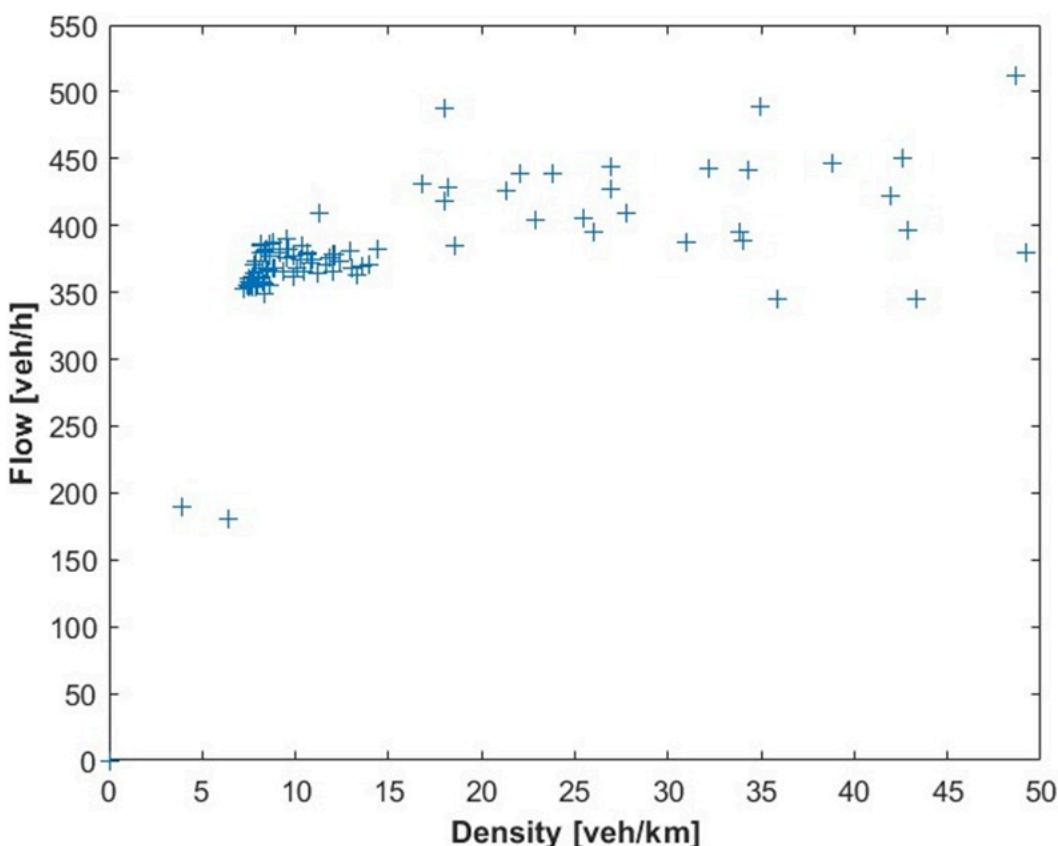
```

56     figure()
57     plot(kw_m,qw_m,'+')
58     ylim([0 550])
59     xlim([0 50])
60     xlabel('Density [veh/km]', 'FontWeight', 'bold')
61     ylabel('Flow [veh/h]', 'FontWeight', 'bold')

```

Nella realtà noi non vediamo troppo la curva di free flow perché abbiamo pochi punti nella nostra simulazione, però possiamo iniziare ad osservare un andamento di dispersione che ricorda l'andamento classico dell'MFD, ossia una parabola.

Quindi siamo in grado di individuare un valore critico di sweet spot, in questo caso non si vede benissimo il ramo instabile ma potremmo comunque ipotizzare che il valore di densità critico sia tra 20 e 30.



Quindi, una volta noto il valore di k che io voglio ottenere, allora posso andare a testare cosa accade nella mia simulazione quando io utilizzo un controllo di accesso (quindi un gating) alla nostra area.

Ora lo script del controllo è molto simile allo script sul ramp metering per la tangenziale: abbiamo la lista detector che ci dice gli id che abbiamo, la lista semafori che ci dice tutti i semafori che abbiamo nella nostra simulazione. Poi aggiungiamo un'ulteriore riga perché noi effettivamente non dobbiamo utilizzare tutti i detector sulla rete.

Nel nostro caso noi vogliamo che i veicoli che aspettano fuori dall'area non aspettino all'infinito, ma vogliamo che arrivati a una certa altezza di coda, indipendentemente da quelle

che sono le condizioni di deflusso osservate nella nostra area, il semaforo diventi verde. Quindi abbiamo un numero di detector che devono essere trattati in maniera diversa, motivo per cui andiamo a definire una variabile con riferimento al nome dei detector addetti non al monitoraggio delle condizioni di deflusso dell'area ma al monitoraggio delle CODE sugli archi soggetti al fenomeno di gating.

Abbiamo poi il tempo di aggregazione, settato a 3 minuti. Abbiamo definito una variabile contatrice che mi permette di sapere quanti intervalli di aggregazione sto utilizzando (e ne abbiamo due perché abbiamo due intervalli di aggregazione diversi all'accesso all'area e all'interno dell'area). Abbiamo una variabile di stato *gating attivato* che è una variabile booleana (0-1) e un'altra variabile che è il *flag disattivato*, ossia una variabile di stato che mi permette di operare su un arco alla volta, quindi è una variabile di stato dell'arco i-esimo che io posso controllare. Nel nostro caso, visto che gli archi di ingresso sono 3, avremo 3 variabili flag.

```
3 [traciVersion sumoVersion]= traci.init(8873);
4 tutto_assieme_occ = [];
5 tempo_simulazione=18300;
6 lista_detector = traci.inductionloop.getIDList();
7 lista_semafori = traci.trafficlights.getIDList();
8 lista_detector_semafori = lista_detector(:,20:22);
9 tempo_aggregazione = 180;
10 contatore_intervalli_tre_minuti = 1;
11 flag_disattivato_1 = 0;flag_disattivato_2 = 0;flag_disattivato_3 = 0;
12 contatore_intervalli_20_secondi=1;
13 TLEvaluation_time = 20; %60
14 gating_attivato = 0; contatore = 0;
15
16
```

A questo punto non mi rimane che iniziare con la mia logica di controllo per quanto riguarda questa parte MFD, fissando un valore critico di densità pari a 15, non appena questo valore viene superato io blocco tutti gli accessi all'area.

La mia logica è molto semplice: non mi interessa operare per coppia o/d, ma chiudo tutto. Quindi vado in un ciclo *for* dove vado a leggere la lista dei semafori e imposto ogni semaforo a rosso. Ovviamente se mettiamo tutti i semafori a rosso la variabile di stato *gating attivo* viene settata a 1.

Un altro aspetto è che se questa variabile di densità scende sotto al valore di 15, allora entro in un altro ciclo *for* in cui leggo la lista dei semafori e li setto tutti a verde, quindi riapro tutti gli accessi all'area. Ovviamente ponendo la variabile di gating, a quel punto, a 0.

Con la variabile contatore posso vedere quante volte si è attivato il gating.

```

37         if sum(OCCUPANCY_AGG_MFD(contatore_intervalli_tre_minuti-1,:).*lunghezza_archi)...
38             /sum(lunghezza_archi) >=15
39             for d = 1:length(lista_semafori)
40                 traci.trafficlights.setRedYellowGreenState(lista_semafori{d}, 'r');
41             end
42             disp('GatingActivated')
43             gating_attivato = 1;
44             contatore = contatore +1;
45         else
46             for d = 1:length(lista_semafori)
47                 traci.trafficlights.setRedYellowGreenState(lista_semafori{d}, 'G'); %else
48             end
49             disp('GatingDisactivated')
50             gating_attivato = 0;
51         end
52     end

```

Effettivamente però abbiamo lo stesso problema che abbiamo riscontrato nel caso della ramp metering sulla tangenziale, ovvero che se la coda agli accessi cresce troppo, io devo necessariamente riaprire i miei archi, a prescindere dalle condizioni di deflusso all'interno dell'area.

Quindi, se e solo se il gating è attivo, posso andare a vedere il valore di occupancy rilevato dal detector relativo al semaforo i-esimo. Se questo valore è maggiore di un valore soglia, che poniamo a 80, allora quel semaforo viene settato a verde.

In realtà c'è un *else if* perché quando io metto a verde il semaforo, quando si è dissipata la maggior parte della coda, io posso richiederlo nuovamente. Per questo abbiamo bisogno dei *flag* di stato del semaforo che mi controlla il gating.

Questa logica la applico per tutti i semafori, cioè per tutti gli archi di accesso all'area.

```

Editor - /Users/dario/Desktop/esercitazione_MFD_3/MFD_per_Lezione/script_finale_gating.m
+1 lettura_inductionloop_RM_post_processed.m × lettura_induction_loop_mfd.m × script_finale_gating.m × +
1
56 %CREA PROBLEMI AD ALTRI ARCHI
57 if i == TLEvaluation_time*contatore_intervalli_20_secondi
58     if gating_attivato == 1
59         for r = 1:length(lista_detector_semafori)
60             occupancy_evaluation(i,r) =traci.inductionloop.getLastStepOccupancy(lista_detector_
61             if occupancy_evaluation(i,r) >=80 & r==1 %PERCENTUALE
62                 traci.trafficlights.setRedYellowGreenState(lista_semafori{r}, 'G');%nome semafor
63                 flag_disattivato_1 = 1;
64             elseif occupancy_evaluation(i,r) <=20 & r==1 && flag_disattivato_1 == 1
65                 traci.trafficlights.setRedYellowGreenState(lista_semafori{r}, 'r');%nome semafor
66                 flag_disattivato_1 = 0;
67             end
68             if occupancy_evaluation(i,r) >=80 & r==2 %PERCENTUALE
69                 traci.trafficlights.setRedYellowGreenState(lista_semafori{r}, 'G');%nome semafor
70                 flag_disattivato_2 = 1;
71             elseif occupancy_evaluation(i,r) <=20 & r==2 && flag_disattivato_2 == 1
72                 traci.trafficlights.setRedYellowGreenState(lista_semafori{r}, 'r');%nome semafor
73                 flag_disattivato_2 = 0;
74             end

```

Ora però come possiamo valutare se il nostro gating ha migliorato la situazione? Un parametro che ci può dire se la nostra logica è andata a buon fine o meno è la *production*, cioè il numero di veicoli che mediamente escono dalla nostra area.

Noi abbiamo detto che i centroidi di ingresso nella nostra rete sono tre, ma l'uscita è una sola; quindi posso monitorare i flussi in uscita, valutarli con il gating e senza gating, e fare le opportune valutazioni diagrammandoli e vedendo se i flussi in uscita col gating sono mediamente più elevati rispetto a quelli senza gating.

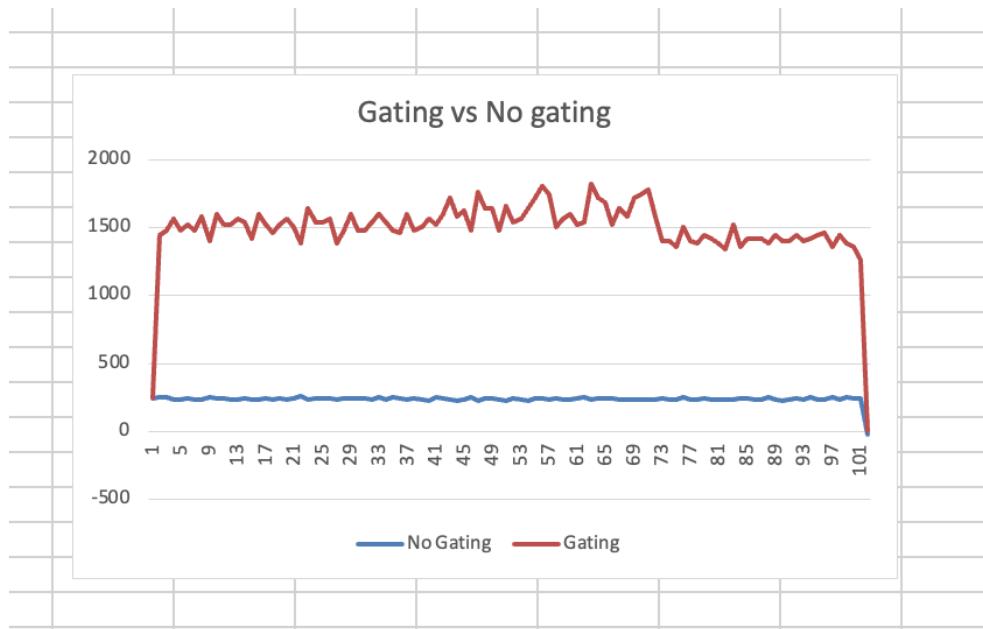
Banalmente cosa possiamo fare, a fine simulazione possiamo aprire in ambiente excel il file .txt relativo al detector posizionato in corrispondenza dell'uscita della rete:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	begin	end	id	VehContr	flow	occupancy	speed	onicMeanS	length	VehEntered		No Gating	Gating
2	0	180	z_uscita_1	12	240	2,46	12,27	12,19	4,5	12		243,8	240
3	180	360	z_uscita_1	72	1440	14,36	12,59	12,54	4,5	72		250,8	1440
4	360	540	z_uscita_1	74	1480	14,51	12,8	12,75	4,5	74		255	1480
5	540	720	z_uscita_1	78	1560	16,31	12,05	11,96	4,5	78		239,2	1560
6	720	900	z_uscita_1	74	1480	15,67	11,97	11,81	4,5	74		236,2	1480
7	900	1080	z_uscita_1	76	1520	15,38	12,42	12,36	4,5	76		247,2	1520
8	1080	1260	z_uscita_1	74	1480	15,83	11,8	11,69	4,5	74		233,8	1480
9	1260	1440	z_uscita_1	79	1580	16,61	12,02	11,89	4,5	79		237,8	1580
10	1440	1620	z_uscita_1	70	1400	13,86	12,67	12,63	4,5	70		252,6	1400
11	1620	1800	z_uscita_1	80	1600	16,24	12,37	12,32	4,5	80		246,4	1600
12	1800	1980	z_uscita_1	76	1520	15,54	12,32	12,23	4,5	76		244,6	1520
13	1980	2160	z_uscita_1	76	1520	16,41	11,69	11,58	4,5	76		231,6	1520
14	2160	2340	z_uscita_1	78	1560	16,64	11,82	11,72	4,5	78		234,4	1560
15	2340	2520	z_uscita_1	76	1520	15,9	12,12	12,04	4,5	77		240,8	1540
16	2520	2700	z_uscita_1	72	1440	15,4	11,73	11,6	4,5	71		232	1420
17	2700	2880	z_uscita_1	80	1600	16,94	11,98	11,8	4,5	80		236	1600
18	2880	3060	z_uscita_1	76	1520	15,43	12,36	12,31	4,5	76		246,2	1520
19	3060	3240	z_uscita_1	73	1460	15,37	11,96	11,87	4,5	73		237,4	1460
20	3240	3420	z_uscita_1	76	1520	15,64	12,24	12,15	4,5	76		243	1520
21	3420	3600	z_uscita_1	78	1560	16,83	11,74	11,59	4,5	78		231,8	1560
22	3600	3780	z_uscita_1	75	1500	15,27	12,37	12,28	4,5	75		245,6	1500
23	3780	3960	z_uscita_1	69	1380	13,34	12,97	12,93	4,5	69		258,6	1380
24	3960	4140	z_uscita_1	81	1620	17,55	11,65	11,59	4,5	82		231,8	1640
25	4140	4320	z_uscita_1	78	1560	16,12	12,13	12,03	4,5	77		240,6	1540
26	4320	4500	z_uscita_1	77	1540	15,91	12,2	12,1	4,5	77		242	1540
27	4500	4680	z_uscita_1	78	1560	16,2	12,11	12,04	4,5	78		240,8	1560
28	4680	4860	z_uscita_1	69	1380	14,87	11,77	11,6	4,5	69		232	1380
29	4860	5040	z_uscita_1	74	1480	15,36	12,16	12,05	4,5	74		241	1480
30	5040	5220	z_uscita_1	80	1600	16,34	12,31	12,24	4,5	80		244,8	1600
31	5220	5400	z_uscita_1	74	1480	15,11	12,34	12,25	4,5	74		245	1480
32	5400	5580	z_uscita_1	74	1480	15,29	12,2	12,1	4,5	74		242	1480
33	5580	5760	z_uscita_1	77	1540	16,08	12,08	11,97	4,5	77		239,4	1540
34	5760	5940	z_uscita_1	80	1600	15,94	12,59	12,55	4,5	80		251	1600
35	5940	6120	z_uscita_1	76	1520	16,11	11,9	11,83	4,5	77		236,6	1540
36	6120	6300	z_uscita_1	75	1500	14,94	12,57	12,51	4,5	74		250,2	1480
37	6300	6480	z_uscita_1	73	1460	14,72	12,5	12,4	4,5	73		248	1460
38	6480	6660	z_uscita_1	80	1600	16,93	11,91	11,81	4,5	80		236,2	1600
39	6660	6840	z_uscita_1	74	1480	15,16	12,31	12,2	4,5	74		244	1480
40	6840	7020	z_uscita_1	75	1500	15,72	12,04	11,93	4,5	75		238,6	1500
41	7020	7200	z_uscita_1	77	1540	17,14	11,42	11,32	4,5	78		226,4	1560
42	7200	7380	z_uscita_1	77	1540	15,09	12,68	12,64	4,5	76		252,8	1520
43	7380	7560	z_uscita_1	79	1580	16,48	12,14	12,07	4,5	80		241,4	1600
44	7560	7740	z uscita 1	87	1740	18,17	11,99	11,89	4,5	86		237,8	1720

Pronto Accessibilità: verifica

Possiamo a questo punto calcolare il flusso come il parametro *numero di veicoli* per 3600 diviso l'intervallo di aggregazione, che è 180 secondi, perché noi aggregiamo le informazioni ogni 3 minuti.

Andando a incolonnare queste informazioni di flusso nelle colonne di *gating* e *no gating*, possiamo andare a creare un grafico a serie e dal banale confronto delle due serie possiamo osservare che la linea relativa alla situazione di *gating* si trovi sempre superiormente rispetto alla linea relativa alla situazione senza *gating*.



Inoltre è possibile calcolare la variazione percentuale rispetto alle condizioni senza gating e andare a diagrammare queste percentuali di miglioramento.

Immaginando di voler capire qual è il miglior parametro per ottenere l'incremento maggiore potremmo fare tante simulazioni in cui cambiamo il valore di densità in corrispondenza del quale vogliamo il maggior valore di production.

