ANALISI DEI SISTEMI DI PRODUZIONE E MODELLI MATEMATICI PER L'OPERAZIONE DI ASSEMBLAGGIO

INTRODUZIONE

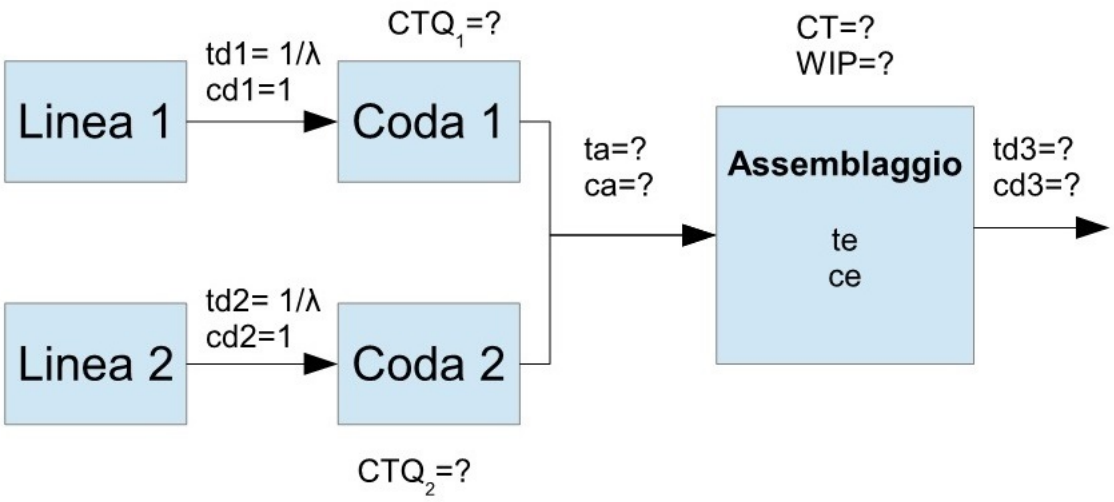
Obiettivo di questo lavoro è la modellizzazione dei parametri di una stazione di assemblaggio di due componenti complementari, prodotti a monte da due linee aventi tempi di interuscita descritti dalla medesima distribuzione esponenziale.

La semplicità del setting iniziale scelto, avente un ristretto numero di variabili, consente di evidenziare alcune dinamiche e problematiche presenti anche in sistemi più complessi e a valutare soluzioni generalmente applicabili.

La scelta di un'eguale distribuzione esponenziale descrivente i tempi di interuscita di entrambe le linee a monte ci permette di riflettere sui risultati inaspettati che questo comporta.

Il tema dell'assemblaggio viene ripetutamente richiamato in volumi quali Factory Physics ([Hopp W.J., Spearman M.L.]-Factory Physics, second edition-McGrawHill), Stochastic Models Of Manufactruing Systems ([Buzacott.JA,Shabthikumar.JG]-Stochastic Models Of Manufactruing Systems-1993-PrenticeHall) e Foundations Of Inventory Management ([Zipkin.PH]-Foundations Of Inventory Management-2009-McGrawHill) senza mai scendere nei dettagli. Da qui la curiosità, da qui il lavoro.

DESCRIZIONE e PRIME CONSIDERAZIONI



Come si evince dall'immagine, entrambe le linee a monte presentano un flusso di uscita dei componenti da loro prodotti caratterizzato dalla medesima distribuzione di probabilità esponenenziale, la cui formula generica è:

$$f(x) = \lambda e^{-\lambda x}$$

Questa è caratterizzata da un unico parametro λ che ne determina la media $(1/\lambda)$ e la varianza $(1/\lambda^2)$.

Dimostrazione del calcolo della media:

$$E[x] = \int_0^{+\infty} x\lambda e^{-\lambda x} dx = \int_0^{+\infty} e^{-\lambda x} dx = \frac{1}{\lambda}$$

Dimostrazione del calcolo della varianza:

$$VAR[X] = E\left[x^2\right] - E[x]^2$$

$$E\left[x^2\right] = \int_0^{+\infty} x^2 \lambda e^{-\lambda x} dx = 2\int_0^{+\infty} x e^{-\lambda x} dx =$$

$$= \frac{2}{\lambda}\int_0^{+\infty} e^{-\lambda x} dx = \frac{2}{\lambda^2}$$

$$E[x]^2 = \left(\frac{1}{\lambda}\right)^2 = \frac{1}{\lambda^2}$$

$$VAR[X] = E\left[x^2\right] - E[x]^2 =$$

$$= \frac{2}{\lambda^2} - \frac{1}{\lambda^2} = \frac{1}{\lambda^2}$$

La distribuzione della probabilità cumulata della distribuzione esponenziale è:

$$F(x) = \mathbb{P}[X \leq x] = \int_0^x \lambda e^{-\lambda x} dx = 1 - e^{-\lambda x}$$

Caratteristica peculiare di questa distribuzione è l'assenza di memoria (memoryless property), ovvero l'ininfluenza degli eventi passati sui valori attesi futuri.

Dimostrazione:

$$E\left[x - y / y \in \mathbb{R}^+\right] = \int_y^{+\infty} (x-y)\lambda e^{-\lambda(x-y)} dx =$$

$$= \int_y^{+\infty} x\lambda e^{-\lambda(x-y)} dx - \int_y^{+\infty} y\lambda e^{-\lambda(x-y)} dx =$$

$$= ye^{-\lambda y}e^{\lambda y} + e^{\lambda y}\int_y^{+\infty} e^{-\lambda x} dx - ye^{\lambda y}\int_y^{+\infty} \lambda e^{-\lambda x} dx =$$

$$= y + \frac{1}{\lambda} e^{\lambda y} e^{-\lambda y} - y e^{-\lambda y} e^{\lambda y} = y + \frac{1}{\lambda} - y = \frac{1}{\lambda}$$

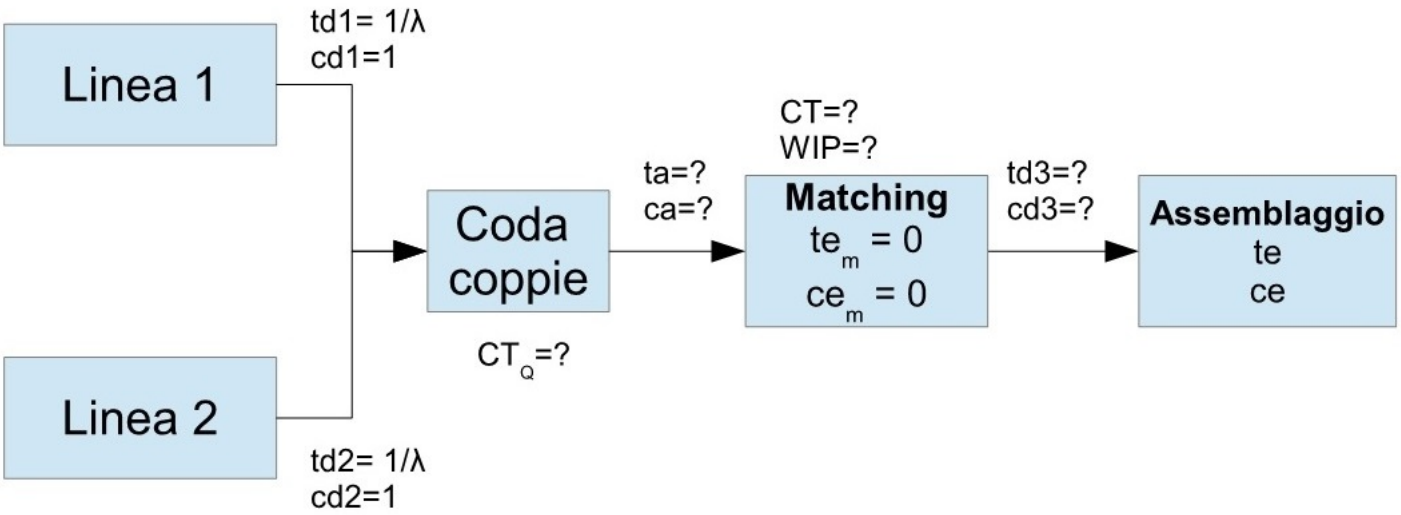
$$E\left[x - y / y \in \mathbb{R}^{+}\right] = \frac{1}{\lambda} = E[x]$$

Come si può osservare, in entrambi i casi il valore atteso è $\frac{1}{\lambda}$.

Osservando l'immagine 1 si palesa immediatamente l'inapplicabilità, in questo contesto, delle formule standard presenti in Factory Physics riguardanti la gestione delle code. Pur essendo parallele, le due linee trattate producono componenti diversi e complementari, mentre la formula M\M\2 richiede su entrambe la produzione di un identico articolo.

Dovranno essere studiati strumenti specifici per questo caso.

A tal fine risulta utile scorporare la stazione di assemblaggio in due sottostazioni in cascata aventi compiti differenti. La prima, che chiameremo stazione di accoppiamento o di matching, sarà il luogo dove verranno a formarsi i pacchetti fisicamente lavorati nella seconda, che definiremo stazione di assemblaggio. Così facendo si riducono ulteriormente il numero di variabili e dinamiche coinvolte in quanto, isolando le lavorazioni fisiche nella seconda stazione gestibile con una normale formula per le code G\G\1, il comportamento della stazione di matching sarà influenzato solo dai flussi di rifornimento delle linee 1 e 2.



La prime domande a cui dare risposta sono:

1. Quale è l'unità oggetto di analisi? E' il singolo componente o la coppia?
2. Come definire concettualmente i tempi di interarrivo, di attesa e di interuscita della stazione? Quali eventi assumeranno il ruolo di inizio e fine misurazione?
3. Da quale distribuzione di probabilità saranno descritti? Quale sarà la loro media e la loro varianza?
4. Quali sono i livelli di WIP?

La prima domanda ha una risposta obbligata: la coppia. Questa sarà l'unità base dell'analisi che effettueremo. Gli intertempi in ingresso ed in uscita, i tempi di attesa e i livelli di WIP faranno tutti riferimento ad essa.

Più complessa è la risposta al secondo quesito, riguardante la natura degli eventi costituenti i limiti delle misurazioni cronometriche. E' utile allo scopo visualizzare la coppia come l'insieme di due entità

che arriveranno in momenti diversi alla stazione di matching. Il primo a giungere a destinazione verrà definito «primo all'ingresso» mentre il secondo come «utimo all'ingresso», rispettivamente abbreviati in FTA e LTA.

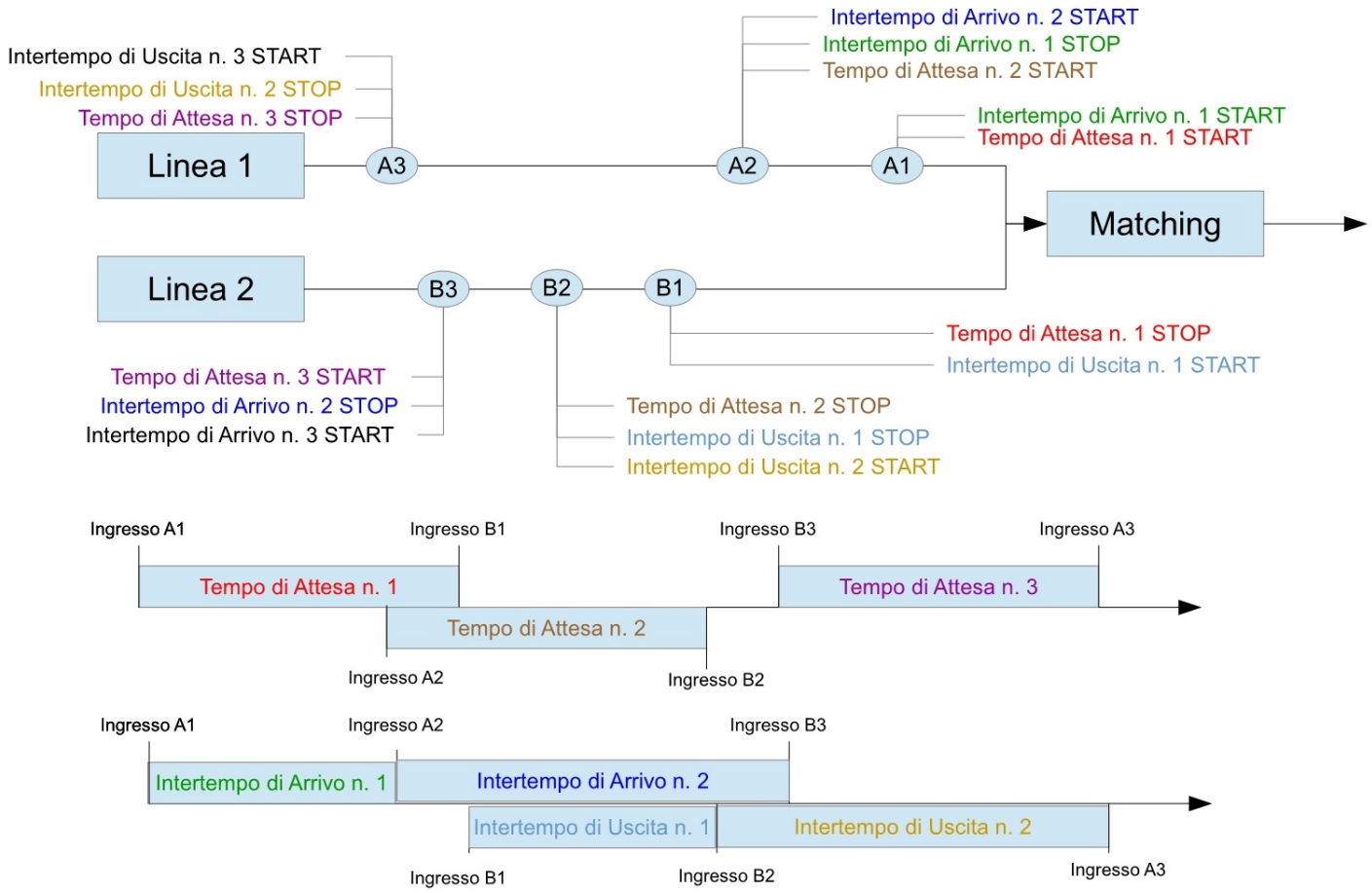
Si stabilisce che l'ingresso alla stazione di matching del FTA di una coppia, successiva alla prima, appartenente ad un generico lotto di produzione, inneschi i seguenti eventi:

- fine della misurazione dell'intertempo in ingresso della coppia di appartenenza;
- inizio della misurazione dell'intertempo in ingresso della coppia successiva a quella di appartenenza;
- inizio della misurazione del tempo di attesa della coppia di appartenenza.

Conseguente, l'ingresso alla stazione del LTA della medesima coppia di cui sopra, indicherà:

- fine della misurazione del tempo di attesa della coppia di appartenenza;
- fine della misurazione dell'intertempo di uscita della coppia di appartenenza;
- inizio della misurazione dell'intertempo di uscita della coppia successiva a quella di appartenenza.

Descrizione	Istante iniziale	Istante finale
Intertempo in ingresso	Ingresso alla stazione di FTA della coppia precedente	Ingresso alla stazione di FTA della coppia attuale
Intertempo In uscita	Ingresso alla stazione di LTA della coppia precedentemente	Ingresso alla stazione di LTA della coppia attuale
Tempo di attesa	Ingresso alla stazione di FTA della coppia attuale	Ingresso alla stazione di LTA della coppia attuale



Occorre soffermarsi su alcune considerazioni:

- due FTA appartenti a due coppie consecutive saranno provenienti dalla stessa linea a monte solo se, nel tempo intercorrente tra i loro ingressi, l'altra linea non avrà rilasciato un numero di componenti pari all'ammontare di elementi in coda alla stazione di matching più uno;
- a seguito dello scorporo delle operazioni fisiche di assemblaggio nell'omologa stazione, all'interno della stazione di matching l'istante di ingresso del LTA di una coppia rappresenterà anche l'istante di rilascio della coppia dalla stessa, non essendoci altre tempistiche di cui tener conto;
- gli LTA dei componenti in coda potranno giungere alla stazione di matching solo nel medesimo ordine dei propri FTA;
- avendo la stazione di matching una natura particolare, il tempo di attesa intercorrente tra gli ingressi di FTA e LTA di una coppia rappresenta anche il tempo di «lavorazione».

Per trovare le risposte alle altre domande dovremo proseguire nell'analisi.

ANALISI DEL SISTEMA

Tempi di attesa

Si consideri la linea in uno stato in cui la stazione di matching, avente al momento una coda nulla, sia pronta a ricevere il primo FTA. Quanto tempo si dovrà attendere?

Si definisca:

- N,numero di coppie facente parte del lotto preso in considerazione;
- LX, linea dedicata alla produzione del componente X;
- 'x_i' , variabile indicante il tempo di interuscita dell'i-esimo componente prodotto da LX, con $0 \leq i \leq N-1$;
- LY, linea dedicata alla produzione del componente Y;
- 'y_i', variabile indicante il tempo di interuscita dell'i-esimo componente prodotto da LY, con $0 \leq i \leq N-1$;
- 't_i', variabile indicante il tempo di interarrivo dell'i-esima coppia alla stazione di matching, , con $0 \leq i \leq N-1$;
- 'z_i', variabile indicante il tempo atteso per il proprio complementare all'interno della stazione di matching dal FTA della coppia $(i-1)$,, con $1 \leq i \leq N$.

Si potrà calcolare il tempo di interarrivo atteso medio della prima coppia nel modo seguente:
$E[t_0] = \int_0^{+\infty} x\lambda e^{-\lambda x} \int_x^{+\infty} \lambda e^{-\lambda y} dy dx + \int_0^{+\infty} y\lambda e^{-\lambda y} \int_y^{+\infty} \lambda e^{-\lambda x} dx dy =$
$= \int_0^{+\infty} x\lambda e^{-\lambda x} e^{-\lambda x} dx + \int_0^{+\infty} y\lambda e^{-\lambda y} e^{-\lambda y} dy = \int_0^{+\infty} x\lambda e^{-2\lambda x} dx + \int_0^{+\infty} y\lambda e^{-2\lambda y} dy =$
$= \frac{1}{2}(\int_0^{+\infty} e^{-2\lambda x} dx + \int_0^{+\infty} e^{-2\lambda y} dy) = \frac{1}{2}(\frac{1}{2\lambda} + \frac{1}{2\lambda}) = \frac{1}{2\lambda}$
Il valore di t_0 sarà pari al minore tra quelli di x_0 e y_0 (pari a x_0 quando $x_0 \leq y_0$, a y_0 altrimenti). I due integrali, addendi della formula precedente, rappresentano rispettivamente il valore medio di x_0, pesato nell'area in cui questa è minore di y_0, e di y_0, pesato nell'area in cui vige il caso contrario.

Si dovrà quindi quantificare il tempo di attesa medio della prima coppia, ovvero l'arco temporale intercorrente tra l'arrivo dei relativi FTA e LTA alla stazione di matching.

Allo scopo, sarà utile definire l'integrale che ne descrive la probabilità cumulata e successivamente derivarlo per ottenenerne la distribuzione di densità di probabilità, da cui si potrà ricavare il valore medio atteso e la varianza.
$\mathbb{P}[Z \leq z_1] = \int_0^{+\infty} \lambda e^{-\lambda x} \int_x^{z_1+x} \lambda e^{-\lambda y} dy dx + \int_0^{+\infty} \lambda e^{-\lambda y} \int_y^{z_1+y} \lambda e^{-\lambda x} dx dy =$
$= \int_0^{+\infty} \lambda e^{-\lambda x}(e^{-\lambda x} - e^{-\lambda x} e^{-\lambda z_1}) dx + \int_0^{+\infty} \lambda e^{-\lambda y}(e^{-\lambda y} - e^{-\lambda y} e^{-\lambda z_1}) dy =$
$= (1 - e^{-\lambda z_1})(\int_0^{+\infty} \lambda e^{-2\lambda x} dx + \int_0^{+\infty} \lambda e^{-\lambda y} dy = (1 - e^{-\lambda z_1})(\frac{1}{2} + \frac{1}{2}) \Longrightarrow$
$\Longrightarrow \mathbb{P}[Z \leq z_1] = (1 - e^{-\lambda z_1})$
Derivando si ottiene:

$$f(z_1) = \frac{d\mathbb{P}[Z \leq z_1]}{dz_1} = \lambda e^{-\lambda z_1}$$

Il valore atteso di z_1 sarà:

$$E[z_1] = \frac{1}{\lambda}$$

Si dovranno quindi modellizzare le coppie successive.

Nel caso in cui la X della coppia precedente a quella in analisi fosse FTA, lo sarà nuovamente anche nella coppia attuale solo nel caso in cui arrivi alla stazione di matching antedecemente al sopraggiungere alla

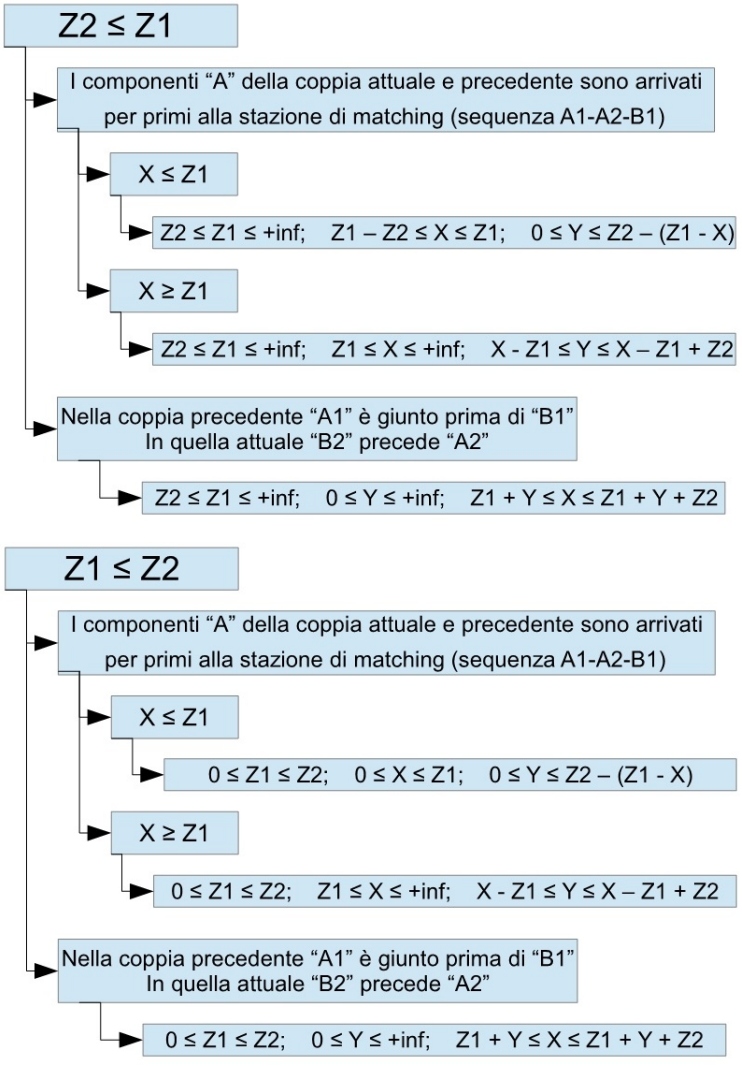
stessa di una quantità di complementari (prodotti da LY) superiore di un'unità al numero di arretrati in coda ($x_i < y_i + z_i$,), mentre Y sarà FTA nel caso contrario. Discorso analogo ma opposto nel caso in cui Y fosse FTA della coppia precedente.

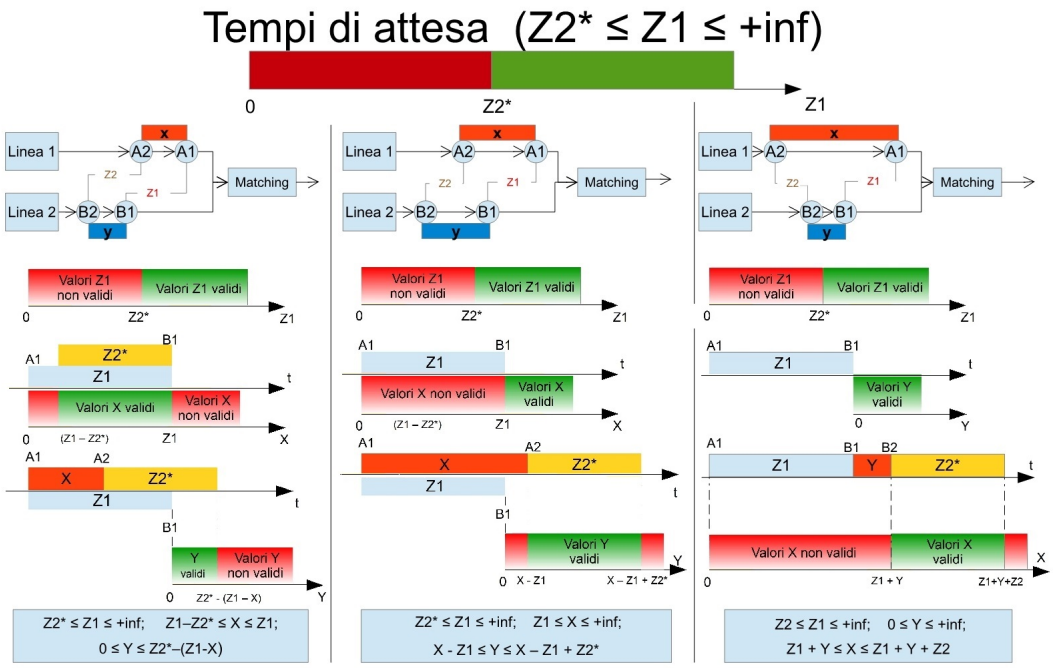
Da notarsi che:

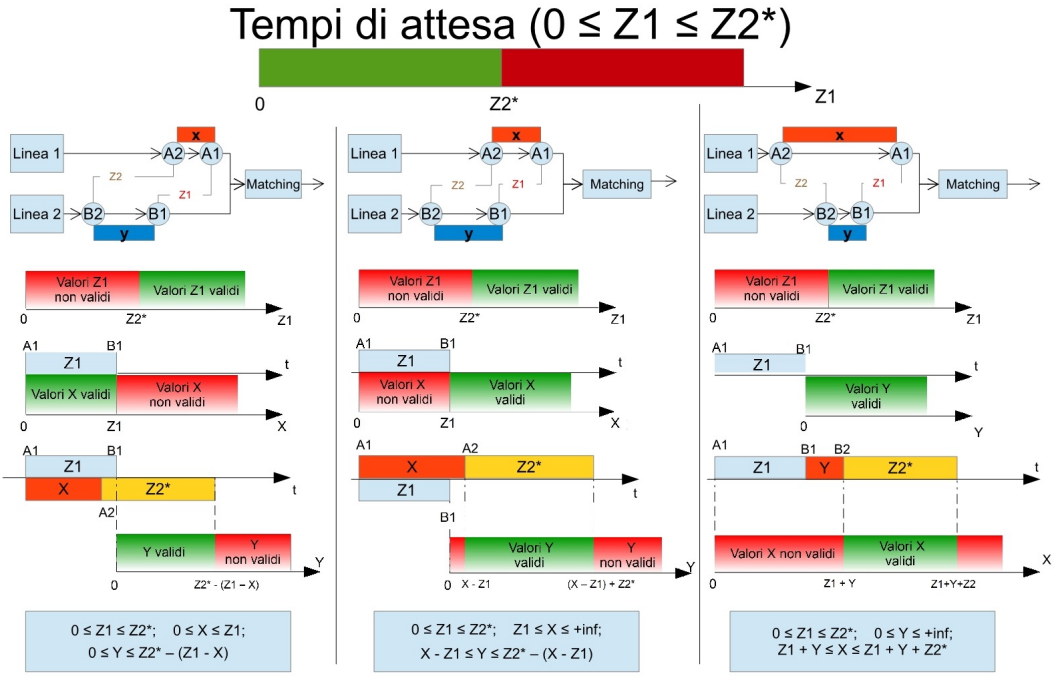
- essendo la stazione di matching gestita con una strategia FIFO, il tempo di attesa z_i conterrà al suo interno tutti i ritardi pregressi non ancora smaltiti;
- per qualsiasi i avremo x_i e y_i sempre caratterizzate dalle rispettive funzione esponenziale gemelle ($\overline{x}_i = \overline{x}_j = \overline{y}_i = \overline{y}_j$per ogni $i, j \in [0, N-1]$)
- non ci saranno due coppie aventi la stessa distribuzione di probabilità dell'intertempo di ingresso, in quanto influenzata dallo stato pregresso del sistema (rappresentato da z_i) e quindi specifico per ogni singola coppia;
- impossibilità di modellizzare gli intertempi di ingresso senza una previa definizione dei tempi di attesa.

Essendo il tempo di attesa z_{i+1} della coppia i inflenzato solo da z_i, x_i, y_i, è ipotizzabile l'esistenza di un algoritrmo possa calcolare tutti gli z_i.

Per ogni coppia successiva alla prima avremo (vedi immagine successiva):







Per il calcolo della probabilità cumulata dei tempi di attesa si avranno dodici integrali, di cui sei inerenti il caso in cui X sia stato FTA della coppia precedente (50% di probabilità) e altrettanti nel caso contrario.

Essendoci una totale specularità, sarà sufficiente sviluppare solo quelli specifici del primo caso, in quanto i rimanenti daranno dei risultati identici.

Si avranno:

$$(1) = \int_{z_*}^{+\infty} f(z) \int_{z-z_*}^{z} \lambda e^{-\lambda x} \int_{0}^{z_*-z+x} \lambda e^{-\lambda y} dy dx dz$$

$$(2) = \int_{z_*}^{+\infty} f(z) \int_{z}^{+\infty} \lambda e^{-\lambda x} \int_{x-z}^{x-z+z^*} \lambda e^{-\lambda y} dy dx dz$$

$$(3) = \int_{z_*}^{+\infty} f(z) \int_{0}^{+\infty} \lambda e^{-\lambda y} \int_{y+z}^{y+z+z^*} \lambda e^{-\lambda x} dx dy dz$$

$$(4) = \int_{0}^{z_*} f(z) \int_{0}^{z} \lambda e^{-\lambda x} \int_{0}^{z_*-z+x} \lambda e^{-\lambda y} dy dx dz$$

$$(5) = \int_{0}^{z_*} f(z) \int_{z}^{+\infty} \lambda e^{-\lambda x} \int_{x-z}^{x-z+z^*} \lambda e^{-\lambda y} dy dx dz$$

$(6) = \int_0^{z_*} f(z) \int_0^{+\infty} \lambda e^{-\lambda y} \int_{y+z}^{y+z+z^*} \lambda e^{-\lambda x} dxdydz$

Riunendo alcuni di questi integrali, si definiscono:

$$(A) = (1) = \int_{z_*}^{+\infty} f(z) \int_{z-z_*}^{z} \lambda e^{-\lambda x} \int_0^{z_*-z+x} \lambda e^{-\lambda y} dydxdz$$

$$(B) = (4) = \int_0^{z_*} f(z) \int_0^{z} \lambda e^{-\lambda x} \int_0^{z_*-z+x} \lambda e^{-\lambda y} dydxdz$$

$$(C) = (2) + (5) = \int_0^{+\infty} f(z) \int_z^{+\infty} \lambda e^{-\lambda x} \int_{x-z}^{x-z+z^*} \lambda e^{-\lambda y} dydxdz$$

$$(D) = (3) + (6) = \int_0^{+\infty} f(z) \int_0^{+\infty} \lambda e^{-\lambda y} \int_{y+z}^{y+z+z^*} \lambda e^{-\lambda x} dxdydz$$

Sviluppo di (A):

$\int_{z_*}^{+\infty} f(z) \int_{z-z_*}^{z} \lambda e^{-\lambda x} \int_0^{z_*-z+x} \lambda e^{-\lambda y} dydxdz =$

$= \int_{z_*}^{+\infty} f(z) \int_{z-z_*}^{z} \lambda e^{-\lambda x} (-e^{-\lambda(z_*-z)} e^{-\lambda x} + 1) dxdz =$

$= \int_{z_*}^{+\infty} f(z) \left[- \int_{z-z_*}^{z} \lambda e^{-\lambda(z_*-z)} e^{-2\lambda x} dx + \int_{z-z_*}^{z} \lambda e^{-\lambda x} dx \right] dz =$

$= \int_{z_*}^{+\infty} f(z) \left[e^{-\lambda z_*} e^{\lambda z} \left(\frac{e^{-2\lambda z}}{2} - \frac{e^{-2\lambda(z-z_*)}}{2} \right) - e^{-\lambda z} + e^{-\lambda(z-z_*)} \right] dz =$

$= \int_{z_*}^{+\infty} f(z) \left[\frac{e^{-\lambda z_*} e^{-\lambda z}}{2} - \frac{e^{\lambda z_*} e^{-\lambda z}}{2} - e^{-\lambda z} + e^{\lambda z_*} e^{-\lambda z} \right] dz =$

$= \int_{z_*}^{+\infty} f(z) \left[\frac{e^{-\lambda z_*} e^{-\lambda z}}{2} + \frac{e^{\lambda z_*} e^{-\lambda z}}{2} - e^{-\lambda z} \right] dz$

Sviluppo di (B):

$\int_0^{z_*} f(z) \int_0^{z} \lambda e^{-\lambda x} \int_0^{z_*-z+x} \lambda e^{-\lambda y} dydxdz =$

$= \int_0^{z_*} f(z) \int_0^{z} \lambda e^{-\lambda x} \left(-e^{-\lambda(z_*-z)} e^{-\lambda x} + 1 \right) dxdz =$

$= \int_0^{z_*} f(z) \left[- \int_0^{z} \lambda e^{-\lambda(z_*-z)} e^{-2\lambda x} dx + \int_0^{z} \lambda e^{-\lambda x} dx \right] dz =$

$= \int_0^{z_*} f(z) \left[e^{-\lambda z_*} e^{\lambda z} \left(\frac{e^{-2\lambda z}}{2} - \frac{1}{2} \right) - e^{-\lambda z} + 1 \right] dz =$

$= \int_0^{z_*} f(z) \left[\frac{e^{-\lambda z_*} e^{-\lambda z}}{2} - \frac{e^{-\lambda z_*} e^{\lambda z}}{2} - e^{-\lambda z} + 1 \right] dz$

Sviluppo di (C):

$\int_0^{+\infty} f(z) \int_z^{+\infty} \lambda e^{-\lambda x} \int_{x-z}^{x-z+z_*} \lambda e^{-\lambda y} dydxdz =$

$= \int_0^{+\infty} f(z) \int_z^{+\infty} \lambda e^{-\lambda x} \left[-e^{-\lambda(z_*-z)} e^{-\lambda x} + e^{\lambda z} e^{-\lambda x} \right] dxdz =$

$= \int_0^{+\infty} f(z) \left[- \int_z^{+\infty} \lambda e^{-\lambda(z_*-z)} e^{-2\lambda x} dx + \int_z^{+\infty} \lambda e^{\lambda z} e^{-2\lambda x} dx \right] dz =$

$= \int_0^{+\infty} f(z) \left[e^{-\lambda z_*} e^{\lambda z} \left(0 - \frac{e^{-2\lambda z}}{2} \right) + \frac{e^{\lambda z} e^{-2\lambda z}}{2} \right] dz =$

$= \int_0^{+\infty} f(z) \left[- \frac{e^{-\lambda z_*} e^{-\lambda z}}{2} + \frac{e^{-\lambda z}}{2} \right] dz$

Sviluppo di (D):

$\int_0^{+\infty} f(z) \int_0^{+\infty} \lambda e^{-\lambda y} \int_{y+z}^{y+z+z_*} \lambda e^{-\lambda x} dxdydz =$

$= \int_0^{+\infty} f(z) \int_0^{+\infty} \lambda e^{-\lambda y} \left(-e^{-\lambda(z+z_*)} e^{-\lambda y} + e^{-\lambda z} e^{-\lambda y}\right) dy dz =$

$= \int_0^{+\infty} f(z) \left[- \int_0^{+\infty} \lambda e^{-\lambda(z+z_*)} e^{-2\lambda y} dy + \int_0^{+\infty} \lambda e^{-\lambda z} e^{-2\lambda y} dy \right] dz =$

$= \int_0^{+\infty} f(z) \left[-\frac{e^{-\lambda z_*} e^{-\lambda z}}{2} + \frac{e^{-\lambda z}}{2} \right] dz$

Essendo (C) = (D), si può definire:

$(C^*) = (C) + (D) =$

$= 2 \int_0^{+\infty} f(z) \left[-\frac{e^{-\lambda z_*} e^{-\lambda z}}{2} + \frac{e^{-\lambda z}}{2} \right] dz = \int_0^{+\infty} f(z) \left[-e^{-\lambda z_*} e^{-\lambda z} + e^{-\lambda z} \right] dz$

Quindi:

$\mathbb{P}[Z \leq z^*] = (A) + (B) + (C^*) =$

$= \int_{z_*}^{+\infty} f(z) \left[\frac{e^{-\lambda z_*} e^{-\lambda z}}{2} + \frac{e^{\lambda z_*} e^{-\lambda z}}{2} - e^{-\lambda z} \right] dz + \int_0^{z_*} f(z) \left[\frac{e^{-\lambda z_*} e^{-\lambda z}}{2} - \frac{e^{-\lambda z_*} e^{\lambda z}}{2} - e^{-\lambda z} + 1 \right] dz + \int_0^{+\infty} f(z) \left[-e^{-\lambda z_*} e^{-\lambda z} \right.$

$= - \int_0^{+\infty} f(z) \frac{e^{-\lambda z_*} e^{-\lambda z}}{2} dz + \int_{z_*}^{+\infty} f(z) \frac{e^{\lambda z_*} e^{-\lambda z}}{2} dz - \int_0^{z^*} f(z) \frac{e^{-\lambda z_*} e^{\lambda z}}{2} dz + \int_0^{z_*} f(z) dz$

Si sceglie di riformulare nel seguente modo:

$P[Z_i \leq z_*] = - \int_0^{+\infty} f_{i-1}(z) \frac{e^{-\lambda z_*} e^{-\lambda z}}{2} dz + \int_{z_*}^{+\infty} f_{i-1}(z) \frac{e^{\lambda z_*} e^{-\lambda z}}{2} dz - \int_0^{z_*} f_{i-1}(z) \frac{e^{-\lambda z_*} e^{\lambda z}}{2} dz + \int_0^{z_*} f_{i-1}(z) dz$

dove:

- Z_i rappresenta il tempo di attesa del pezzo i-esimo;
- $f_i(z)$ è la distribuzione di probabilità dell'i-esimo tempo di attesa per $2 \leq i \leq N$.

Tornerà utile in seguito la seguente definizione di funzioni:

- $h_1(f(z)) = \int_0^{z_*} f(z) dz$;
- $h_2(f(z)) = - \int_0^{z_*} f(z) \frac{e^{-\lambda z_*} e^{\lambda z}}{2} dz$;
- $h_3(f(z)) = \int_{z_*}^{+\infty} f(z) \frac{e^{\lambda z_*} e^{-\lambda z}}{2} dz$;
- $h_4(f(z)) = - \int_0^{+\infty} f(z) \frac{e^{-\lambda z_*} e^{-\lambda z}}{2} dz$.

Si ricordi che il tempo di attesa alla stazione di matching del FTA della prima coppia (coppia 0) è caratterizzato dalla funzione:

$$f_1(z) = \lambda e^{-\lambda z}$$

Sostituendo e risolvendo all'interno dell'integrale relativo alla distribuzione di probabilità cumulata dei tempi di attesa della seconda coppia (coppia 1) si otterrà:

$$P[Z_2 \leq z_*] = 1 - e^{-\lambda z_*} - \frac{z_* \lambda e^{-\lambda z_*}}{2}$$

Derivando si avrà:

$$f_2(z) = \frac{\lambda e^{-\lambda z}}{2} + \frac{z \lambda^2 e^{-\lambda z}}{2}$$

Reiterando il processo per ogni i che sia maggiore o uguale a 2, sarà possibile notare la possibilità di riformulare nel seguente modo:

$$f_i(z) = \sum_{m=1}^{i} \alpha_{m,i} z^{m-1} \lambda^m e^{-\lambda z}$$

Si definisca per comodità:

$$g_m(z) = z^{m-1} \lambda^m$$

Sostituendo, si otterrà:

$$f_i(z) = \sum_{m=1}^{i} \alpha_{m,i} z^{m-1} \lambda^m e^{-\lambda z} = \sum_{m=1}^{i} \alpha_{m,i} g_m(z) e^{-\lambda z}$$

Sarà possibile riorganizzare la formula $P\left[Z_i \leq z_*\right]$ nella maniera seguente:

$P\left[Z_{i+1} \leq z_*\right] = h_1(f_i(z)) + h_2(f_i(z)) + h_3(f_i(z)) + h_4(f_i(z)) =$

$= h_1(\sum_{m=1}^{i} \alpha_{m,i} g_m(z) e^{-\lambda z}) + h_2(\sum_{m=1}^{i} \alpha_{m,i} g_m(z) e^{-\lambda z}) + h_3(\sum_{m=1}^{i} \alpha_{m,i} g_m(z) e^{-\lambda z}) + h_4(\sum_{m=1}^{i} \alpha_{m,i} g_m(z) e^{-\lambda z}) =$

$= \sum_{w=1}^{4} h_w(\sum_{m=1}^{i} \alpha_{m,i} g_m(z) e^{-\lambda z})$

dove:

$$h_1(f_i(z)) = \int_0^{z_*} (\sum_{m=1}^{i} \alpha_{m,i} g_m(z) e^{-\lambda z}) dz$$

$$h_2(f_i(z)) = -\int_0^{z_*} \frac{(\sum_{m=1}^{i} \alpha_{m,i} g_m(z) e^{-\lambda z_*})}{2} dz$$

$$h_3(f_i(z)) = \int_{z_*}^{+\infty} \frac{(\sum_{m=1}^{i} \alpha_{m,i} g_m(z) e^{\lambda z_*} e^{-2\lambda z})}{2} dz$$

$$h_4(f_i(z)) = -\int_0^{+\infty} \frac{(\sum_{m=1}^{i} \alpha_{m,i} g_m(z) e^{-\lambda z_*} e^{-2\lambda z})}{2} dz$$

Per le proprietà degli integrali e delle sommatorie si avrà:

$$\mathbb{P}\left[Z_{i+1} \leq z_*\right] = \sum_{w=1}^{4} h_w(\sum_{m=1}^{i} \alpha_{m,i} g_m(z) e^{-\lambda z}) =$$

$$= \sum_{m=1}^{i} [\alpha_{m,i} (\sum_{w=1}^{4} h_w(g_m(z) e^{-\lambda z})]$$

dove:

$$h_1(g_m(z)) = \int_0^{z_*} g_m(z)e^{-\lambda z}dz$$

$$h_2(g_m(z)) = -\int_0^{z_*} \frac{g(z)e^{-\lambda z_*}}{2}dz$$

$$h_3(g_m(z)) = \int_{z_*}^{+\infty} \frac{g(z)e^{\lambda z_*}e^{-2\lambda z}}{2}dz$$

$$h_4(g_m(z)) = -\int_0^{+\infty} \frac{g(z)e^{-\lambda z_*}e^{-2\lambda z}}{2}dz$$

Sviluppo di $h_1(g_m(z))$:

$h_1(g_m(z)) = \int_0^{z_*} g_m(z)e^{-\lambda z}dz = \int_0^{z_*} z^{m-1}\lambda^m e^{-\lambda z}dz =$

$= -z_*^{m-1}\lambda^{m-1}e^{-\lambda z_*} - (m-1)z_*^{m-2}\lambda^{m-2}e^{-\lambda z_*} - (m-1)(m-2)z_*^{m-3}\lambda^{m-3}e^{-\lambda z_*} - ..... - \prod_{j=1}^{m-1}(je^{-\lambda z_*}) + \prod_{j=1}^{m-1} j =$

$= -z_*^{m-1}\lambda^{m-1}e^{-\lambda z_*} - e^{-\lambda z_*}\sum_{k=0}^{m-2}\left[z_*^k\lambda^k \prod_{j=k+1}^{m-1} j\right]$

Sviluppo di $h_2(g_m(z))$:

$h_2(g_m(z)) = -\int_0^{z_*} \frac{g(z)e^{-\lambda z_*}}{2}dz = -\frac{1}{2m}z_*^m\lambda^m e^{-\lambda z_*}$

Sviluppo di $h_3(g_m(z))$:

$h_3(g_m(z)) = \int_{z_*}^{+\infty} \frac{g(z)e^{\lambda z_*}e^{-2\lambda z}}{2}dz = \int_{z_*}^{+\infty} \frac{z_*^{m-1}\lambda^m e^{\lambda z_*}e^{-2\lambda z}}{2}dz =$

$= \frac{1}{2}[\frac{1}{2}z_*^{m-1}\lambda^{m-1}e^{-\lambda z_*} + \frac{1}{4}(m-1)z_*^{m-2}\lambda^{m-2}e^{-\lambda z_*} + \frac{1}{8}(m-1)(m-2)z_*^{m-3}\lambda^{m-3}e^{-\lambda z_*} + ... + (\frac{1}{2})^m(\prod_{j=1}^{m-1} j)e^{-\lambda z_*}] =$

$= \frac{1}{4}z_*^{m-1}\lambda^{m-1}e^{-\lambda z_*} + \frac{e^{-\lambda z_*}}{2}\sum_{k=0}^{m-2}\left[z_*^k\lambda^k(\frac{1}{2})^{m-k}\prod_{j=k+1}^{m-1} j\right] =$

$= \frac{1}{4}z_*^{m-1}\lambda^{m-1}e^{-\lambda z_*} + e^{-\lambda z_*}\sum_{k=0}^{m-2}\left[z_*^k\lambda^k(\frac{1}{2})^{m+1-k}\prod_{j=k+1}^{m-1} j\right]$

Sviluppo di $h_4(g_m(z))$:

$h_4(g_m(z)) = -\int_0^{+\infty} \frac{g(z)e^{-\lambda z_*}e^{-2\lambda z}}{2}dz = -\left(\frac{1}{2}\right)^{m+1}(\prod_{j=1}^{m-1} j)e^{-\lambda z_*}$

Si definisca:

$\Delta_m(z) = \sum_{w=1}^{4} h_w(g_m(z)) =$

$= -\frac{1}{2m}z_*^m\lambda^m e^{-\lambda z_*} - \frac{3}{4}(z_*\lambda)^{m-1}e^{-\lambda z_*} - e^{-\lambda z_*}\sum_{k=1}^{m-2}\left[(z_*\lambda)^k\left(1-\left(\frac{1}{2}\right)^{m+1-k}\right)\prod_{j=k+1}^{m-1} j\right] - e^{-\lambda z_*}\prod_{j=1}^{m-1} j + \prod_{j=1}^{m-1} j =$

$= a_m + b_m e^{-\lambda z_*} + \sum_{k=1}^{m-2}[c_{k,m}(z\lambda)^k e^{-\lambda z_*}] + d_m(z\lambda)^{m-1}e^{-\lambda z_*} + l_m(z_*\lambda)^m e^{-\lambda z_*}$

dove:

$a_m = \prod_{j=1}^{m-1} j, \forall m$

$b_m = -\prod_{j=1}^{m-j} j, \forall m, k = 0$

$c_{k,m} = -\left(1 - \left(\frac{1}{2}\right)^{m+1-k}\right) \prod_{j=k+1}^{m-1} j, \forall m, 1 \leq k \leq m-2$

$d_m = -\frac{3}{4}, \forall m$

$l_m = -\frac{1}{2m}, \forall m$

La tabella dei coeficienti dei $\Delta_m(z)$ potrà essere formulata nel modo seguente:

	Δ_1	Δ_2	Δ_3	Δ_4	...	Δ_m
numero	1	1	a_3	a_4		a_m
$e^{-\lambda z}$	-1	-1	b_3	b_4		b_m
(k=1) $(z\lambda)^k e^{-\lambda z}$	$-\frac{1}{2}$	$-\frac{3}{4}$	$c_{1,3}$	$c_{1,4}$		$c_{1,m}$
(k=2) $(z\lambda)^k e^{-\lambda z}$	0	$-\frac{1}{4}$	$-\frac{3}{4}$	$c_{2,4}$		$c_{2,m}$
(k=3) $(z\lambda)^k e^{-\lambda z}$	0		l_3	$-\frac{3}{4}$		$c_{3,m}$
(k=4) $(z\lambda)^k e^{-\lambda z}$	0			l_4		$c_{4,m}$
...						
(k=m-2) $(z\lambda)^k e^{-\lambda z}$	0					$c_{k,m}$
(k=m-1) $(z\lambda)^k e^{-\lambda z}$	0					d_m
(k=m) $(z\lambda)^k e^{-\lambda z}$	0					l_m

Ogni colonna conterrà il vettore di coeficienti proprio della funzione $\Delta_m(z)$. di cui $\Delta_{x,m}$ ne sarà l'elemento x-esimo.

Quindi:

$$\Delta_m(z) = a_m + b_m e^{-\lambda z_*} + \sum_{k=1}^{m-2} \left[c_{k,m} (z\lambda)^k e^{-\lambda z_*}\right] + d_m (z\lambda)^{m-1} e^{-\lambda z_*} + l_m (z_*\lambda)^m e^{-\lambda z_*} =$$

$$= \Delta_{1,m} + \sum_{x=2}^{m+2} (\Delta_{x,m} (z_*\lambda)^{x-2} e^{-\lambda z_*})$$

Si ricordi che:

- $\mathbb{P}\left[Z_{i+1} \leq z_*\right] = \sum_{m=1}^{i} \left[\alpha_{m,i} \sum_{w=1}^{4} h_w(g_m(z_*))\right]$
- $\sum_{w=1}^{4} h_w(g_m(z_*)) = \Delta_m(z_*)$
- $\Delta_m(z_*) = \Delta_{1,m} + \sum_{x=2}^{m+2} (\Delta_{x,m} (z_*\lambda)^{x-2} e^{-\lambda z_*})$

Sostituendo si ottiene:

$$\mathbb{P}\left[Z_{i+1} \leq z_*\right] = \sum_{m=1}^{i} \left[\alpha_{m,i} \Delta_m(z_*)\right] =$$

$$= \sum_{m=1}^{i} \left[\alpha_{m,i} (\Delta_{1,m} + \sum_{x=2}^{m+2} (\Delta_{x,m} (z_*\lambda)^{x-2} e^{-\lambda z_*}))\right] =$$

$= I_{1,i} + \sum_{x=2}^{m+2} (I_{x,i}(z_*\lambda)^{x-2}e^{-\lambda z_*});$

dove:

$I_{x,i} = \sum_{m=1}^{i} (\alpha_{m,i}\Delta_{x,m})$

Sarà possibile creare la tabella dei coeficienti delle funzioni di probabilità cumulata dei tempi di attesa Z della coppia i-esima (P_i) nel modo seguente:

	P_1	P_2	P_3	P_4	P_5	...
(x=1) numero	1	1	$I_{1,3}$	$I_{1,4}$	$I_{1,5}$	
(x=2) $(z\lambda)^{x-2}e^{-\lambda z}$	-1	-1	$I_{2,3}$	$I_{2,4}$	$I_{2,5}$	
(x=3) $(z\lambda)^{x-2}e^{-\lambda z}$	$-\frac{1}{2}$	$-\frac{5}{8}$	$I_{3,3}$	$I_{3,4}$	$I_{3,5}$	
(x=4) $(z\lambda)^{x-2}e^{-\lambda z}$	0	$-\frac{1}{8}$	$I_{4,3}$	$I_{4,4}$	$I_{4,5}$	
(x=5) $(z\lambda)^{x-2}e^{-\lambda z}$	0	0	$I_{5,3}$	$I_{5,4}$	$I_{5,5}$	
(x=6) $(z\lambda)^{x-2}e^{-\lambda z}$	0	0	0	$I_{6,4}$	$I_{6,5}$	
(x=7) $(z\lambda)^{x-2}e^{-\lambda z}$	0	0	0	0	$I_{7,5}$	

Come per la tabella dei Δ, anche in questo caso ogni colonna rappresenta il vettore dei coeficienti della corrispondente funzione P_i.

Derivando la funzione di probabilità cumulata appena ottenuta, si potrà calcolare la corrispondente funzione di densità della probabilità.

$f_{i+1}(z_*) = \frac{dP_i}{dz_*} =$

$= \sum_{m=1}^{i} \left[-(I_{m+1,i} - m * I_{m+2,i})z_*^{m-1}\lambda^m e^{-\lambda z_*}\right] - I_{i+1,i+1}z_*^i\lambda^{i+1}e^{-\lambda z_*} =$

$= \sum_{m=1}^{i+1} \left[\alpha_{m,i+1}z_*^{m-1}\lambda^m\right] e^{-\lambda z_*}$

dove:

- per $1 \leq m \leq i$ si avrà $\alpha_{m,i+1} = -I_{m+1,i} + (m * I_{m+2,i})$
- per $m = i + 1$ si avrà $\alpha_{i+1,i+1} = -I_{i+2,i}$

	f_1	f_2	f_3	f_4	f_5	...
(m=1) $z^{m-1}\lambda^m e^{-\lambda z}$	1	$\frac{1}{2}$	$\frac{3}{8}$	$\alpha_{1,4}$	$\alpha_{1,5}$	
(m=2) $z^{m-1}\lambda^m e^{-\lambda z}$		$\frac{1}{2}$	$\frac{3}{8}$	$\alpha_{2,4}$	$\alpha_{2,5}$	
(m=3) $z^{m-1}\lambda^m e^{-\lambda z}$			$\frac{1}{8}$	$\alpha_{3,4}$	$\alpha_{3,5}$	
(m=4) $z^{m-1}\lambda^m e^{-\lambda z}$				$\alpha_{4,4}$	$\alpha_{4,5}$	
(m=5) $z^{m-1}\lambda^m e^{-\lambda z}$					$\alpha_{5,5}$	
(m=6) $z^{m-1}\lambda^m e^{-\lambda z}$						

Si potrà infine calcolare la media e la varianza di «z» al turno i-esimo.

Calcolo della media:

$\overline{z_i} = \int_0^{+\infty} z f_i(z)dz = \int_0^{\infty} z \sum_{m=1}^{i} \left[\alpha_{m,i}g_m(z)e^{-\lambda z}\right] dz =$

$= \sum_{m=1}^{i} \left[\alpha_{m,i} \int_0^{\infty} z g_m(z)e^{-\lambda z}dz\right] =$

$= \sum_{m=1}^{i} \left[\alpha_{m,i} \int_0^{\infty} z^m\lambda^m e^{-\lambda z}dz\right] =$

$$= \sum_{m=1}^{i} \left[\alpha_{m,i} \tfrac{1}{\lambda} \prod_{j=1}^{m} j \right] =$$

$$= \tfrac{1}{\lambda} \sum_{m=1}^{i} \left[\alpha_{m,i} \prod_{j=1}^{m} j \right]$$

Calcolo della varianza:

$\delta_i^2 = \int_0^{\infty} z^2 f_i(z) dz - \overline{z_i}^2$

Sviluppo del primo addendo:

$$\int_0^{+\infty} z^2 f_i(z) dz = \int_0^{+\infty} z^2 \sum_{m=1}^{i} \left[\alpha_{m,i} g_m(z) e^{-\lambda z} \right] dz =$$

$$= \sum_{m=1}^{i} \left[\alpha_{m,i} \int_0^{\infty} z^{m+1} \lambda^m e^{-\lambda z} dz \right] = \tfrac{1}{\lambda^2} \sum_{m=1}^{i} \left[\alpha_{m,i} \prod_{j=1}^{m+1} j \right]$$

Quindi:

$$\delta_i^2 = \tfrac{1}{\lambda^2} \sum_{m=1}^{i} \left[\alpha_{m,i} \prod_{j=1}^{m+1} j \right] - \overline{z_i}^2$$

Su un intero lotto di N pezzi, facendo la media dei valori medi e delle varianze specifici delle singole coppie, si avrà:

- $\overline{z_N} = \tfrac{1}{N} \sum_{i=1}^{N} \overline{z_i}$
- $\delta_N^2 = \tfrac{1}{N} \sum_{i=1}^{N} \delta_i^2$

Ottenute le funzioni di densita e cumulate della probabilità dei tempi di attesa di ogni singola coppia facente parte dell'ipotetico lotto, è possibile elaborare una funzione di densità della probabilità dei tempi di attesa generica per una qualsiasi coppia all'interno del suddetto lotto, che per comodità verrà definita «globale».

Si procederà calcolando una media delle distribuzioni di probabilità cumulata specifiche delle singole coppie :

$$\mathbb{P}_N \left[Z_{g,N} \leq z_* \right] = \tfrac{1}{N} \sum_{i=1}^{N} \int_0^{z_*} f_i(z) \, dz = \tfrac{1}{N} \sum_{i=1}^{N} \left[\sum_{m=1}^{i} \alpha_{m,i} \int_0^{z_*} z^{m-1} \lambda^m e^{-\lambda z} dz \right]$$

Si noti che:

$$\int_0^{z_*} z^{m-1} \lambda^m e^{-\lambda z} dz = - \sum_{k=1}^{m-1} \left[\tfrac{(m-1!)}{k!} z_*^k \lambda^k e^{-\lambda z_*} \right] - (m-1)! e^{-\lambda z_*} + (m-1)!$$

Quindi:

$$\mathbb{P}_N \left[Z_{g,N} \leq z_* \right] = \tfrac{1}{N} \alpha_{1,1} \left(1 - e^{-\lambda z_*} \right) - \tfrac{1}{N} \sum_{i=2}^{N} \left\{ \sum_{m=2}^{i} \left[-\alpha_{1,i} + \alpha_{1,i} e^{-\lambda z_*} + \alpha_{m,i} \sum_{k=0}^{m-1} \left(\tfrac{(m-1!)}{k!} z_*^k \lambda^k e^{-\lambda z_*} \right) \right] \right\}$$

Si definisca:

$\Omega_{0,N} = \tfrac{1}{N} \alpha_{1,1} \left(1 - e^{-\lambda z_*} \right) - \tfrac{1}{N} \sum_{i=2}^{N} \sum_{m=2}^{i} \left[\alpha_{m,i} \left(m-1 \right)! \right]$, coeficiente di $e^{-\lambda z}$;

$\Omega_{k,N} = -\tfrac{1}{N} \sum_{i=k+1}^{N} \sum_{m=k+1}^{i} \left[\alpha_{m,i} \tfrac{(m-1)!}{k!} \right]$, coeficiente di $z_*^k \lambda^k e^{-\lambda z}$per $1 \leq k \leq N-1$.

Si otterrà:

$\mathbb{P}_N\left[Z_{g,N} \leq z_*\right] = \sum_{k=0}^{N-1} \left(\Omega_{k,N} z_*^k \lambda^k e^{-\lambda z_*}\right) + Costante$

Derivando, si avrà il risultato desiderato:

$f_{g,N}(z) = \frac{d}{dz}\mathbb{P}_N\left[Z_{g,N} \leq z_*\right] = \sum_{k=0}^{N-1} \omega_{k,N} g_{k+1}(z)\, e^{-\lambda z}$

Dove:

$\omega_{k,N} = -\Omega_k + (k+1)\,\Omega_{k+1}$;

$\omega_{N-1,N} = -\Omega_{N-1}$.

Calcolo della media dei tempi di attesa su un lotto di N coppie utilizzando la funzione di densità di probabilità globale.

$\overline{z}_{g,N} = \int_0^\infty z f_{g,N}(z)\, dz = \sum_{k=0}^{N-1} \left[\omega_{k,n} \int_0^\infty z^{k+1}\lambda^{k+1} e^{-\lambda z} dz\right] = \sum_{k=0}^{N-1} \left[\omega_{k,N} \frac{(k+1)!}{\lambda}\right]$

Calcolo della varianza dei tempi di attesa su un lotto di N coppie utilizzando la funzione di densità di probabilità globale.

$Var_{g,N}[z] = \int_0^\infty (z - \overline{z}_{g,N})^2 f_{g,N}(z)\, dz = \sum_{k=0}^{N-1} \left[\omega_{k,N} \left(\int_0^\infty z^{k+2}\lambda^{k+1} e^{-\lambda z} dz + \overline{z}_{g,N}^2 \int_0^\infty z^k \lambda^{k+1} e^{-\lambda z} dz - 2\overline{z}_{g,N} \int_0^\infty z^{k+1}\lambda^{k+1}\right.\right.$

$= \sum_{k=0}^{N-1} \left[\omega_{k,N} \frac{(k+2)!}{\lambda^2}\right] + \overline{z}_{g,N}^2 \int_0^\infty \sum_{k=0}^{N-1} \left[\omega_{k,N} g_{k-1}(z)\, e^{-\lambda z}\right] dz - 2\overline{z}_{g,N} \int_0^\infty z \sum_{k=0}^{N-1} \left[\omega_{k,N} g_{k-1}(z)\, e^{-\lambda z}\right] dz =$

$= \sum_{k=0}^{N-1} \left[\omega_{k,N} \frac{(k+2)!}{\lambda^2}\right] + \overline{z}_{g,N}^2 \int_0^\infty f_{g,N}(z)\, dz - 2\overline{z}_{g,N} \int_0^\infty z f_{g,N}(z)\, dz = \sum_{k=0}^{N-1} \left[\omega_{k,N} \frac{(k+2)!}{\lambda^2}\right] - \overline{z}_{g,N}^2$

Intertempi di arrivo

Calcolate le funzioni di probabilità dei tempi di attesa, si potranno calcolare le funzioni dei tempi di interarrivo alla stazione di accoppiamento.

Come in precedenza, si separi l'analisi della prima coppia da quella delle successive.

Il primo FTA giungerà alla stazione di matching in un momento in cui questa sarà vuota ed in attesa. La funzione di probabilità cumulata del suo tempo di arrivo sarà:

$\mathbb{P}[T \leq t] = \int_0^t \lambda e^{-\lambda x} \int_x^{+\infty} \lambda e^{-\lambda y} dy dx + \int_0^t \lambda e^{-\lambda y} \int_x^{+\infty} \lambda e^{-\lambda x} dx dy =$

$= \int_0^t \lambda e^{-\lambda x} e^{-\lambda x} dx + \int_0^t \lambda e^{-\lambda y} e^{-\lambda y} dy = -\frac{e^{-2\lambda t}}{2} + \frac{1}{2} - \frac{e^{-2\lambda t}}{2} + \frac{1}{2} =$

$= 1 - e^{-2\lambda t}$

Derivando si ottiene:

$\frac{d\mathbb{P}[T \leq t]}{dt} = 2\lambda e^{-2\lambda t}$

Si definisca

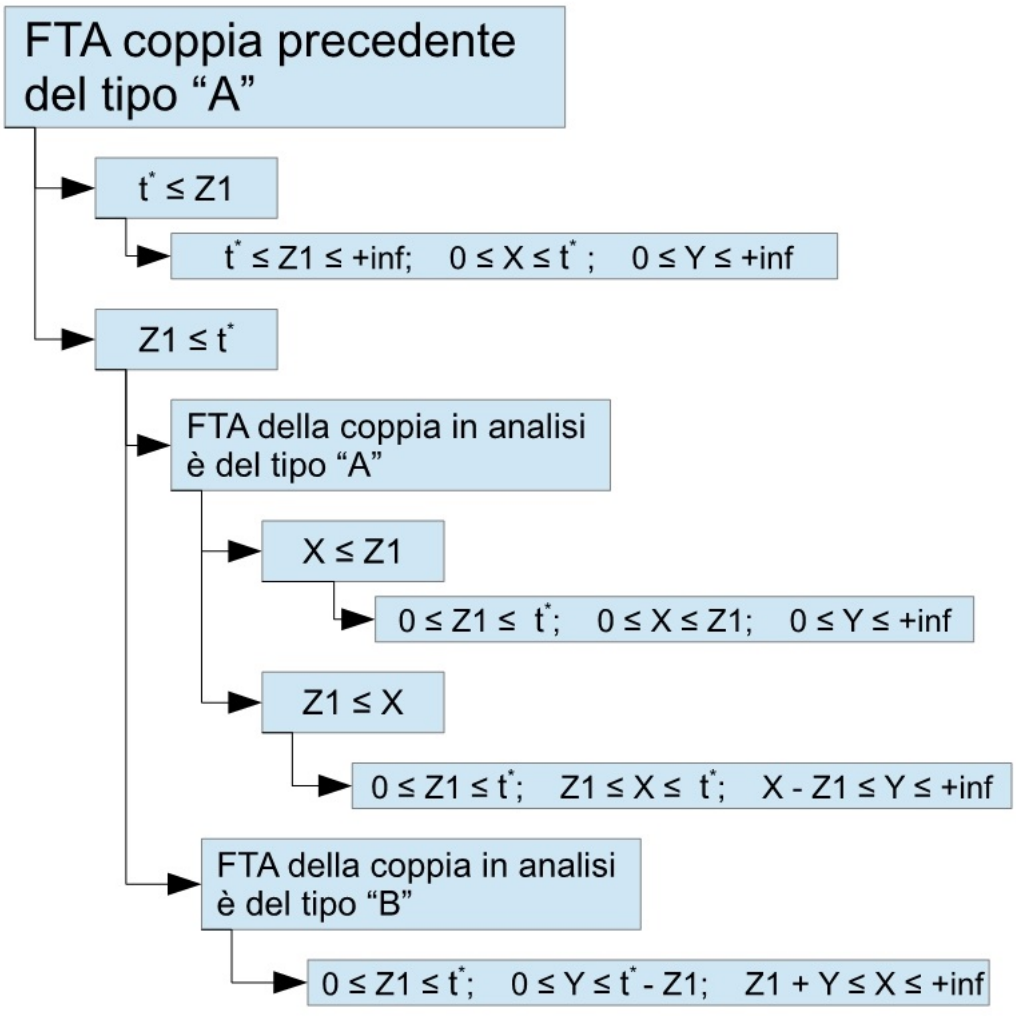
- $h_0(t) = 2\lambda e^{-2\lambda t}$;
- $h_i(t)$ come la funzione di densità della probabilità del tempo di interarrivo alla stazione di matching del (i+1)esimo FTA.

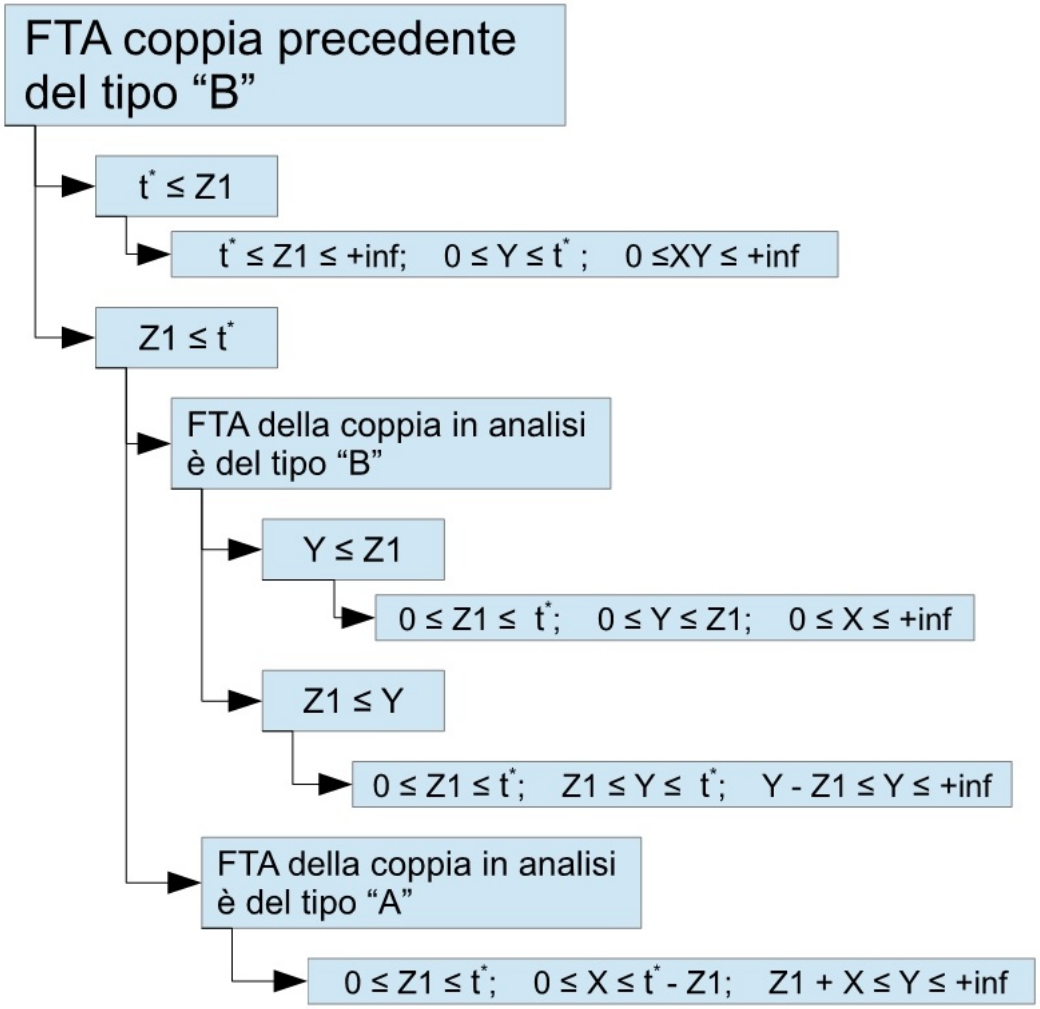
Il primo FTA in ingresso avrà:

- $E[t] = \bar{t} = \int_0^{+\infty} 2t\lambda e^{-2\lambda t} dt = \int_0^{+\infty} e^{-2\lambda t} dt = \frac{1}{2\lambda}$

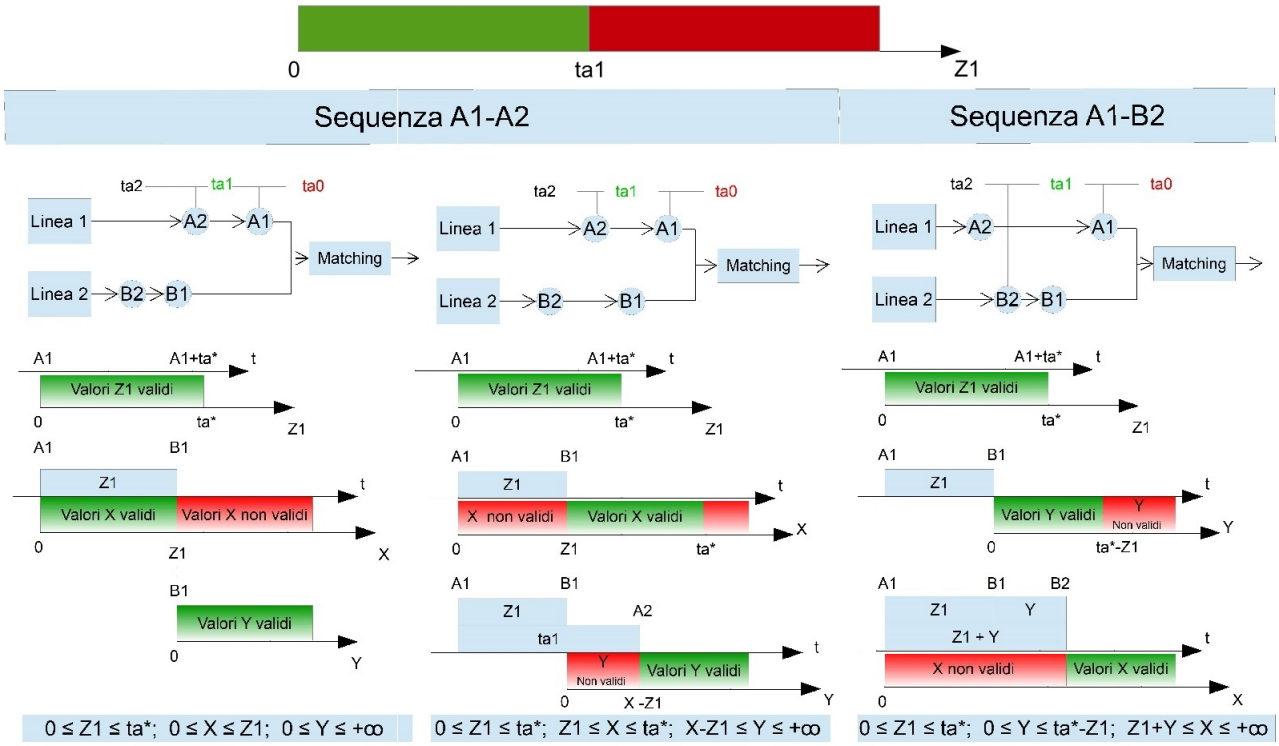
- $\delta_{t0}^2 = \int_0^{+\infty} 2t^2\lambda e^{-2\lambda t} dt - \bar{t}^2 = \int_0^{+\infty} \frac{e^{-2\lambda t}}{\lambda} dt - \bar{t}^2 = \frac{1}{2\lambda^2} - \frac{1}{4\lambda^2} = \frac{1}{4\lambda^2}$

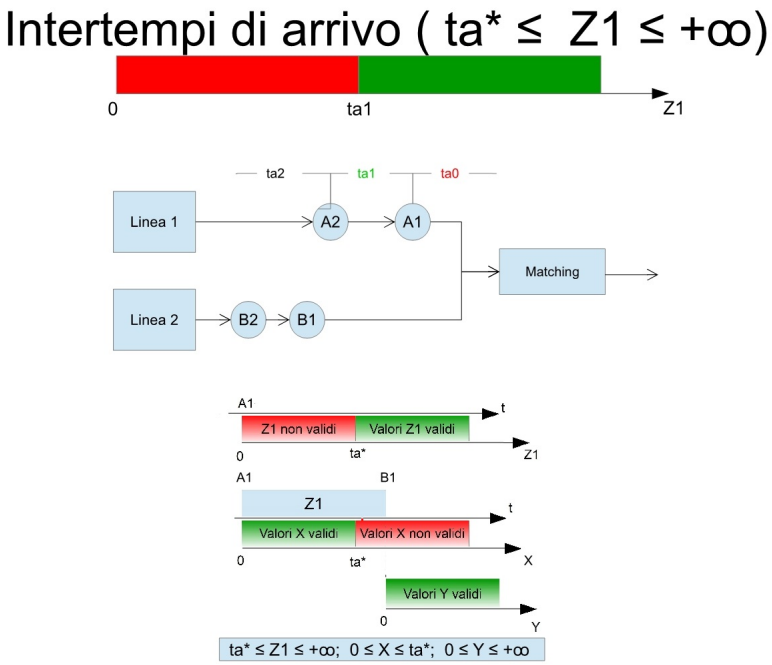
Come in precedenza, dal secondo FTA in poi, nel calcolo dei tempi di interarrivo entrano in gioco anche i tempi di attesa.





Intertempi di arrivo ($0 \le Z1 \le ta^*$)





Si eseguano i calcoli sul primo ramo dello schema, avente come presupposto un precedente «X primo in ingresso».

Sviluppo del primo campo (A):

$\frac{1}{2}\int_t^{+\infty} f_i(z)\int_0^t \lambda e^{-\lambda x}\int_0^{+\infty}\lambda e^{-\lambda y}dydxdz =$

$\frac{1}{2}\int_t^{+\infty} f_i(z)\int_0^t \lambda e^{-\lambda x}dxdz = \frac{1}{2}\int_t^{+\infty} f_i(z)(1-e^{-\lambda z})dz$

Sviluppo del secondo campo (B):

$\frac{1}{2}\int_0^t f_i(z)\int_0^z \lambda e^{-\lambda x}\int_0^{+\infty}\lambda e^{-\lambda y}dydxdz =$

$= \frac{1}{2}\int_0^t f_i(z)\int_0^z \lambda e^{-\lambda x}dxdz = \frac{1}{2}\int_0^t f_i(z)(1-e^{-\lambda z})dz$

Sviluppo del terzo campo (C):

$\frac{1}{2}\int_0^t f_i(z)\int_z^t \lambda e^{-\lambda x}\int_{x-z}^{+\infty}\lambda e^{-\lambda y}dydxdz =$

$= \frac{1}{2}\int_0^t f_i(z)\int_z^t \lambda e^{-\lambda x}e^{-\lambda(x-z)}dxdz = \frac{1}{2}\int_0^t \frac{1}{2}f_i(z)e^{\lambda z}(e^{-2\lambda z}-e^{-2\lambda t})dz$

Sviluppo del quarto campo (D):

$\frac{1}{2}\int_0^t f_i(z)\int_0^{t-z} \lambda e^{-\lambda y}\int_{z+y}^{+\infty}\lambda e^{-\lambda x}dxdydz =$

$= \frac{1}{2}\int_0^t f_i(z)\int_0^{t-z} \lambda e^{-\lambda y}e^{-\lambda(z+y)}dydz = \frac{1}{2}\int_0^t \frac{1}{2}f_i(z)e^{-\lambda z}(1-e^{-2\lambda(t-z)}dz$

I calcoli sull'altro ramo daranno risultati simmetrici:

- (A)=(E);
- (B)=(F);
- (C)=(G);
- (D)=(H);

Quindi:

$\mathbb{P}[T_i \leq t] = (A)+(B)+(C)+(D)+(E)+(F)+(G)+(H) =$

$= 2(A)+2(B)+2(C)+2(D) =$

$= \int_t^{+\infty} f_i(z)dz - e^{-\lambda t}\int_t^{+\infty} f_i(z)dz + \int_0^t f_i(z)(1 - e^{-\lambda z} + \frac{e^{-\lambda z}}{2} - \frac{e^{-2\lambda t}e^{\lambda z}}{2} + \frac{e^{-\lambda z}}{2} - \frac{e^{-2\lambda t}e^{\lambda z}}{2})dz =$

$= \int_t^{+\infty} f_i(z)dz - \int_t^{+\infty} e^{-\lambda t}f_i(z)dz + \int_0^t f_i(z)(1 - e^{-2\lambda t}e^{\lambda z})dz =$

$= \int_0^{+\infty} f_i(z)dz - \int_t^{+\infty} e^{-\lambda t}f_i(z)dz - \int_0^t f_i(z)e^{-2\lambda t}e^{\lambda z}dz =$

$= 1 - e^{-\lambda t}\int_t^{+\infty} f_i(z)dz - e^{-2\lambda t}\int_0^t f_i(z)e^{\lambda z}dz$

Si definiscano:

- $(1) = -e^{-\lambda t}\int_t^{+\infty} f_i(z)dz$
- $(2) = -e^{-2\lambda t}\int_0^t f_i(z)e^{\lambda z}dz$

Si ricordi che:

- $f_i(z) = \sum_{m=1}^{i} \alpha_{m,i} g_m(z)e^{-\lambda z}$
- $g_m(z) = z^{m-1}\lambda^m$

Sostituendo si ottiene:

- $(1) = -e^{-\lambda t}\int_t^{+\infty}(\sum_{m=1}^{i} \alpha_{m,i} g_m(z)e^{-\lambda z)}dz$
- $(2) = -e^{-2\lambda t}\int_0^t(\sum_{m=1}^{i} \alpha_{m,i} g_m(z)e^{-\lambda z})e^{\lambda z}dz$

Sviluppo di (1):

$(1) = -e^{-\lambda t}\int_t^{+\infty}(\sum_{m=1}^{i} \alpha_{m,i} g_m(z)e^{-\lambda z)}dz =$

$= -e^{-\lambda t}\sum_{m=1}^{i}(\alpha_{m,i}\int_t^{+\infty} g(z)e^{-\lambda z}dz)$

L'integrale contenuto all'interno della sommatoria, diventa:

$\int_t^{\infty} g(z)e^{-\lambda z}dz =$

$= t^{m-1}\lambda^{m-1}e^{-\lambda t} + (m-1)t^{m-2}\lambda^{m-2}e^{-\lambda t} + (m-1)(m-2)t^{m-3}\lambda^{m-3}e^{-\lambda t} + ...$

$... + (\prod_{j=2}^{m-1} j)t\lambda e^{-\lambda t} + (\prod_{j=2}^{m-1} j)e^{-\lambda t} =$

$= t^{m-1}\lambda^{m-1}e^{-\lambda t} + \sum_{k=0}^{m-2}\left[(\prod_{j=k+1}^{m-1} j)(t\lambda)^k\right]e^{-\lambda t}$

Quindi:

$(1) = -e^{-\lambda t}\sum_{m=1}^{i}(\alpha_{m,i}\int_t^{\infty} g(z)e^{-\lambda z}dz) =$

$= -e^{-\lambda t}\sum_{m=1}^{i}\left\{\alpha_{m,i}\left(t^{m-1}\lambda^{m-1}e^{-\lambda t} + \sum_{k=0}^{m-2}\left[(\prod_{j=k+1}^{m-1} j)(t\lambda)^k\right]e^{-\lambda t}\right)\right\}$

Si noti che il vincolo della prima sommatoria crea problemi alla seconda, generandole un limite superiore negativo.

Sviluppo di una tabella che permetta una miglior visualizzare dell'evoluzione di $\alpha_{m,i} \int_t^{+\infty} g_m(z)\, e^{-\lambda z} dz$:

m=1	$\alpha_{1,m}*$	$e^{-\lambda t}$
m=2	$\alpha_{2,m}*$	$e^{-\lambda t}(1+t\lambda)$
m=3	$\alpha_{3,m}*$	$e^{-\lambda t}(2+2t\lambda+t^2\lambda^2)$
m=4	$\alpha_{4,m}*$	$e^{-\lambda t}(6+6t\lambda+3t^2\lambda^2+t^3\lambda^3)$
m=5	$\alpha_{5,m}*$	$e^{-\lambda t}(24+24t\lambda+12t^2\lambda^2+4t^3\lambda^3+t^4\lambda^4)$
...	...	...

Quindi:

$$\sum_{m=1}^{i} \alpha_{m,i} \int_t^{+\infty} g_m(z)\, e^{-\lambda z} dz =$$

$$= \alpha_{i,i}(t\lambda)^{i-1}e^{-\lambda t} + \left[\alpha_{i-1,i} + \alpha_{i,i}(i-1)\right](t\lambda)^{i-2}e^{-\lambda t} + \left[\alpha_{i-2,i} + \alpha_{i-1,i}(i-2) + \alpha_{i,i}\prod_{j=i-2}^{i-1} j\right](t\lambda)^{i-3}e^{-\lambda t} +$$

$$\left[\alpha_{i-2,i} + \alpha_{i-1,i}(i-2) + \alpha_{i,i}\prod_{j=i-2}^{i-1} j\right](t\lambda)^{i-3}e^{-\lambda t} + \left[\alpha_{i-3,i} + \alpha_{i-2,i}(i-3) + \alpha_{i-1,i}\prod_{j=i-3}^{i-2} j + \alpha_{i,i}\prod_{j=i-3}^{i-1} j\right](t\lambda)^{i-4}e^{-}$$

$$\ldots =$$

$$= \alpha_{i,i}(t\lambda)^{i-1}e^{-\lambda t} + e^{-\lambda t}\sum_{w=1}^{i-1}\left\{(t\lambda)^{i-1-w}\left[\alpha_{i-w,i} + \sum_{k=0}^{w-1}\left(\alpha_{i-k,i}\prod_{j=i-w}^{i-1-k} j\right)\right]\right\}$$

Ne consegue:

$$(1) = -e^{-\lambda t}\sum_{m=1}^{i}\left(\alpha_{m,i}\int_t^{+\infty} g(z)e^{-\lambda z}dz\right) =$$

$$= -\alpha_{i,i}(t\lambda)^{i-1}e^{-2\lambda t} - e^{-2\lambda t}\sum_{w=1}^{i-1}\left\{(t\lambda)^{i-1-w}\left[\alpha_{i-w,i} + \sum_{k=0}^{w-1}\left(\alpha_{i-k,i}\prod_{j=i-w}^{i-1-k} j\right)\right]\right\}$$

Sviluppo di (2):

$$(2) = -e^{-2\lambda t}\int_0^t\left(\sum_{m=1}^{i}\alpha_{m,i}g_m(z)e^{-\lambda z}\right)e^{\lambda z}dz =$$

$$= -e^{-2\lambda t}\left[\sum_{m=1}^{i}\left(\alpha_{m,i}\int_0^t g_m(z)dz\right)\right] =$$

$$= -e^{-2\lambda t}\left[\sum_{m=1}^{i}\left(\alpha_{m,i}\lambda^m\int_0^t z^{m-1}dz\right)\right] =$$

$$= -e^{-2\lambda t}\left[\sum_{m=1}^{i}\left(\tfrac{\alpha_{m,i}}{m}\lambda^m t^m\right)\right] = -e^{-2\lambda t}\sum_{m=1}^{i}\left[\tfrac{\alpha_{m,i}}{m}(\lambda t)^m\right]$$

Sommando (1) e (2) si ottiene:

$$\mathbb{P}[T_i \le t] = 1 + (1) + (2) =$$

$$= 1 - \alpha_{i,i}(\lambda t)^i e^{-2\lambda t} - \left(\alpha_{i,i} + \tfrac{\alpha_{i-1,i}}{i-1}\right)(\lambda t)^{i-1}e^{-2\lambda t} - e^{-2\lambda t}\sum_{w=1}^{i-2}\left\{(\lambda t)^w\left[\tfrac{\alpha_{w,i}}{w} + \alpha_{w+1,i} + \sum_{k=0}^{i-w-k}\left(\alpha_{i-k,i}\prod_{j=w+1}^{i-1-k} j\right)\right]\right\} -$$

$$e^{-2\lambda t}\left[\alpha_{1,i} + \sum_{k=0}^{i-2}\left(\alpha_{i-k,i}\sum_{j=2}^{i-1-k} j\right)\right]$$

Si potrà costruire una tabella dei coeficienti dell'integrale della funzione di probabilità cumulata dei tempi di interarrivo:

fomula\n. pezzo in ingresso	0	1	2	3	4	...
$(t\lambda)^0 e^{-2\lambda t}(t\lambda)^0 e^{-2\lambda t}$	-1	-1	-1	$M_{0,3}$	$M_{0,40}$	
$(t\lambda)^1 e^{-2\lambda t}$		-1	-1	$M_{1,3}$	$M_{1,4}$	
$(t\lambda)^2 e^{-2\lambda t}$			$-\frac{1}{4}$	$M_{2,3}$	$M_{2,4}$	
$(t\lambda)^3 e^{-2\lambda t}$				$M_{3,3}$	$M_{3,4}$	
$(t\lambda)^4 e^{-2\lambda t}$					$M_{4,4}$	
...						

Dove:

- $\forall y,\ M_{0,y} = -\left\{\alpha_{1,y} + \sum_{k=0}^{y-2}\left[\alpha_{y-k,y}\left(\prod_{j=2}^{y-1-k} j\right)\right]\right\}$;
- $\forall\ 1 \leq x \leq y-2,\ M_{x,y} = -\left\{\frac{\alpha_{x,y}}{x} + \alpha_{x+1,y} + \sum_{k=0}^{y-x-2}\left[\alpha_{y-k,y}\left(\prod_{j=x+1}^{y-1-k} j\right)\right]\right\}$;
- $M_{y-1,y} = -\left(\alpha_{y,y} + \frac{\alpha_{y-1,y}}{y-1}\right)$;
- $M_{y,y} = -\frac{\alpha_{y,y}}{y}$;
- con x e y, solo per questa tabella, saranno i contatori di riga e colonna a partire da 0;
- x rappresenta, oltre al numero della riga, anche il coeficiente per il quale viene elevato il $(t\lambda)$.

Si ottiene quindi:

$$\mathbb{P}\left[T_i \leq t\right] = 1 - \sum_{x=0}^{i}\left[M_{x,i}\,(\lambda t)^x\, e^{-2\lambda t}\right]$$

Derivando su t si avrà:

$$l_i(t) = \frac{d\mathbb{P}[T_i \leq t]}{dt} = \sum_{x=0}^{i-1}\left[\left(-2M_{x,i} + (x+1)\,M_{x+1,i}\right)t^x\lambda^{x+1}\right]e^{-2\lambda t} - 2M_{i,i}t^i\lambda^{i+1}e^{-2\lambda t} = \sum_{x=0}^{i}\left(a_{x,i}t^x\lambda^{x+1}\right)e^{-2\lambda t}$$

Si potrà costruire la tabella dei coeficienti della funzione di densità di probabilità dei tempi di interarrivo:

funzione \ pezzo	0	1	2	3	4	...
(x=0) $t^x\lambda^{x+1}e^{-2\lambda t}$	2	1	1	$a_{0,2}$	$a_{0,3}$	
(x=1) $t^x\lambda^{x+1}e^{-2\lambda t}$		2	$\frac{3}{2}$	$a_{1,2}$	$a_{1,3}$	
(x=2) $t^x\lambda^{x+1}e^{-2\lambda t}$			$\frac{1}{2}$	$a_{2,2}$	$a_{2,3}$	
(x=3) $t^x\lambda^{x+1}e^{-2\lambda t}$				$a_{3,2}$	$a_{3,3}$	
(x=4) $t^x\lambda^{x+1}e^{-2\lambda t}$					$a_{4,3}$	
...						

Dove:

- $\forall a_{x,y}/x < y,\ a_{x,y} = -2M_{x,y} + M_{x+1,y}(x+1)$;
- $\forall a_{x,y}/x = y,\ a_{x,y} = -2M_{y,y}$.

Calcolo del generico tempo medio $\bar{t}_i$:

$$\bar{t}_i = \int_0^{+\infty} t l_i(t)dt = \sum_{x=0}^{i}\left[a_{x,i}\int_0^{\infty}(t\lambda)^{x+1}e^{-2\lambda t}dt\right]$$

Si noti che:

$\int_0^{+\infty}(t\lambda)^{x+1}e^{-2\lambda t}dt = \frac{1}{\lambda}(\prod_{j=1}^{x+1} j)\left(\frac{1}{2}\right)^{x+2} \Rightarrow$

$\Longrightarrow \bar{t}_i = \int_0^{\infty} t l_i(t)dt = \sum_{x=0}^{i}\left[a_{x,i}\int_0^{\infty}(t\lambda)^{x+1}e^{-2\lambda t}dt\right] =$

$= \frac{1}{\lambda}\sum_{x=0}^{i}\left[a_{x,i}(\prod_{j=1}^{x+1} j)\left(\frac{1}{2}\right)^{x+2}\right]$

Calcolo della varianza:

$E_i[t^2] = \int_0^{+\infty} t^2 l_i(t)\,dt = \sum_{x=0}^{i}\left[a_{x,i}\int_0^{+\infty} t^{x+2}\lambda^{x+1}e^{-2\lambda t}dt\right] =$

$= \frac{1}{\lambda^2}\sum_{x=0}^{i}\left[a_{x,i}(\prod_{j=1}^{x+2} j)\left(\frac{1}{2}\right)^{x+3}\right]$

$Var_i[t] = E_i[t^2] - \overline{t_i}^2$

La media degli intertempi di arrivo medi specifici delle singole coppie sarà:

$\overline{t_N} = (\sum_{i=1}^{N}\bar{t}_i)/N,$

La media delle varianze degli intertempi di arrivo specifici delle singole coppie sarà:

$\delta^2_{ta_N} = \frac{1}{N}\sum_{i=1}^{N} Var_i[t]$

Sarà adesso possibile la formulazione di una funzione globale di distribuzione di probabilità degli intertempi di arrivo.

Si ricordi che la funzione di densità di probabilità degli intertempi di ingresso specifica dell'i-esima coppia è:

$l_i(t) = \sum_{m=0}^{i}\left[a_{m,i}g_{m+1}(t)\,e^{-2\lambda t}dt\right]$

dove:

$g_{m+1}(t) = t^m\lambda^{m+1}.$

Come per i tempi di attesa, anche in questo caso si sceglie di procedere calcolando la media delle funzioni di probabilità cumulata degli intertempi di arrivo specifiche delle singole coppie su un campo $0 \le t \le t_*$:

$P_N[TA_g < t_*] = \frac{1}{N}\int_0^{t_*}\sum_{i=0}^{N-1}\left[\sum_{m=0}^{i}\left(a_{m,1}g_{m+1}(t)\,e^{-2\lambda t}dt\right)\right] =$

$= \frac{1}{N}\sum_{i=0}^{N-1}\sum_{m=0}^{i}\left(a_{m,1}\int_0^{t_*} g_{m+1}(t)\,e^{-2\lambda t}dt\right)$

Si noti che:

$\int_0^{t_*} g_{m+1}(t)\,e^{-2\lambda t}dt = \int_0^{t_*} t^m\lambda^{m+1}e^{-2\lambda t}dt =$

$= -\frac{1}{2}t_*^m\lambda^m e^{-2\lambda t_*} - \frac{1}{4}t_*^{m-1}\lambda^{m-1}e^{-2\lambda t_*} - ... - \frac{m!}{k!}\frac{1}{2^{m-k+1}}t^k\lambda^k e^{-2\lambda t_*} - ... - \frac{m!}{2^{m+1}}e^{-2\lambda t_*} + \frac{m!}{2^{m+1}}$

Quindi:

$P_N[TA_g < t_*] = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{m=0}^{i}\left\{a_{m,i}\left[\frac{m!}{2^{m+1}} - \frac{m!}{2^{m+1}}e^{-2\lambda t_*} - \sum_{k=1}^{m}\left(\frac{m!}{k!}\frac{1}{2^{m-k+1}}t^k\lambda^k e^{-2\lambda t_*}\right)\right]\right\} =$

$= \frac{1}{N}\sum_{i=0}^{N-1}\left\{a_{0,i}\left(\frac{1}{2} - \frac{1}{2}e^{-2\lambda t_*}\right) + \sum_{m=1}^{i}\left[a_{m,i}\left(\frac{m!}{2^{m+1}} - \frac{m!}{2^{m+1}}e^{-2\lambda t_*} - \sum_{k=1}^{m}\left(\frac{m!}{k!}\frac{1}{2^{m-k+1}}t^k\lambda^k e^{-2\lambda t_*}\right)\right)\right]\right\}$

Si definiscono i coeficienti:

$A_{0,N} = -\frac{1}{N}\left\{\frac{a_{0,0}}{2} + \sum_{i=1}^{N-1}\left[\frac{a_{0,i}}{2} + \sum_{m=1}^{i}\left(\frac{m!}{2^{m+1}}a_{m,i}\right)\right]\right\}$, coeficiente di $e^{-2\lambda t_*}$ per k=0;

$A_{1,N} = -\frac{1}{N}\sum_{i=1}^{N-1}\sum_{m=1}^{i}\left(\frac{m!}{2^m}a_{m,i}\right)$, coeficiente di $t_*\lambda e^{-2\lambda t_*}$ per k=1;

$A_{k,N} = -\frac{1}{N}\sum_{i=k}^{N-1}\sum_{m=k}^{i}\left(\frac{1}{2^{m-k+1}}\frac{m!}{k!}a_{m,i}\right)$, coeficiente di $t_*^k\lambda^k e^{-2\lambda t_*}$ per $1 \leq k \leq N-1$;

Si potrà quindi scrivere:

$$P_N\left[TA_g < t_*\right] = \sum_{k=0}^{N-1}\left(A_{k,N}t_*^k\lambda^k e^{-2\lambda t_*}\right) + Costante$$

Derivando si otterrà la funzione di densità della probabilità degli intertempi di ingresso globale su un lotto:

$$l_{g,N}\left(t\right) = \frac{d}{dt_*}P_N\left[TA_g < t_*\right] = \sum_{k=0}^{N-1}\left(\gamma_{k,N}g_{k+1}\left(t\right)e^{-2\lambda t}\right)$$

Dove:

$\gamma_{k,N} = \left[-2A_{k,N} + \left(i+1\right)A_{i+1,N}\right]$ per $0 \leq k \leq N-2$;

$\gamma_{N-1,N} = -2A_{N-1,N}$ per $k = N-1$.

Calcolo dell'intertempo di ingresso medio utilizzando la funzione globale:

$$\overline{ta}_{g,N} = \int_0^\infty tl_{g,N}\left(t\right)dt = \sum_{k=0}^{N-1}\left(\gamma_{k,N}\int_0^\infty tg_{k+1}\left(t\right)e^{-2\lambda t}dt\right) = \frac{1}{\lambda}\sum_{k=0}^{N-1}\left[\gamma_{k,N}\frac{(k+1)!}{2^{k+2}}\right]$$

Calcolo della varianza utilizzando la funzione globale:

$$Var\left[ta_{g,N}\right] = \int_0^\infty\left(t-\overline{ta}_{g,N}\right)^2 l_{g,N}\left(t\right)dt = \int_0^\infty\left(t-\overline{ta}_{g,N}\right)^2\sum_{k=0}^{N-1}\left(\gamma_{k,N}g_{k+1}\left(t\right)e^{-2\lambda t}\right)dt =$$

$$= \sum_{k=0}^{N-1}\left[\gamma_{k,N}\int_0^\infty\left(t-\overline{ta}_{g,N}\right)^2 g_{k+1}\left(t\right)e^{-2\lambda t}dt\right] = \sum_{k=0}^{N-1}\left[\gamma_{k,N}\left(\int_0^\infty t^{k+2}\lambda^{k+1}e^{-2\lambda t}dt + \overline{ta}_{g,N}^2\int_0^\infty t^k\lambda^k e^{-2\lambda t}dt + 2\overline{ta}_{g,N}\int_0^\infty t^{k+1}\right.\right.$$

$$= \sum_{k=0}^{N-1}\left\{\gamma_{k,N}\left[\left(\frac{1}{\lambda}\right)^2\frac{(k+2)!}{2^{k+3}} + \overline{ta}_{g,N}^2\frac{k!}{2^{k+1}} - 2\frac{\overline{ta}_{g,N}}{\lambda}\frac{(k+1)!}{2^{k+2}}\right]\right\}$$

Intertempi di uscita

Restano da modellizzare gli intertempi di uscita.

Per la prima coppia si avrà:

$$P\left[K_i < k^*\right] = \int_0^{k_*}\lambda e^{-\lambda y}\int_0^y\lambda e^{-\lambda x}dxdy + \int_0^{k_*}\lambda e^{-\lambda x}\int_0^x\lambda e^{-\lambda y}dydx =$$

$$= \int_0^{k_*}\lambda e^{-\lambda y}(1-e^{-\lambda y})dy + \int_0^{k_*}\lambda e^{-\lambda x}(1-e^{-\lambda x})dx = 2\left(\frac{1}{2} - e^{-\lambda k_*} + \frac{1}{2}e^{-2\lambda k_*}\right) =$$

$$= 1 - 2e^{-\lambda k_*} + e^{-2\lambda k_*}$$

Quindi:

$$h_0\left(k\right) = 2\lambda e^{-\lambda k} - 2\lambda e^{-2\lambda k}$$

Valore attesa e varianza saranno:

$$\overline{k}_0 = \int_0^{+\infty} k\left(2\lambda e^{-\lambda k} - 2\lambda e^{-2\lambda k}\right) dk = \int_0^{+\infty} 2k\lambda e^{-\lambda k} dk - \int_0^{+\infty} 2\lambda k e^{-2\lambda k} dk =$$

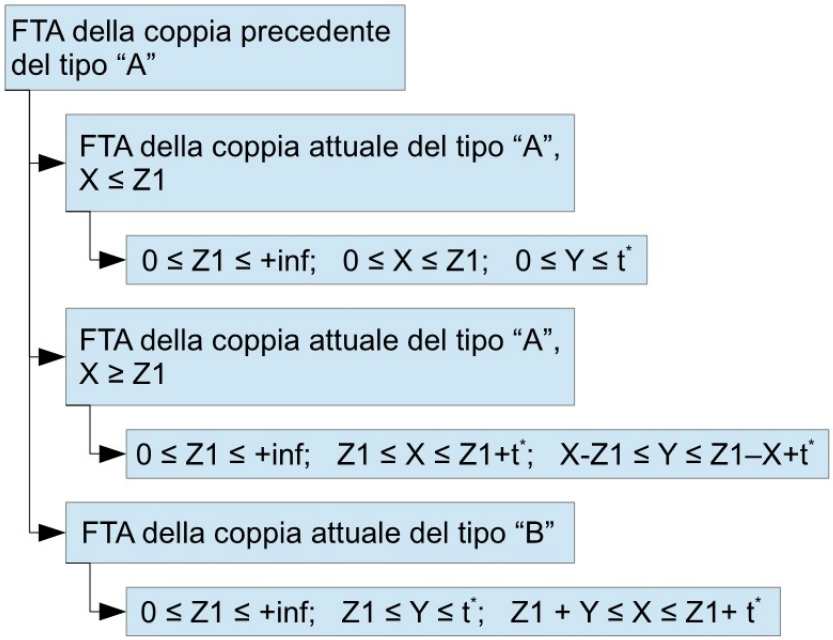
$$= \int_0^{+\infty} 2e^{-\lambda k} dk - \int_0^{+\infty} e^{-2\lambda k} dk = \frac{2}{\lambda} - \frac{1}{2\lambda} = \frac{3}{2}\frac{1}{\lambda}$$

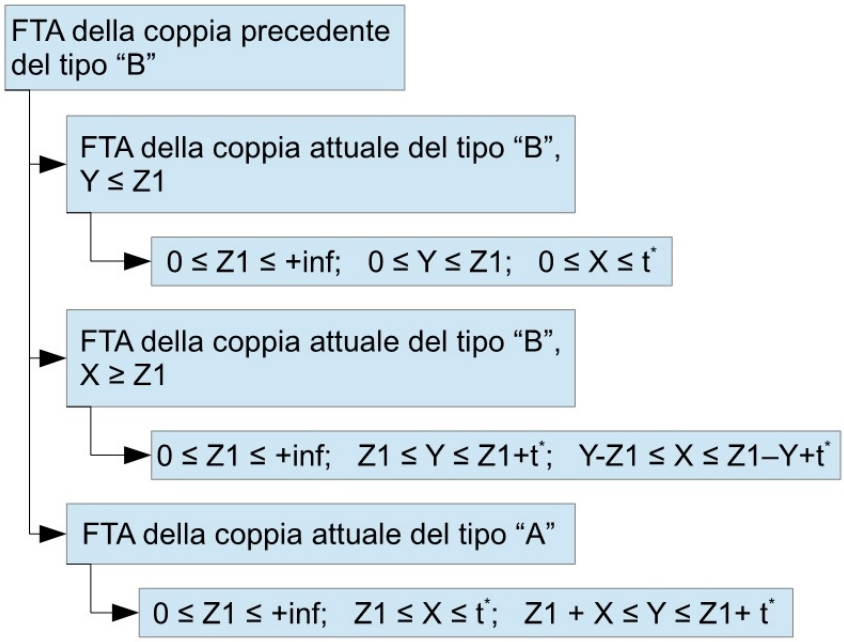
$$E\left[k^2\right] = \int_0^{+\infty} k^2\left(2\lambda e^{-\lambda k} - 2\lambda e^{-2\lambda k}\right) dk = \int_0^{+\infty} 2k^2\lambda e^{-\lambda k} dk - \int_0^{+\infty} 2\lambda k^2 e^{-2\lambda k} dk =$$

$$= \int_0^{+\infty} \frac{4}{\lambda} e^{-\lambda k} dk - \int_0^{+\infty} \frac{1}{\lambda} e^{-2\lambda k} dk = \frac{4}{\lambda^2} - \frac{1}{2\lambda^2} = \frac{7}{2}\frac{1}{\lambda^2}$$

$$Var[k_0] = \frac{7}{2}\frac{1}{\lambda^2} - \frac{9}{4}\frac{1}{\lambda^2} = \frac{5}{4}\frac{1}{\lambda^2}$$

Come nel caso precedente, per tutte le coppie successive alla prima, gli intertempi di uscita saranno influenzati dai tempi di attesa.





Otterremo sei integrali la cui somma sarà la funzione di probabilità cumulata.

Sviluppo del primo integrale del ramo X:

$$(A) = \int_0^{\infty} f_i(z) \int_0^{z} \lambda e^{-\lambda x} \int_0^{k} \lambda e^{-\lambda y} dy dx dz = \int_0^{\infty} f_i(z) \int_0^{z} \lambda e^{-\lambda x} \left(1 - e^{-\lambda k}\right) dx dz =$$

$$= \left(1 - e^{-\lambda k}\right) \int_0^{\infty} f_i(z) \left(1 - e^{-\lambda z}\right) dz = \left(1 - e^{-\lambda k}\right) - \left(1 - e^{-\lambda k}\right) \int_0^{\infty} f_i(z) e^{-\lambda z} dz$$

Si ricordi che:

$$f_i(z) = \sum_{m=1}^{i} \left(\alpha_{m,i} g_m(z) e^{-\lambda z}\right) \text{ con } g_m(z) = z^{m-1}\lambda^m$$

Quindi:

$$\int_0^{\infty} f_i(z) e^{-\lambda z} dz = \sum_{m=1}^{i} \left[\alpha_{m,i} \int_0^{\infty} g_m(z) e^{-2\lambda z} dz\right]$$

Si definisca:

$$\beta_i = \int_0^{\infty} f_i(z) e^{-\lambda z} dz = \frac{\alpha_{1,i}}{2} + \sum_{m=2}^{i} \left(\alpha_{m,i} \frac{(m-1)!}{2^m}\right)$$

Si otterrà:

$$(A) = \left(1 - e^{-\lambda k}\right) - \left(1 - e^{-\lambda k}\right) \int_0^{\infty} f_i(z) e^{-\lambda z} dz = \left(1 - e^{-\lambda k}\right) - \left(1 - e^{-\lambda k}\right) \beta_i \Rightarrow$$

$$\Rightarrow (A) = (1 - \beta_i) - (1 - \beta_i) e^{-\lambda k}$$

Sviluppo del secondo integrale del ramo X:

$$(B) = \int_0^{\infty} f_i(z) \int_z^{z+k} \lambda e^{-\lambda x} \int_{x-z}^{k-x+z} \lambda e^{-\lambda y} dy dx dz = \int_0^{\infty} f_i(z) \int_z^{z+k} \lambda e^{-\lambda x} \left(e^{\lambda z} e^{-\lambda x} - e^{-\lambda k} e^{-\lambda z} e^{\lambda x}\right) =$$

$$= \int_0^{\infty} f_i(z) \left[e^{\lambda z} \int_z^{z+k} \lambda e^{-2\lambda x} dx - e^{-\lambda k} e^{-\lambda z} \lambda \int_z^{z+k} dx\right] dz = \int_0^{\infty} f_i(z) \left[e^{\lambda z} \left(\frac{e^{-2\lambda z} - e^{-2\lambda k} e^{-2\lambda z}}{2}\right) - k\lambda e^{-\lambda k} e^{-\lambda z} \lambda\right] dz =$$

$$= \int_0^{\infty} f_i(z) \left[\frac{e^{-\lambda z} - e^{-2\lambda k} e^{-\lambda z}}{2} - k\lambda e^{-\lambda k} e^{-\lambda z} \lambda\right] dz = \left(\frac{1}{2} - \frac{e^{-2\lambda k}}{2} - k\lambda e^{-\lambda k}\right) \beta_i$$

Sviluppo del terzo integrale del ramo X:

$$(C) = \int_0^{\infty} f_i(z) \int_0^{k} \lambda e^{-\lambda y} \int_{z+y}^{z+k} \lambda e^{-\lambda x} dx dy dz = \int_0^{\infty} f_i(z) \int_0^{k} \lambda e^{-\lambda y} \left(e^{-\lambda z} e^{-\lambda y} - e^{-\lambda z} e^{-\lambda k}\right) dy dz =$$

$$= \int_0^{\infty} f_i(z) \left[e^{-\lambda z} \int_0^{k} \lambda e^{-2\lambda y} dy - e^{-\lambda z} e^{-\lambda k} \int_0^{k} \lambda e^{-\lambda y} dy\right] dz =$$

$$= \int_0^{\infty} f_i(z) \left[\frac{e^{-\lambda z}}{2} \left(1 - e^{-2\lambda k}\right) - e^{-\lambda z} e^{-\lambda k} \left(1 - e^{-\lambda k}\right)\right] dz =$$

$$= \int_0^{\infty} f_i(z) e^{-\lambda z} \left[\frac{1 + e^{-2\lambda k} - 2e^{-\lambda k}}{2}\right] dz = \left[\frac{1 + e^{-2\lambda k}}{2} - e^{-\lambda k}\right] \beta_i$$

Gli integrali sul ramo Y saranno simmetrici e daranno risultato identico a quelli del ramo X.

Siccome i due rami hanno entrambi un peso pari al 50%, la funzione di probabilità cumulata sarà:

$$P[K_i < k^*] = 1 - e^{-\lambda k^*} - \beta_i k^* \lambda e^{-\lambda k^*}$$

Derivando si otterrà la funzione di densità della probabilità:

$$h_i(k) = \lambda e^{-\lambda k} - \beta_i \lambda e^{-\lambda k} + \beta_i k \lambda^2 e^{-\lambda k}$$

Il valore medio dell'intertempo di uscita dell'i-esima coppia rispetto alla precedente sarà:

$$E[k_i] = \int_0^{\infty} k h_i(k) dk = \int_0^{\infty} (1 - \beta_i) k\lambda e^{-\lambda k} dk + \int_0^{\infty} \beta_i k^2 \lambda^2 e^{-\lambda k} dk =$$

$= (1-\beta_i)\,\frac{1}{\lambda} + \beta_i \frac{2}{\lambda} = (1+\beta_i)\,\frac{1}{\lambda}$

$\overline{k_i} = (1+\beta_i)\,\frac{1}{\lambda}$

Calcolo della varianza:

$E\left[k_i^2\right] = \int_0^\infty k^2 h_i\left(k\right) dk = \int_0^\infty \left(1-\beta_i\right) k^2 \lambda e^{-\lambda k} dk + \int_0^\infty \beta_i k^3 \lambda^2 e^{-\lambda k} dk =$

$= \left(1-\beta_i\right) \frac{2}{\lambda^2} + \beta_i \frac{6}{\lambda^2} = \left(2+4\beta_i\right) \frac{1}{\lambda^2}$

$Var\left[k_i\right] = \delta_{k_i}^2 = E\left[k_i^2\right] - \overline{k_i}^2 = \left(2+4\beta_i\right) \frac{1}{\lambda^2} - \overline{k_i}^2$

La media dei valori medi degli intertempi di uscita specifici delle singole coppie sarà:

$\overline{tu}_N = \frac{1}{N} \sum_{i=0}^{N-1} \overline{k_i}$

La media delle varianze degli intertempi di uscita specifici delle singole coppie sarà:

$\delta_{tu,N}^2 = \frac{1}{N} \sum_{i=0}^{N-1} \delta_{k_i}^2$

Si può adesso calcolare la funzione di densità della probabilità degli intertempi di uscita globale.

Si procederà come visto in precedenza.

$\mathbb{P}_{g,N}\left[T < t_*\right] = \frac{1}{N} \sum_{i=0}^{N-1} \left[\int_0^{t_*} \left(\lambda e^{-\lambda k} - \beta_i \lambda e^{-\lambda k} + \beta_i k \lambda^2 e^{-\lambda k}\right) dk\right] =$

$= \frac{1}{N} \sum_{i=0}^{N-1} \left[\left(1-\beta_i\right) \int_0^{t_*} \lambda e^{-\lambda k} dk + \beta_i \int_0^{t_*} k \lambda^2 e^{-\lambda k} dk\right] =$

$= \frac{1}{N} \sum_{i=0}^{N-1} \left[\left(1-\beta_i\right)\left(1-e^{-\lambda t_*}\right) + \beta_i \left(-t_* \lambda e^{-\lambda t_*} + \int_0^{t_*} \lambda e^{-\lambda k} dk\right)\right] =$

$= \frac{1}{N} \sum_{i=0}^{N-1} \left[\left(1-\beta_i\right)\left(1-e^{-\lambda t_*}\right) - \beta_i t_* \lambda e^{-\lambda t_*} + \beta_i \left(1-e^{-\lambda t_*}\right)\right] =$

$= \frac{1}{N} \sum_{i=0}^{N-1} \left[1 - e^{-\lambda t_*} - \beta_i t_* \lambda e^{-\lambda t_*}\right] = 1 - e^{-\lambda t_*} - \left(\sum_{i=0}^{N-1} \beta_i\right) \frac{t_* \lambda e^{-\lambda t_*}}{N}$

Derivando si ottiene:

$h_{g,N}\left(t\right) = \left(1 - \sum_{i=0}^{N-1} \frac{\beta_i}{N}\right) \lambda e^{-\lambda t} + \left(\sum_{i=0}^{N-1} \frac{\beta_i}{N}\right) t \lambda^2 e^{-\lambda t}$

Calcolo della media:

$\overline{tu}_{g,N} = \int_0^\infty t \left(1 - \sum_{i=0}^{N-1} \frac{\beta_i}{N}\right) \lambda e^{-\lambda t} dt + \int_0^\infty \left(\sum_{i=0}^{N-1} \frac{\beta_i}{N}\right) t^2 \lambda e^{-\lambda t} dt =$

$= \left(1 + \sum_{i=0}^{N-1} \frac{\beta_i}{N}\right) \frac{1}{\lambda}$

Calcolo della varianza:

$Var\left[tu_g\right] = \int_0^\infty \left(t^2 + \overline{tu}_{g,N}^2 - 2t\overline{tu}_{g,N}\right) \left(1 - \sum_{i=0}^{N-1} \frac{\beta_i}{N}\right) \lambda e^{-\lambda t} dt + \int_0^\infty \left(t^2 + \overline{tu}_{g,N}^2 - 2t\overline{tu}_{g,N}\right) \left(\sum_{i=0}^{N-1} \frac{\beta_i}{N}\right) t \lambda^2 e^{-\lambda t} dt$

Calcolo del primo addendo:

$(A) = \int_0^\infty t^2 \left[\left(1 - \sum_{i=0}^{N-1} \frac{\beta_i}{N}\right) \lambda e^{-\lambda t} + \left(\sum_{i=0}^{N-1} \frac{\beta_i}{N}\right) t \lambda^2 e^{-\lambda t}\right] dt =$

$= \left(1 - \sum_{i=0}^{N-1} \frac{\beta_i}{N}\right) \int_0^\infty t^2 \lambda e^{-\lambda t} dt + \left(\sum_{i=0}^{N-1} \frac{\beta_i}{N}\right) \int_0^\infty t^3 \lambda^2 e^{-\lambda t} dt =$

$$= 2\left(1-\sum_{i=0}^{N-1}\frac{\beta_i}{N}\right)\left(\frac{1}{\lambda}\right)^2 + 6\left(\sum_{i=0}^{N-1}\frac{\beta_i}{N}\right)\left(\frac{1}{\lambda}\right)^2$$

Calcolo del secondo addendo:

$$(B) = \int_0^\infty \overline{tu}_{g,N}^2 \left[\left(1-\sum_{i=0}^{N-1}\frac{\beta_i}{N}\right)\lambda e^{-\lambda t} + \left(\sum_{i=0}^{N-1}\frac{\beta_i}{N}\right) t\lambda^2 e^{-\lambda t}\right] dt =$$

$$= \overline{tu}_{g,N}^2 \left[\left(1-\sum_{i=0}^{N-1}\frac{\beta_i}{N}\right) + \left(\sum_{i=0}^{N-1}\frac{\beta_i}{N}\right)\right] dt = \overline{tu}_{g,N}^2$$

Calcolo del terzo addendo:

$$(C) = -\int_0^\infty 2t\overline{tu}_{g,N} \left[\left(1-\sum_{i=0}^{N-1}\frac{\beta_i}{N}\right)\lambda e^{-\lambda t} + \left(\sum_{i=0}^{N-1}\frac{\beta_i}{N}\right) t\lambda^2 e^{-\lambda t}\right] dt =$$

$$= -2\overline{tu}_{g,N} \left[\left(1-\sum_{i=0}^{N-1}\frac{\beta_i}{N}\right)\int_0^\infty t\lambda e^{-\lambda t} + \left(\sum_{i=0}^{N-1}\frac{\beta_i}{N}\right)\int_0^\infty t^2\lambda^2 e^{-\lambda t}\right] dt =$$

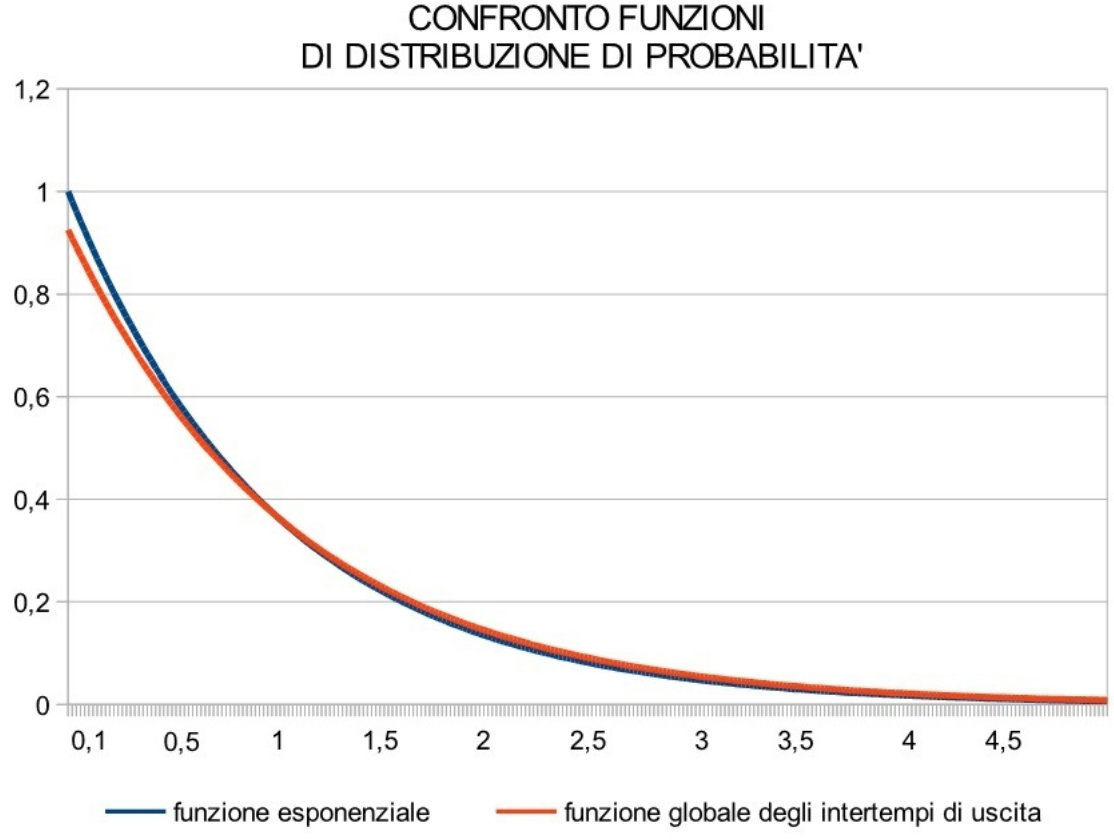
$$= -2\overline{tu}_{g,N} \left[\left(1-\sum_{i=0}^{N-1}\frac{\beta_i}{N}\right)\left(\frac{1}{\lambda}\right) + 2\left(\sum_{i=0}^{N-1}\frac{\beta_i}{N}\right)\left(\frac{1}{\lambda}\right)\right] dt =$$

$$= -2\overline{tu}_{g,N}\overline{tu}_{g,N} = -2\overline{tu}_{g,N}^2$$

Quindi:

$$Var\left[tu_g\right] = (A) + (B) + (C) = 2\left[1 + 2\left(\sum_{i=0}^{N-1}\frac{\beta_i}{N}\right)\right]\left(\frac{1}{\lambda}\right)^2 - \overline{tu}_{g,N}^2$$

Qui di seguito è riportato il grafico di una funzione di distribuzione dei tempi di interuscita globale calcolata su un lotto da 200 coppie. Si può osservare come abbia un andamento pressoché identico alla funzione esponenziale. Modellizzando su lotti la cui dimensione tende ad infinito sarà possibile constatare che questa tendenza si rafforza, sia per gli intertempi d'uscita che d'ingresso.



WIP, CT e TH

Nella sezione precedente sono state formulate le medie dei valori medi e delle varianze dei tempi caratterizzanti ogni singola coppia di un generico lotto N e i valori medi e la varianze globali dei tempi di interarrivo, interuscita e attesa.

Restano da calcolare i livelli di WIP.

Si introduca il concetto di «turno», definito come il tempo intercorrente tra l'arrivo degli FTA di due coppie successive. Si avranno N turni indicizzati da 0 a N-1.

Sarà osservabile per tutti i turni:

- la presenza di un incremento unitario del livello di WIP nel loro istante iniziale;
- la variabilità del valore finale del WIP tra 0 e $i+1$ al termine del turno i-esimo, in quanto in quell'istante saranno potuti entrare nel sistema un massimo di $i+1$ componenti complementari.

Per ragioni pratiche, si sceglie di:

- approssimare il livello medio di WIP durante il turno al valore che questo avrà al termine dello stesso;
- indicare d'ora in avanti con X (e quindi sue funzioni caratterizzanti) la tipologia di componenti già presenti in coda, mentre con Y i complementari di cui sono in attesa.

Si desidera ottenere una tabella come la seguente:

Turni\WIP	0	1	2	3	4	...
0	$W_{0,0}$	$W_{0,1}$				
1	$W_{1,0}$	$W_{1,1}$	$W_{1,2}$			
2	$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$		
3	$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$	$W_{3,4}$	
...						

dove con $W_{i,j}$ si indica la probabilità di avere al termine del turno i un livello di WIP pari a j.

Come visto nella sezione precedente, $f_{i+1}(z)$ definisce la funzione di densità di probabilità del tempo che il componente x_i dovrà attendere per il suo complementare. Di conseguenza si può caratterizzare la probabilità di avere WIP nullo al termine del turno i-esimo come la probabilità di avere il tempo di interarrivo di x_{i+1} superiore al tempo di interarrivo del suo complementare.

Si definisca $P_i[0]$ come la probabilità di avere un livello di WIP al termine dell'i-esimo turno pari a 0.

Si avrà:

$P_i[0] = \int_0^{+\infty} f_{i+1}(z) \int_z^{+\infty} \lambda e^{-\lambda x_{i+1}} \int_{x_{i+1}-z}^{+\infty} \lambda e^{-\lambda y_{i+1}} dy_{i+1} dx_{i+1} dz + \int_0^{+\infty} f_{i+1}(z) \int_z^{+\infty} \lambda e^{-\lambda x_{i+1}} \int_0^{x_{i+1}-z} \lambda e^{-\lambda y_{i+1}} dy_{i+1}$

$= \int_0^{+\infty} f_{i+1}(z) \int_z^{+\infty} \lambda e^{-\lambda x_{i+1}} \int_0^{+\infty} \lambda e^{-\lambda y_{i+1}} dy_{i+1} dx_{i+1} dz =$

$= \int_0^{+\infty} f_{i+1}(z) \int_z^{+\infty} \lambda e^{-\lambda x_{i+1}} dx_{i+1} dz$

Si avrà $W_{i,0} = P_i[0]$ per ogni i.

Ne conseguono i seguenti vincoli di sistema:

- $W_{0,0} = P_0[0]$, $W_{0,1} = 1 - P_0[0]$;

- $W_{1,0} = P_1[0]$, $W_{1,1} + W_{1,2} = 1 - P_1[0]$;
- $W_{2,0} = P_2[0]$, $W_{2,1} + W_{2,2} + W_{2,3} = 1 - P_2[0]$;
- ...
- $W_{i,0} = P_i[0]$, $\sum_{j=1}^{i+1} W_{i,j} = 1 - P_i[0]$;

Le probabilità $W_{i,j}$ del turno i, ad eccezione delle varie $W_{i,0}$ appena analizzate, saranno influenzate dalle probabilità $W_{i-1,j}$ (con $j \in [j-1; i]$) del turno precedente, in quanto nel tempo intercorrente tra i due potrà entrare al massimo un componente aggiuntivo del tipo X, ma tanti complementari quanti componenti siano presenti in coda.

Si formulino:

- $W_{1,1} = W_{0,0} * F_{1,0,1} + W_{0,1} * F_{1,1,1}$
- $W_{1,2} = W_{0,1} * F_{1,1,2}$
- $W_{2,1} = W_{1,0} * F_{2,0,1} + W_{1,1} * F_{2,1,1} + W_{1,2} * F_{2,2,1}$
- $W_{2,2} = W_{1,1} * F_{2,1,2} + W_{1,1} * F_{2,1,2}$
- $W_{2,3} = W_{1,2} * F_{2,2,3}$
- ...

Per convenienza si considerino le varie $F_{i,a,b}$ come le probabilità di arrivo di $[(a+1) - b]$ unità complementari in un periodo di dimensione k^*, rappresentante l'intervallo di tempo intercorrente tra gli arrivi delle unità x_{i+1}e x_{i+2}, che sono gli eventi caratterizzanti gli estremi del turno i-esimo.

Si definisca:

- $P_i[K < k^*]$ come la funzione di probabilità cumulata di arrivo di almeno «j» componenti in un tempo k^*;
- $h_j(k)$ come la funzione di densità della probabilità di arrivo di almeno «j» componenti in un tempo k^*.

Si avrà:

$P_1[K < k^*] = \int_0^{k^*} \lambda e^{-\lambda y} dy = 1 - e^{-\lambda k^*}$

Derivando si otterrà:

$h_1(k) = \lambda e^{-\lambda k}$

Per $j = 2$:

$P_2[K < k^*] = \int_0^{k^*} h_1(k) \int_0^{k^*-k} \lambda e^{-\lambda y} dy dk = \int_0^{k^*} h_1(k) (1 - e^{-\lambda k^*} e^{\lambda k}) dk = \int_0^{k^*} h_1(k) dk - e^{-\lambda k^*} \int_0^{k^*} h_1(k) e^{\lambda k} dk$

Si noti che $\int_0^{k^*} h_1(k) dk = P_1[K < k^*]$, quindi:

$P_2[K < k^*] = 1 - e^{-\lambda k^*} - \lambda k^* e^{-\lambda k^*}$

Derivando:

$h_2(k) = \lambda e^{-\lambda k} - \lambda e^{-\lambda k} + k\lambda^2 e^{-\lambda k} = k\lambda^2 e^{-\lambda k}$

Similmente si vedrà che:

$P_3[K < k^*] = \int_0^{k^*} h_2(k) \int_0^{k^*-k} \lambda e^{-\lambda y} dy dk = \int_0^{k^*} h_2(k) (1 - e^{-\lambda k^*} e^{\lambda k}) dk = \int_0^{k^*} h_2(k) dk - e^{-\lambda k^*} \int_0^{k^*} h_2(k) e^{\lambda k} dk$
$1 - e^{-\lambda k^*} - \lambda k^* e^{-\lambda k^*} - e^{-\lambda k} \frac{k^2 \lambda^2}{2}$

$h_3(k) = \lambda e^{-\lambda k} - \lambda e^{-\lambda k} + k\lambda^2 e^{-\lambda k} - k\lambda^2 e^{-\lambda k} + \frac{k^2\lambda^3}{2} e^{-\lambda k} = \frac{k^2\lambda^3}{2} e^{-\lambda k}$

Si può concludere che:

- $h_1(k) = \lambda e^{-\lambda k}$;
- $h_j(k) = \frac{k^{j-1}\lambda^j}{(j-1)!} e^{-\lambda k}$ per ogni $j \neq 1$.

Siccome $\int_0^{k^*} h_j(k) dk$ rappresenta la probabilità che in un tempo k^*arrivino almeno j unità, per avere una stima della probabilità di arrivo di esattamente j unità, si provvederà a eseguire una differenza tra l'integrale rappresentante la probabilità di arrivo di «almeno j unità» e quello rapprentante l'arrivo di «almeno j+1 unità».

Essendo questi integrali delle stime, non vi è certezza che la loro applicazione non porti alla violazione dei vincoli precedentemente creati.

Il problema è ovviabile tramite la seguente tabella:

Turni\WIP	0	1	2	3	4	...
0	$P_0[0]$	$1 - P_0[0]$				
1	$P_1[0]$	$Temp_{1,1}$	$Temp_{1,2}$			
2	$P_2[0]$	$Temp_{2,1}$	$Temp_{2,2}$	$Temp_{2,3}$		
3	$P_3[0]$	$Temp_{3,1}$	$Temp_{3,2}$	$Temp_{3,3}$	$Temp_{3,4}$	
...						

Per successiva facilità di scrittura, si definisca:

- $\int_a^b x = \int_a^b \lambda e^{-\lambda x} dx$
- $\int_a^b k_i = \int_a^b h_i(k) dk$
- $\int_a^b x \int_x^c k_i = \int_a^b \lambda e^{-\lambda x} \int_x^c h_i(k) dk dx$

Esplicitazione delle formule dei $Temp_{i,j}$:

- $Temp_{1,1} = P_0[0] \int_0^{+\infty} x \int_x^{+\infty} k_1 + W_{0,1} \left[\int_0^{+\infty} x \int_0^x k_1 - \int_0^{+\infty} x \int_x^{+\infty} k_2 \right]$;
- $Temp_{1,2} = W_{0,1} \int_0^{+\infty} x \int_x^{+\infty} k_1$;
- $Temp_{2,1} = P_1[0] \int_0^{+\infty} x \int_x^{+\infty} k_1 + W_{1,1} \left[\int_0^{+\infty} x \int_0^x k_1 - \int_0^{+\infty} x \int_x^{+\infty} k_2 \right] + W_{1,2} \left[\int_0^{+\infty} x \int_0^x k_2 - \int_0^{+\infty} x \int_x^{+\infty}$
- $Temp_{2,2} = W_{1,1} \int_0^{+\infty} x \int_x^{+\infty} k_1 + W_{1,2} \left[\int_0^{+\infty} x \int_0^x k_1 - \int_0^{+\infty} x \int_x^{+\infty} k_2 \right]$;
- $Temp_{2,3} = W_{1,2} \int_0^{+\infty} x \int_x^{+\infty} k_1$;
- ...

Sono utili alcune osservazioni.

Concettualmente $\int_0^{+\infty} x \int_x^{+\infty} k_1$ e $\int_0^{+\infty} k_1 \int_0^{k_1} x$ sono identici, rappresentano entrambi la probabilità che il tempo di arrivo di almeno un complementare sia superiore a quello del successivo componente già presente in coda. Si verifica che:

$\int_0^{+\infty} x \int_x^{+\infty} k_1 = \int_0^{+\infty} \lambda e^{-\lambda x} \int_x^{+\infty} h_1(k)\, dk dx = \int_0^{+\infty} \lambda e^{-\lambda x} \int_x^{+\infty} \lambda e^{-\lambda k} dk dx = \int_0^{+\infty} \lambda e^{-\lambda x} e^{-\lambda x} dx = \frac{1}{2}$

$\int_0^{+\infty} k_1 \int_0^{k_1} x = \int_0^{+\infty} \lambda e^{-\lambda k} \int_0^{k} \lambda e^{-\lambda x} dx dk = \int_0^{+\infty} \lambda e^{-\lambda k} \left(1 - e^{-\lambda k}\right) dk = \frac{1}{2}$

Quindi:

$\int_0^{+\infty} x \int_x^{+\infty} k_1 = \int_0^{+\infty} k_1 \int_0^{k_1} x = \frac{1}{2}$

Un altro componente ricorrente nelle precedenti formule è $\int_0^{+\infty} x \int_0^{x} k_i - \int_0^{+\infty} x \int_0^{x} k_{i+1}$.

Partendo da una riformulazione basata sulla precedente osservazione, è possibile riformularlo come segue:

$\int_0^{+\infty} x \int_0^{x} k_i - \int_0^{+\infty} x \int_0^{x} k_{i+1} = \int_0^{+\infty} k_i \int_{k_i}^{+\infty} x - \int_0^{+\infty} k_{i+1} \int_{k_{i+1}}^{+\infty} x =$

$= \int_0^{+\infty} h_i(k)\, e^{-\lambda k} dk - \int_0^{+\infty} h_{i+1}(k)\, e^{-\lambda k} dk =$

$= \int_0^{+\infty} \frac{k^{i-1}\lambda^i}{(i-1)!} e^{-2\lambda k} dk - \int_0^{+\infty} \frac{k^i \lambda^{i+1}}{i!} e^{-2\lambda k} dk$

Ricordando che $\int_0^{+\infty} \frac{k^i \lambda^{i+1}}{i!} e^{-2\lambda k} dk = \int_0^{+\infty} \frac{k^{i-1}\lambda^i}{2[(i-1)!]} e^{-2\lambda k} dk$, ne consegue:

$\int_0^{+\infty} \frac{k^{i-1}\lambda^i}{(i-1)!} e^{-2\lambda k} dk - \int_0^{+\infty} \frac{k^{i-1}\lambda^i}{2[(i-1)!]} e^{-2\lambda k} dk =$

$= \frac{1}{2} \int_0^{+\infty} \frac{k^{i-1}\lambda^i}{(i-1)!} e^{-2\lambda k} dk = \frac{1}{2} \int_0^{+\infty} h_i(k)\, e^{-\lambda k} dk$

Quindi:

$\int_0^{\infty} x \int_0^{x} k_i - \int_0^{\infty} x \int_0^{x} k_{i+1} = \frac{1}{2} \int_0^{\infty} h_i(k)\, e^{-\lambda k} dk$

Andando a sostituire nelle formule precedenti, si ottiene:

- $Temp_{1,1} = \frac{1}{2} W_{0,0} + \frac{1}{2} W_{0,1} \int_0^{+\infty} h_1(k)\, e^{-\lambda k} dk$;
- $Temp_{1,2} = \frac{1}{2} W_{0,1}$;
- $Temp_{2,1} = \frac{1}{2} W_{1,0} + \frac{1}{2} W_{1,1} \int_0^{+\infty} h_1(k)\, e^{-\lambda k} dk + \frac{1}{2} W_{1,2} \int_0^{+\infty} h_2(k)\, e^{-\lambda k} dk$;
- $Temp_{2,2} = \frac{1}{2} W_{1,1} + \frac{1}{2} W_{1,2} \int_0^{+\infty} h_1(k)\, e^{-\lambda k} dk$;
- $Temp_{2,3} = \frac{1}{2} W_{1,2}$;
- $Temp_{3,1} = \frac{1}{2} W_{2,0} + \frac{1}{2} W_{2,1} \int_0^{+\infty} h_1(k)\, e^{-\lambda k} dk + \frac{1}{2} W_{2,2} \int_0^{+\infty} h_2(k)\, e^{-\lambda k} dk + \frac{1}{2} W_{2,3} \int_0^{+\infty} h_3(k)\, e^{-\lambda k} dk$;
- $Temp_{3,2} = \frac{1}{2} W_{2,1} + \frac{1}{2} W_{2,2} \int_0^{+\infty} h_1(k)\, e^{-\lambda k} dk + \frac{1}{2} W_{2,3} \int_0^{+\infty} h_2(k)\, e^{-\lambda k} dk$;
- $Temp_{3,3} = \frac{1}{2} W_{2,2} + \frac{1}{2} W_{2,3} \int_0^{+\infty} h_1(k)\, e^{-\lambda k} dk$;
- $Temp_{3,4} = \frac{1}{2} W_{2,3}$
- ...

Si dovranno adesso risolvere gli integrali.

Si noti che:

- $\frac{1}{2}\int_0^{+\infty} h_i(k)\, e^{-\lambda k} dk = \frac{1}{2}\int_0^{+\infty} \frac{k^{i-1}\lambda^i}{(i-1)!} e^{-2\lambda k} dk = \frac{1}{2}\frac{(i-1)!}{(i-1)!}\left(\frac{1}{2}\right)^i = \left(\frac{1}{2}\right)^{i+1}$;
- $\frac{1}{2}\int_0^{+\infty} h_1(k)\, e^{-\lambda k} dk = \frac{1}{2}\int_0^{+\infty} \lambda e^{-2\lambda k} dk = \frac{1}{4}$;
- $\forall i \geqq 1$, si avrà $W_{i,0} = P_i[0] = \int_0^{+\infty} f_{i+1}(z) \int_z^{+\infty} \lambda e^{-\lambda x} dx dz = \int_0^{+\infty} e^{-\lambda z} \sum_{m=1}^{i+1} \left(\alpha_{m,i+1} g_m(z)\, e^{-\lambda z}\right) dz = \sum_{m=1}^{i+1}$ $\left(\int_0^{+\infty} \alpha_{m,i+1} z^{m-1} \lambda^m e^{-2\lambda z} dz\right) = \frac{1}{2}\alpha_{1,i+1} + \sum_{m=2}^{i+1} \left\{\left[\left(\frac{1}{2}\right)^m \alpha_{m,i+1}\right](m-1)!\right\}$;
- Se $i = 0$ si avrà $W_{0,0} = P_0[0] = \frac{1}{2}$

Diventa quindi:

- $Temp_{1,1} = \frac{1}{2}W_{0,0} + (\frac{1}{2})^2 W_{0,1}$;
- $Temp_{1,2} = \frac{1}{2}W_{0,1}$;
- $Temp_{2,1} = \frac{1}{2}W_{1,0} + \left(\frac{1}{2}\right)^2 W_{1,1} + \left(\frac{1}{2}\right)^3 W_{1,2}$;
- $Temp_{2,2} = \frac{1}{2}W_{1,1} + \left(\frac{1}{2}\right)^2 W_{1,2}$;
- $Temp_{2,3} = \frac{1}{2}W_{1,2}$;
- $Temp_{3,1} = \frac{1}{2}W_{2,0} + \left(\frac{1}{2}\right)^2 W_{2,1} + \left(\frac{1}{2}\right)^3 W_{2,2} + \left(\frac{1}{2}\right)^4 W_{2,3}$;
- $Temp_{3,2} = \frac{1}{2}W_{2,1} + \left(\frac{1}{2}\right)^2 W_{2,2} + \left(\frac{1}{2}\right)^3 W_{2,3}$;
- $Temp_{3,3} = \frac{1}{2}W_{2,2} + \left(\frac{1}{2}\right)^2 W_{2,3}$;
- $Temp_{3,4} = \frac{1}{2}W_{2,3}$
- ...

Riassumendo:

- $\forall i, j \neq 0, Temp_{i,j} = \sum_{a=j-1}^{i} \left[\left(\frac{1}{2}\right)^{a-(j-2)} W_{i-1,a}\right]$;
- Se $i = 0$ si avrà $W_{0,0} = P_0[0] = \frac{1}{2}$;
- $\forall i \geqq 1$, si avrà $W_{i,0} = P_i[0] = \frac{1}{2}\alpha_{1,i+1} + \sum_{m=2}^{i+1} \left\{\left[\left(\frac{1}{2}\right)^m \alpha_{m,i+1}\right](m-1)!\right\}$.

Non resta che forzare il sistema a rispettare i vincoli $\sum_{j=0}^{i+1} W_{i,j} = 1$.

Introducendo i fattori di scala S_i e pongendo $\forall i, j \neq 0, W_{i,j} = S_i Temp_{i,j}$, si otterrà:

$1 - P_i[0] = S_i \sum_{j=1}^{i+1} Temp_{i,j}$.

$\Rightarrow P_i[0] + S_i \sum_{j=1}^{i+1} Temp_{i,j} = 1 \Rightarrow$

$\Rightarrow S_i = (1 - P_i[0]) \,/\, \sum_{j=1}^{i+1} Temp_{i,j}$

Di conseguenza:

$\forall i, j \neq 0, W_{i,j} = S_i * Temp_{i,j}$

Si deve quindi provvedere ad una definizione del livello di WIP medio presente alla stazione di matching durante la produzione di un lotto di N unità.

Una soluzione possibile è:

$\overline{WIP} = \frac{1}{N} \sum_{i=0}^{N-1} \left[\sum_{j=0}^{i+1} (j * W_{i,j}) \right]$

Essendo però i turni di durata non omogenea, sarebbe più preciso legare la media al tempo.

Calcolerò il peso del WIP del turno i-esimo come la durata dello stesso divisa per il tempo totale di processo del lotto.

$TempoTot = \overline{z}_N + \sum_{i=1}^{N-1} \overline{t}_i$

Dove:

- $\overline{t}_i$ sono i tempi di interarrivo tra gli FTA di due coppie successive;
- $\overline{z}_N$ è il tempo di smaltimento della coda alla fine del lotto, ovvero il tempo che dovrà attendere l'ultimo componente entrato in coda per il suo complementare.

Considerando anche lo smaltimento della coda, l'ultima riga della tabella dei $W_{i,j}$ assumerà la seguente forma:

- $W_{N-1,0} = 0$ in quanto all'inizio dell'ultimo turno sicuramente almeno l'ultimo «primo componente» sarà in coda;
- $\forall j \geq 1$ avremo $W_{N-1,j} = W_{N-2,j-1}$.

Si otterrà:

$\overline{WIP} = \frac{1}{TempoTot} \left\{ \sum_{i=0}^{N-2} \left[\overline{t}_{i+1} \sum_{j=0}^{i+1} (W_{i,j} * j) \right] + \overline{z}_N \sum_{j=1}^{N} \left(W_{N-1,j} * \frac{j}{2} \right) \right\}.$

Rimangono da calcolare il Cicle Time e il Throughput.

Il TH è calcolabile come rapporto tra il numero di coppie componenti il lotto ed il tempo di produzione totale.

$\overline{TH} = N / TempoTot$

Di conseguenza, applicando Little:

$\overline{CT} = \frac{\overline{WIP}}{\overline{TH}}$

Volendo si potrebbe pervenire al $\overline{WIP}$ tramite un'altra via.

Essendo i tempi di processo della stazione di matching nulli, in quanto scorporati nella successiva stazione di assemblaggio, i vari $\overline{z}_i$ oltre a rappresentare i tempi di attesa medi delle varie coppie, coincidono con il Cycle Time medio di ogni coppia. Quindi, ottenendo il $\overline{TH}$come nel caso precedente, potrei calcolare $\overline{CT}$ e $\overline{WIP}$ come:

$\overline{CT} = \frac{1}{N}\left(\sum_{i=1}^{N} \overline{z_i}\right)$;

$\overline{WIP} = \overline{CT} * \overline{TH}$.

Sarebbe anche possibile calcolare il TH come l'inverso della media dei tempi medi di interuscita o come la media del tempo di interuscita «globale», utilizzandolo nella formula di Little assieme al valore di WIP medio ricavato tramite la precedente tabella, ottenendo il CT.

Buffer

Nei paragrafi precedenti sono stati modellizzati i tempi di interarrivo, interuscita e attesa nella loro forma locale o globale.

Come verrà esplicitato dai grafici presenti nella successiva sezione, si constata che per ogni coppia i tempi di interarrivo medi alla stazione di matching saranno sempre inferiori ai tempi di interuscita medi dalla stessa, sia nella modellizzazione locale che globale, generando un'utilizzazione superiore all'unità. Ciò comporta il tendere ad infinito dei livelli di WIP al crescere delle dimensioni del lotto. Una soluzione utile ad ovviare al problema può essere l'introduzione di un buffer all'ingresso della stazione di matching.

Per modellizzare questo sistema modificato, partendo dalla tabella dei WIP precedentemente elaborata, si potrà costruire uno strumento specifico.

Introducendo un buffer di valore «b», il livello di WIP massimo raggiungibile sarà pari a «b+1» coppie, rendendo necessaria la modifica della tabella delle probabilità dei livelli di WIP precedentemente elaborata.

Per $0 \leq i \leq b$ la tabella non cambierà, in quanto sarà impossibile lo sforamento del limite di «b+1» FTA in coda.

Per $i \geq b+1$ e $j = 0$ invece si avrà:

$$W_{b,i,0} = W_{b,i-1,0} \int_0^{+\infty} x \int_0^x k_1 + W_{b,i-1,1} \int_0^{+\infty} x \int_0^x k_2 + \sum_{j=2}^{b} \left[W_{b,i-1,j} \int_0^{+\infty} x \int_0^x k_{j+1} \right] =$$

$$= \sum_{j=0}^{b} \left[W_{b,i-1,j} \int_0^{\infty} x \int_0^x k_{j+1} \right]$$

Si noti che:

$$\int_0^{+\infty} x \int_0^x k_{j+1} = \int_0^{+\infty} \lambda e^{-\lambda x} \int_0^x \frac{k^j \lambda^{j+1} e^{-\lambda k}}{j!} dk dx = \left(\frac{1}{2}\right)^{j+1}$$

$$\Rightarrow W_{b,i,0} = \sum_{j=0}^{b} \left[W_{b,i-1,j} \left(\frac{1}{2}\right)^{j+1} \right].$$

Per $i \geq b+1$ e $1 \leq j \leq b$ si avrà:

$$W_{b,i,j} = \sum_{a=j-1}^{b} \left[W_{b,i-1,a} \left(\frac{1}{2}\right)^{a-j+2} \right].$$

Per $i \geq b+1$ e $j = b+1$ si avrà:

$W_{b,i,b+1} = \frac{1}{2} W_{b,i-1,b}$.

La probabilità di incorrere nel blocco delle accettazioni in coda a causa di raggiunta saturazione del buffer nel turno i-esimo verrà definita come:

$W_{b,i,STOP} = W_{b,i-1,b+1} + W_{b,i-1,STOP}$.

Riassumendo:

- $W_{b,i,j} = W_{i,j}$ per $0 \leq i \leq b$,;
- $W_{b,i,0} = \sum_{j=0}^{b} \left[W_{b,i-1,j} \left(\frac{1}{2}\right)^{j+1} \right]$ per $i \geq b+1, j = 0$;
- $W_{b,i,j} = \sum_{a=j-1}^{b} \left[W_{b,i-1,a} \left(\frac{1}{2}\right)^{a-j+2} \right]$ per $i \geq b+1, 1 \leq j \leq b$;
- $W_{b,i,b+1} = \frac{1}{2} W_{b,i-1,b}$ per $i \geq b+1, j = b+1$;
- $W_{b,i,STOP} = W_{b,i-1,b+1} + W_{b,i-1,STOP}$.

Turni\WIP	0	1	2	3	4	...	b	b+1	STOP
0	$W_{b,0,0}$	$W_{b,0,1}$							
1	$W_{b,1,0}$	$W_{b,1,1}$	$W_{b,1,2}$						
2	$W_{b,2,0}$	$W_{b,2,1}$	$W_{b,2,2}$	$W_{b,2,3}$					
3	$W_{b,3,0}$	$W_{b,3,1}$	$W_{b,3,2}$	$W_{b,3,3}$	$W_{b,3,4}$				
...									
b	$W_{b,b,0}$	$W_{b,b,1}$	$W_{b,b,2}$	$W_{b,b,3}$	$W_{b,b,4}$	...	$W_{b,b,b}$	$W_{b,b,b+1}$	$W_{b,b,STOP}$
b+1	$W_{b,b+1,0}$	$W_{b,b+1,1}$	$W_{b,b+1,2}$	$W_{b,b+1,3}$	$W_{b,b+1,4}$	...	$W_{b,b+1,b}$	$W_{b,b+1,b+1}$	$W_{b,b+1,STOP}$
b+2	$W_{b,b+2,0}$	$W_{b,b+2,1}$	$W_{b,b+2,2}$	$W_{b,b+2,3}$	$W_{b,b+2,4}$	...	$W_{b,b+2,b}$	$W_{b,b+2,b+1}$	$W_{b,b+2,STOP}$
...									

Ottenuta questa tabella sarà possibile calcolare un numero di turni medio alla saturazione del buffer.

$$\overline{NtoSTOP}_b = \left\lceil \sum_{i=b}^{N-2} \left[(i+1) W_{b,i,b+1} \right] / \sum_{i=b}^{N-2} W_{b,i,b+1} \right\rceil$$

Da notare che $\overline{NtoSTOP}_b$ rappresenta anche il numero di FTA giunti alla stazione di accoppiamento prima dello stop della linea.

E' possibile calcolare il TH medio come l'inverso dell'intertempo medio di uscita, ottenuto operando la media della medie degli intertempi medi di uscita della singola coppia su «i» coppie necessarie a saturare il buffer, pesando il tutto con la probabilità di saturazione del buffer all'ingresso della coppia i-esima.

Si avrà:

$$\overline{tu}_{b,N} = \left[\left(\sum_{i=0}^{b-1} \overline{tu}_i \right) + \sum_{i=b}^{N-2} \left(\frac{\overline{tu}_i}{(i+1)} W_{b,i,b+1} \right) + \frac{\overline{tu}_{N-1}}{N} \left(1 - W_{b,N-2,b+2}\right) \right] / \left[\left(\sum_{i=b}^{N-2} W_{b,i,b+1} \right) + W_{b,N-2,b+2} \right];$$

$\overline{TH}_{b,N} = 1/\overline{tu}_{b,N}$.

Dove:

- $\overline{tu}_{b,N}$ sarà il tempo medio di interuscita calcolato su un lotto di produzione di N coppie in una stazione avente un buffer di dimensione b;
- $\overline{TH}_{b,N}$ sarà il TH medio su un lotto di produzione di N coppie in una stazione avente un buffer di dimensione b.

In modo simile si potrà interpretare il Cycle Time (che in questo specifico caso si ricorda coincidere con i tempi di attesa).

$$\overline{CT}_{b,N} = \left[\left(\sum_{i=1}^{b} \overline{z}_i \right) + \sum_{i=b+1}^{N-2} \left(\frac{\overline{z}_i}{i} W_{b,i,b+1} \right) + \frac{\overline{z}_N}{N} \left(1 - W_{b,N-2,b+2}\right) \right] / \left[\left(\sum_{i=b+1}^{N-2} W_{b,i,b+1} \right) + W_{b,N-2,b+2} \right]$$

Da cui, tramite legge di Little, si otterrà:

$\overline{WIP}_{b,N} = \overline{CT}_{b,N} * \overline{TH}_{b,N}$

Simulatore

Ottenuto il modello sorge la necessità di testarne i risultati con un simulatore.

E' stata presa la decisione di crearne uno specifico da zero, onde avere il totale controllo dei processi e variabili coinvolte.

Il simulatore si basa su un'estrazione casuale di un valore «y» in un ampio range prestabilito [0,Y].

Si ricorda che la funzione di probabilità cumulata della distribuzione esponenziale è:

$\mathbb{P}[X < x] = 1 - e^{-\lambda x}$

Quindi:

$\frac{y}{Y} = 1 - e^{-\lambda x} \Rightarrow e^{-\lambda x} = 1 - \frac{y}{Y} \Rightarrow x = -\frac{1}{\lambda} ln\left(1 - \frac{y}{Y}\right)$

Avendo il sistema in analisi due linee a monte della stazione di matching, entrambe con i tempi di interuscita caratterizzati dalla medesima distribuzione esponenziale, si definiscono A e B come vettori dei tempi di interuscita dei componenti prodotti dalle stazioni a monte. Entrambi hanno valori estratti casualmente.

Data una dimensione N del lotto di produzione, si estrarranno N valori casuali per entrambi i vettori e tramite la precedente formula li si trasformerà nei tempi di interuscita del singolo componente.

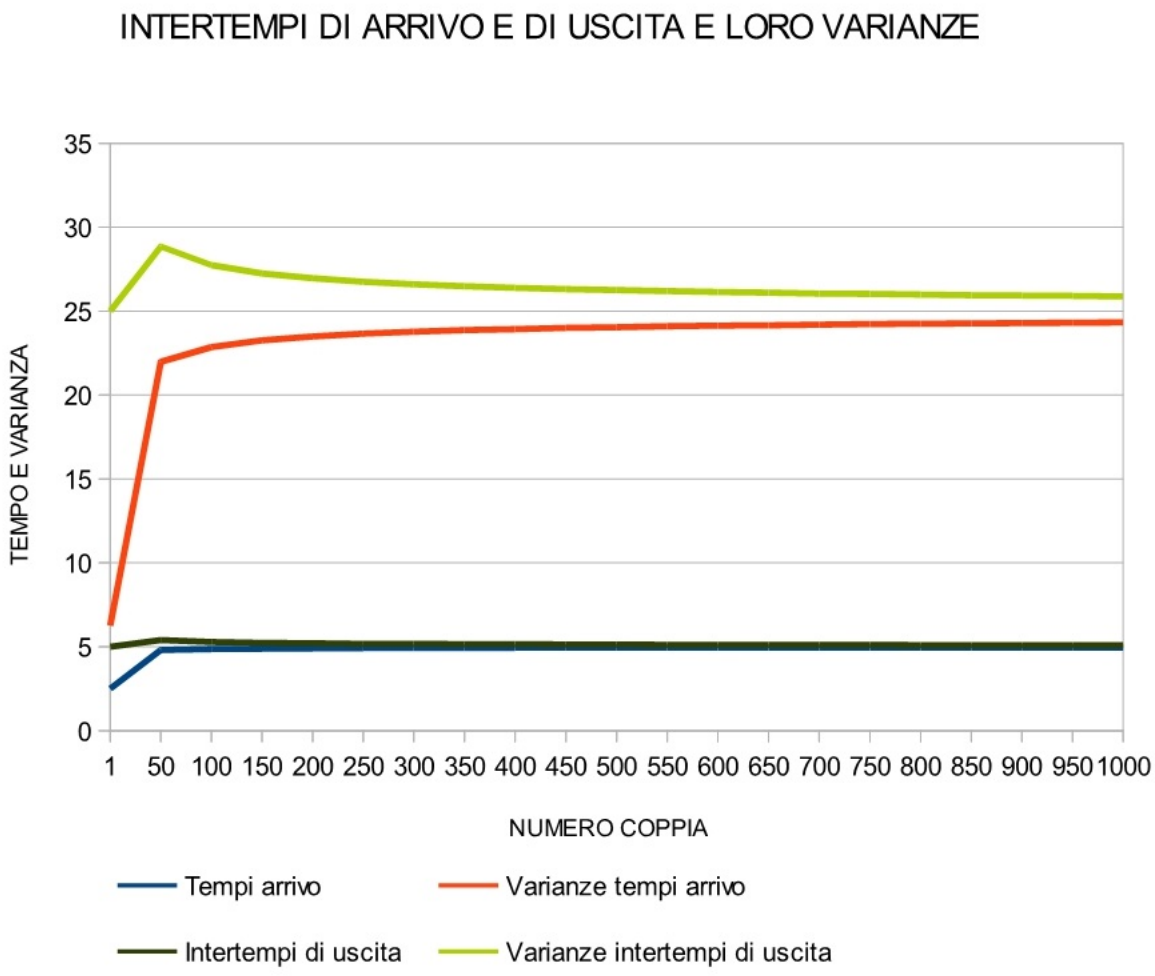
Ottenuto questo si procederà alla creazione delle tabelle delle timeline di ingresso e uscita dalla stazione di matching, sulle quali si baseranno il calcolo dei parametri della stazione (intertempi di ingresso ed uscita delle coppie dalla stazione, tempi di attesa, loro varianze e livelli di WIP).

Per approfondimenti sui dettagli dell'algoritmo utilizzato, fare riferimento al materiale in appendice.

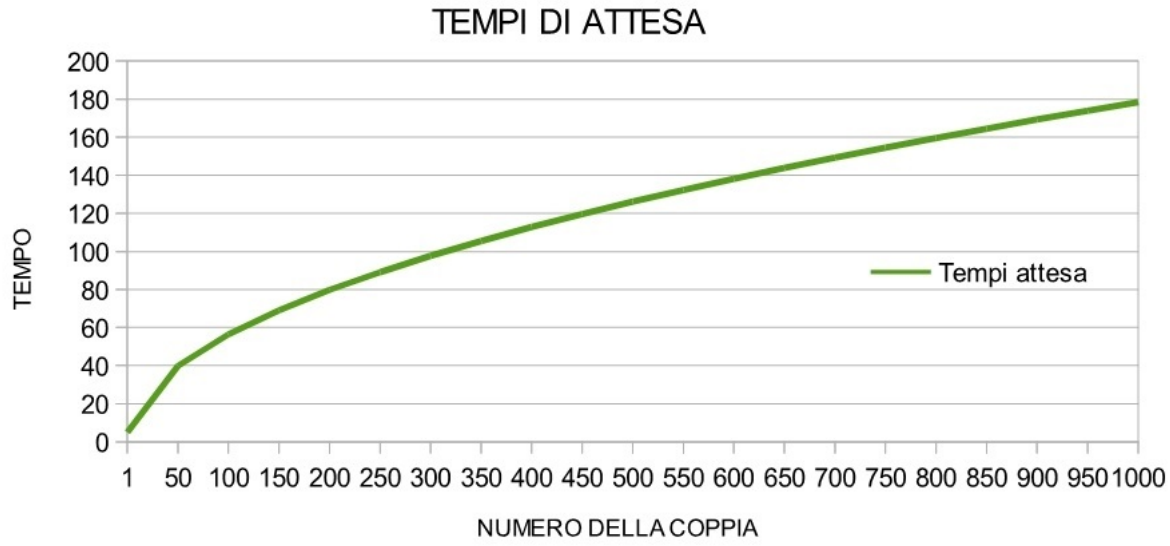
Risultati del modello, confronto con i dati prodotti dal simulatore, suggerimento sull'utilizzo degli strumenti prodotti

Qui di seguito si riportano i grafici dei risultati ottenuti modellizzando un lotto da 1000 coppie i cui componenti sono prodotti da due linee a monte entrambe caratterizzate da una distribuzione esponenziale degli intertempi di uscita aventi media pari a $1/\lambda = 5$ minuti.

Nel primo vengono inseriti gli intertempi medi e le varianze delle singole coppie. Possiamo osservare la convergenza dei valori medi dei tempi di interarrivo al limite inferiore di 5, mentre gli intertempi di uscita tenderanno al limite superiore di 5. Ugualmente si nota la convergenza delle rispettive varianze rispettivamente al limite inferiore e superiore di 25 ($= 1/\lambda^2$).

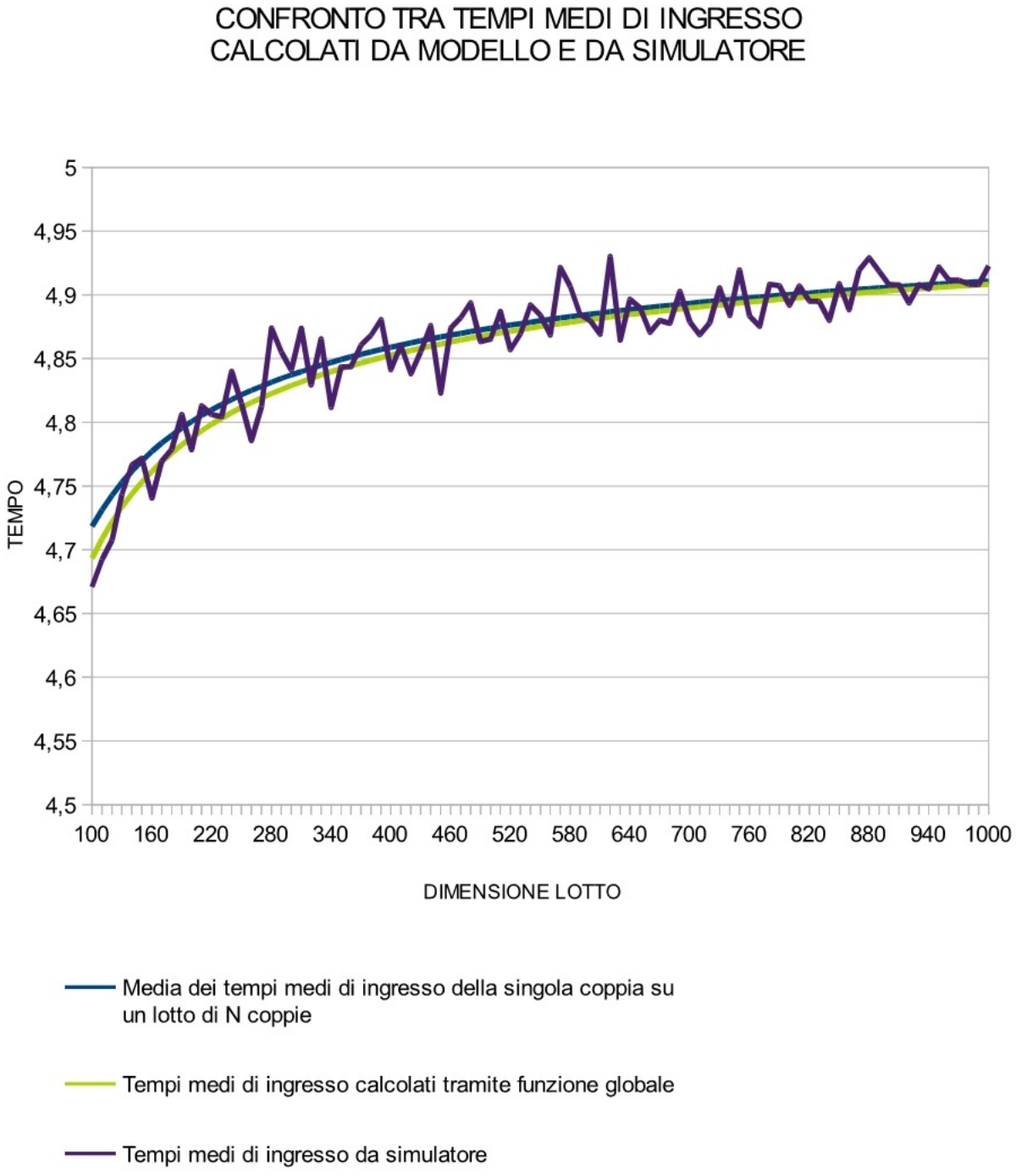


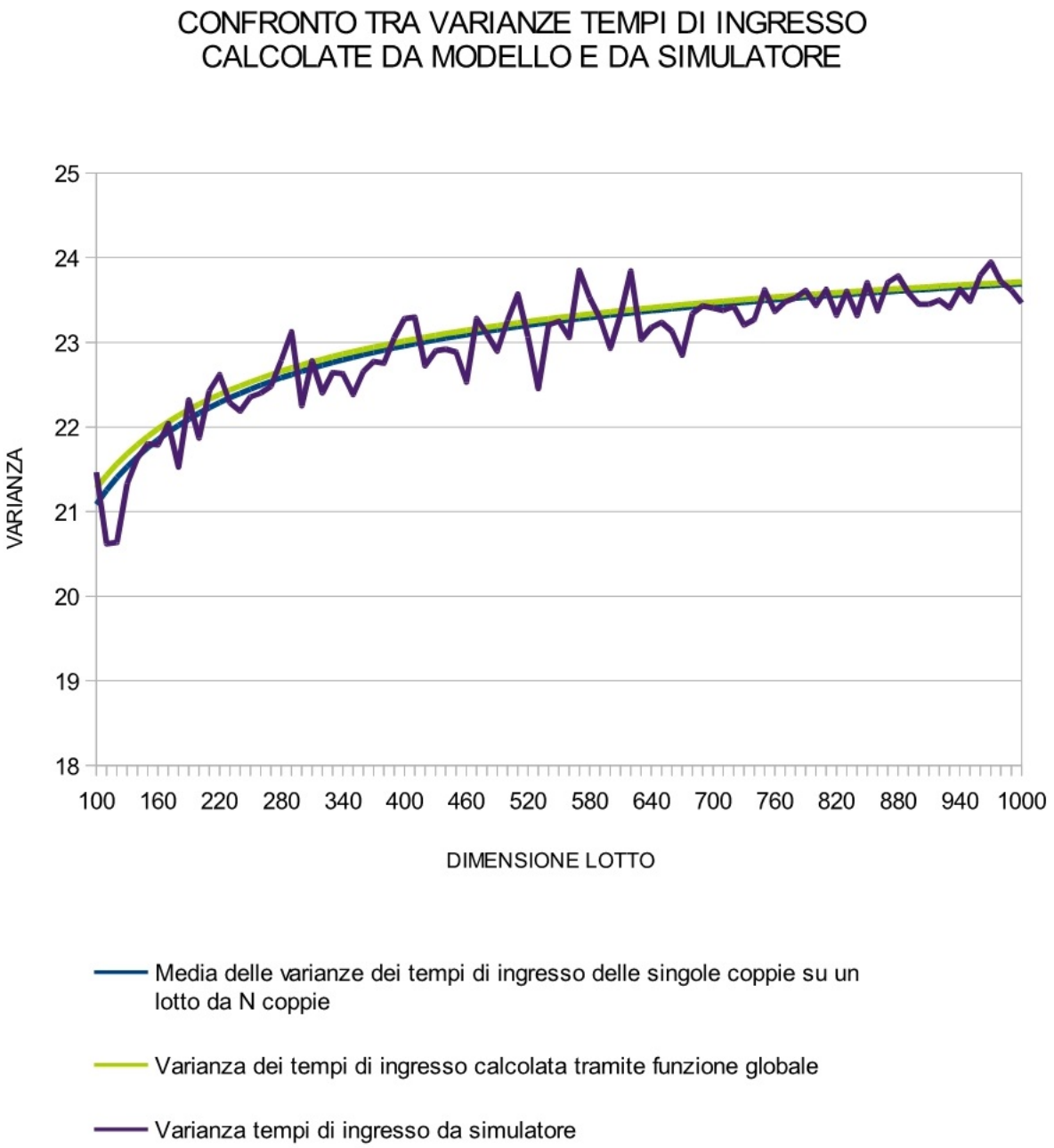
Essendo gli intertempi di uscita sempre superiori agli intertempi di ingresso alla stazione, è prevedibile che l'andamento del tempo medio che ogni singolo FTA dovrà attendere per il suo complementare sarà crescente, come illustrato nell'immagine seguente.

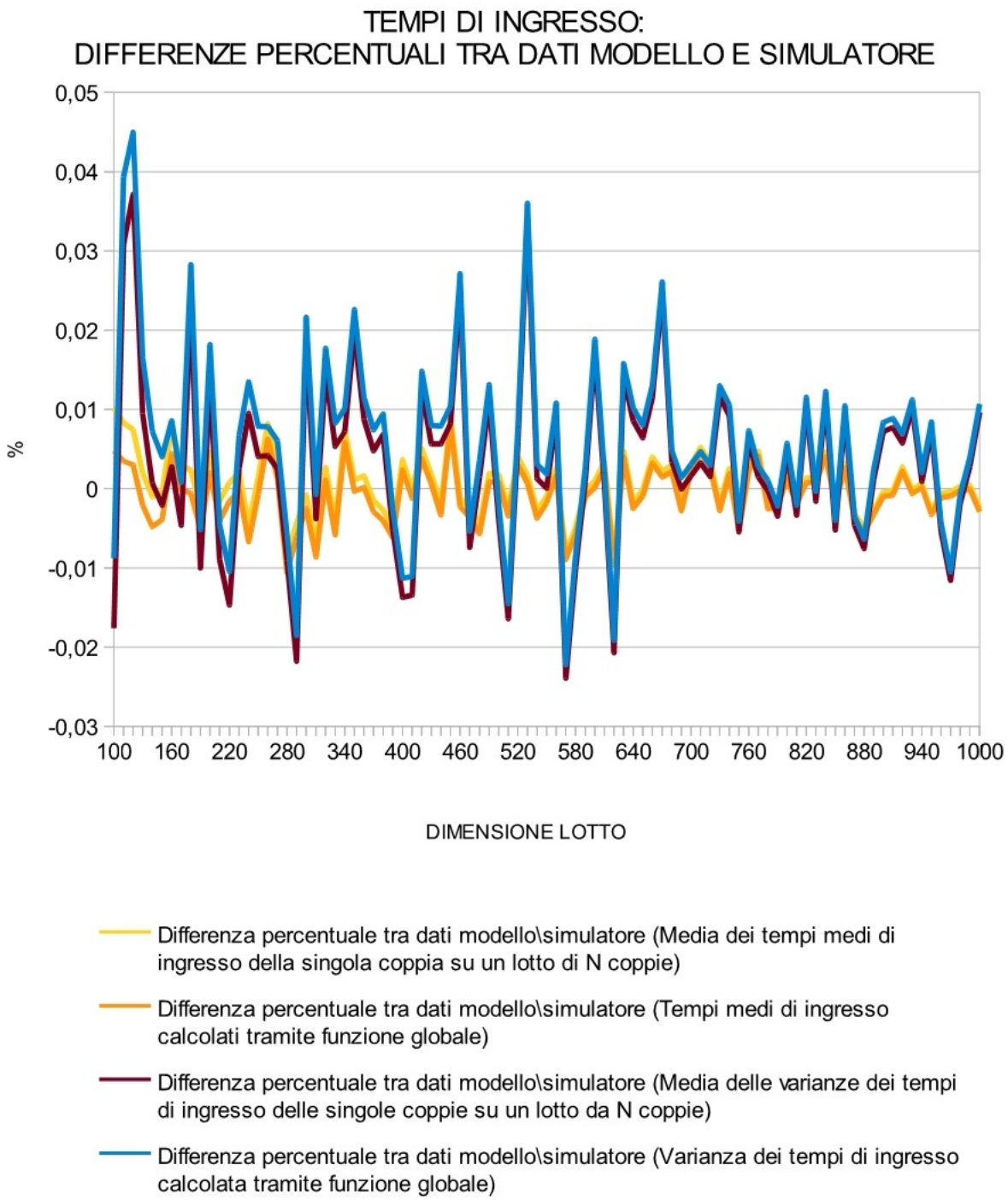


Di seguito si riportano le immagini rappresentanti il confronto tra i dati ottenuti dal modello e dal simulatore su un campione di lotti aventi parametri delle macchine identici al caso precedente ma

dimensione del lotto analizzato crescente fino a 1000, in assenza di buffer.




CONFRONTO TRA VARIANZE TEMPI DI INGRESSO CALCOLATE DA MODELLO E DA SIMULATORE
VARIANZA
25
24
23
22
21
20
19
18
100 160 220 280 340 400 460 520 580 640 700 760 820 880 940 1000
DIMENSIONE LOTTO
Media delle varianze dei tempi di ingresso delle singole coppie su un lotto da N coppie
Varianza dei tempi di ingresso calcolata tramite funzione globale
Varianza tempi di ingresso da simulatore



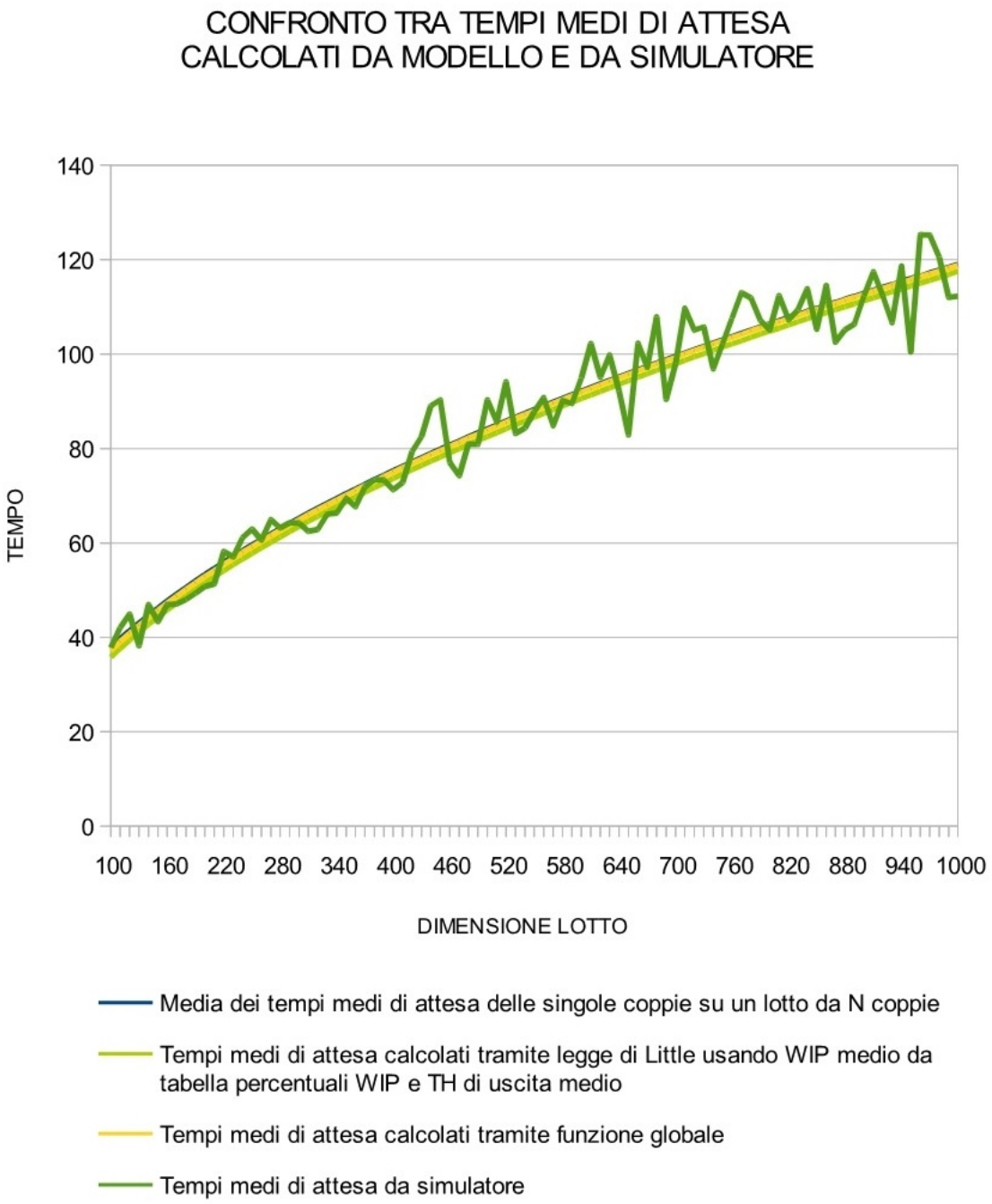
Dai grafici precedenti si evince che:

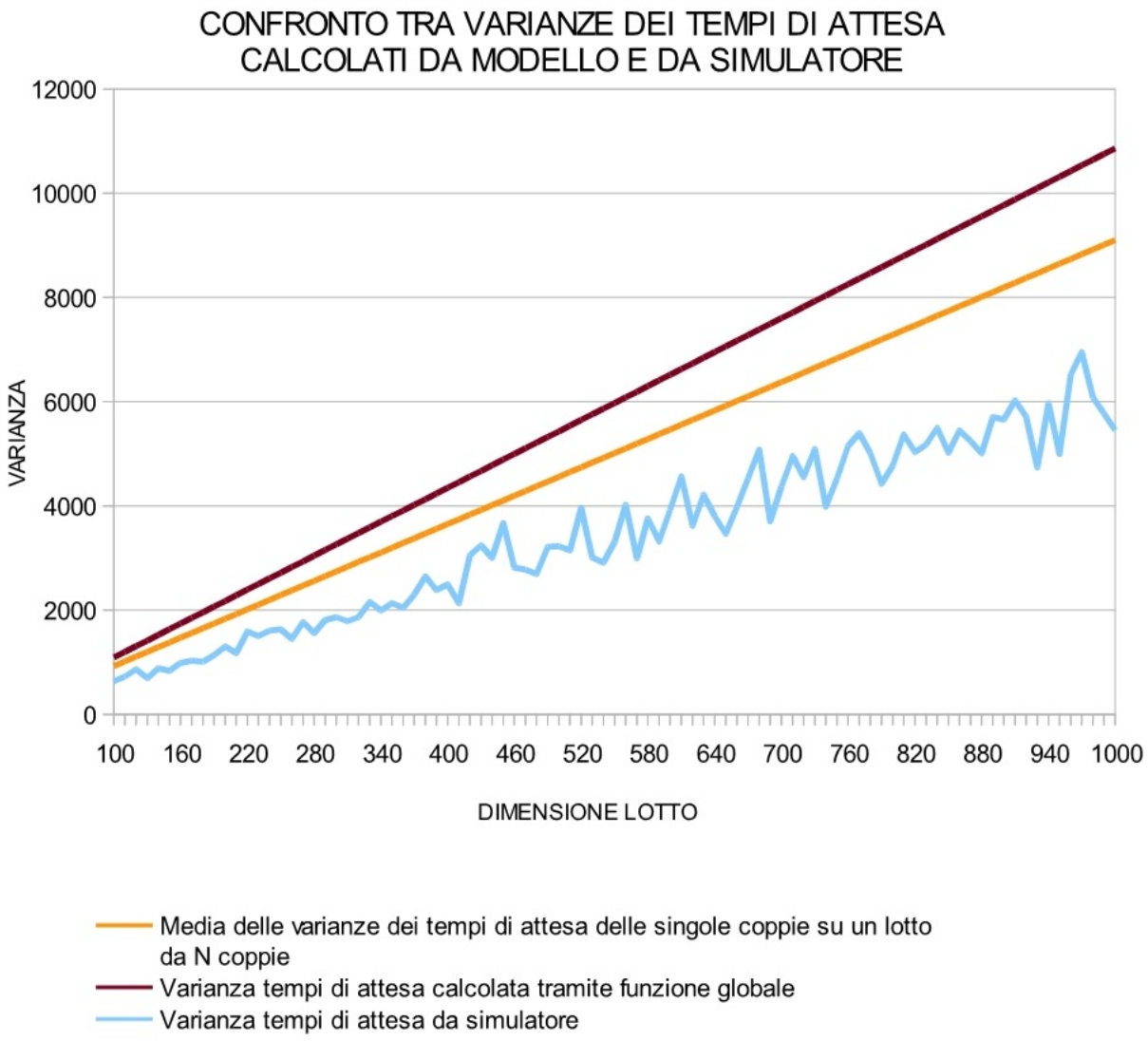
- riguardo il valore medio degli intertempi in ingresso delle coppie alla stazione prodotti dal modello calcolato tramite media dei valori medi delle singole coppie, indipendentemente dalla dimensione del lotto modellizzato, presenterà sempre, rispetto ai dati prodotti dal simulatore, un errore percentuale compreso tra -1% e +1% con una media pari al 0.0511% ed una varianza del $1,47{*}10^{-5}$;

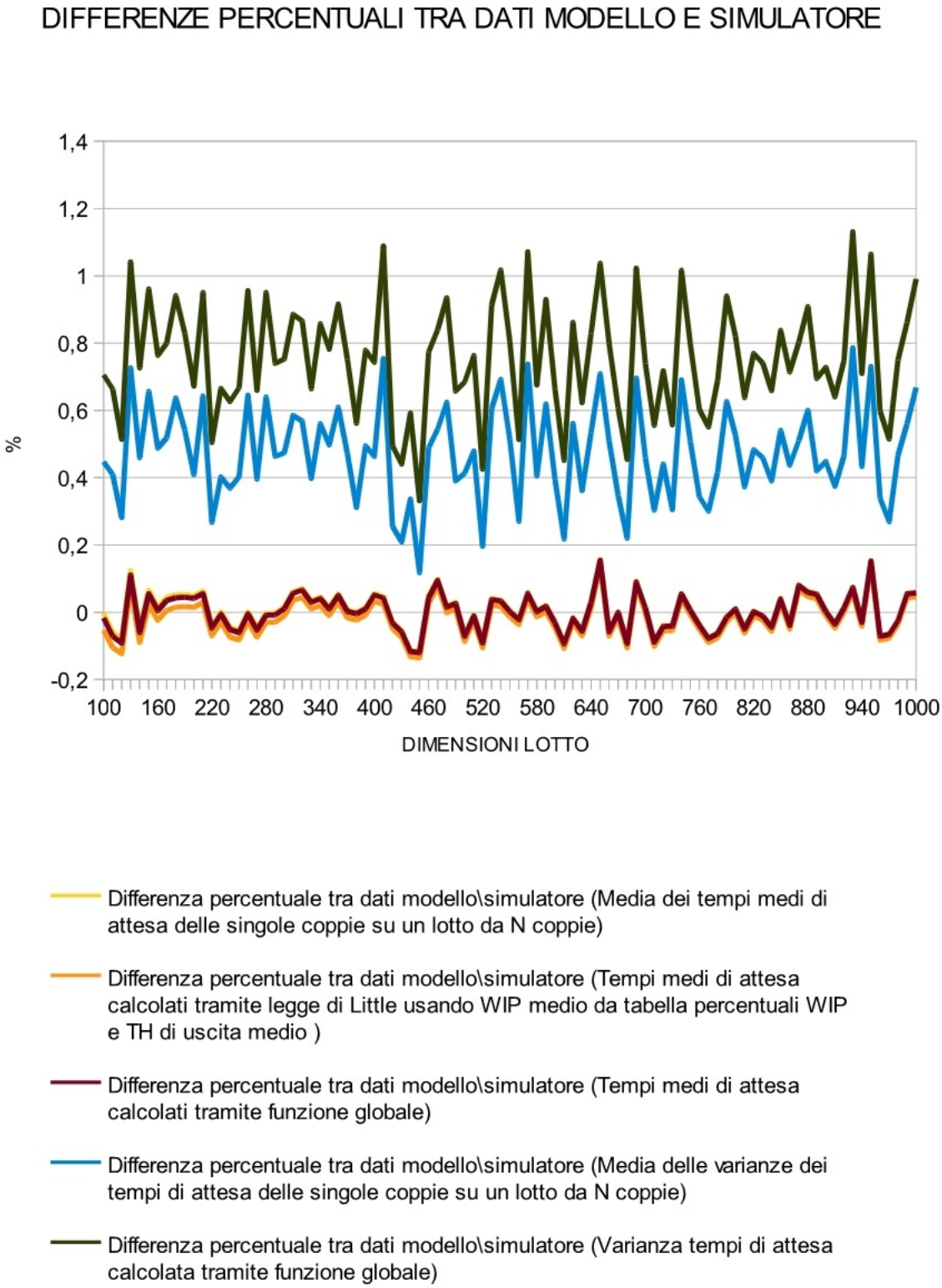
- riguardo il valore medio degli intertempi in ingresso delle coppie alla stazione prodotti dal modello calcolato tramite funzione globale, indipendentemente dalla dimensione del lotto modellizzato, presenterà sempre, rispetto ai dati prodotti dal simulatore, un errore percentuale compreso tra -1% e +1% con una media pari al 0.083% ed una varianza del $1,32 * 10^{-5}$;

- riguardo il valore della varianza degli intertempi in ingresso delle coppie alla stazione prodotti dal modello calcolato tramite media delle varianze delle singole coppie, indipendentemente dalla dimensione del lotto modellizzato, presenterà sempre, rispetto ai dati prodotti dal simulatore, un errore percentuale compreso tra -2,5% e +5% con una media pari al 0.327% ed una varianza del $1,27 * 10^{-4}$;

- riguardo il valore della varianza degli intertempi in ingresso delle coppie alla stazione prodotti dal modello calcolato tramite funzione globale, indipendentemente dalla dimensione del lotto modellizzato, presenterà sempre, rispetto ai dati prodotti dal simulatore, un errore percentuale compreso tra -2,5% e +5% con una media pari al 0.578% ed una varianza del $1,35 * 10^{-5}$.




CONFRONTO TRA VARIANZE DEI TEMPI DI ATTESA CALCOLATI DA MODELLO E DA SIMULATORE
12000
10000
8000
6000
4000
2000
0
VARIANZA
100 160 220 280 340 400 460 520 580 640 700 760 820 880 940 1000
DIMENSIONE LOTTO
Media delle varianze dei tempi di attesa delle singole coppie su un lotto da N coppie
Varianza tempi di attesa calcolata tramite funzione globale
Varianza tempi di attesa da simulatore



Dai grafici precedenti si evince che:

- riguardo il valore medio dei tempi di attesa delle coppie prodotti dal modello calcolato tramite media dei valori medi delle singole coppie, indipendentemente dalla dimensione del lotto modellizzato, presenterà sempre un errore percentuale, rispetto ai dati prodotti dal simulatore, compreso tra -20% e +20% con una media pari al 0.35% ed una varianza di $3,19 * 10^{-3}$;

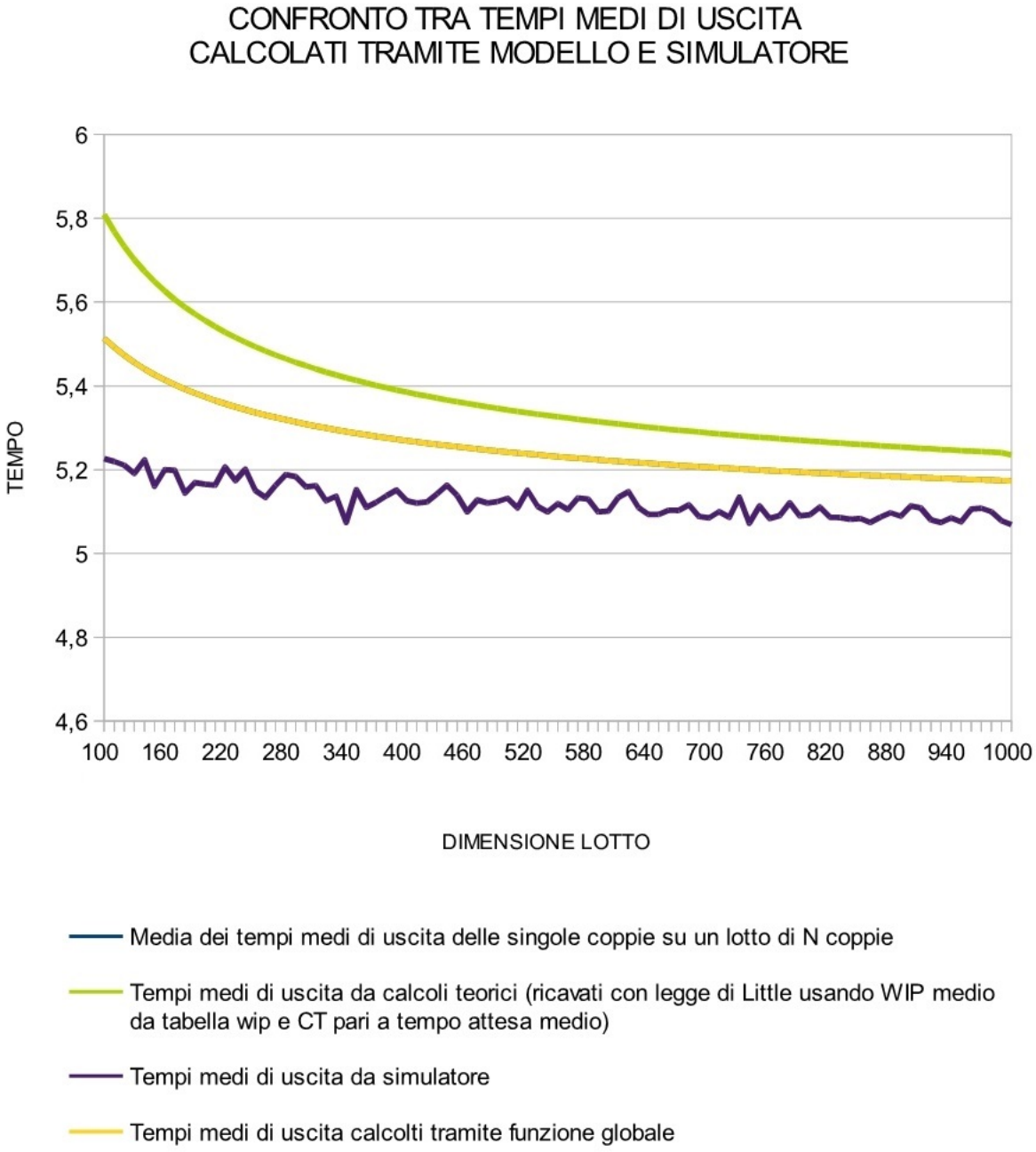
- riguardo il valore medio dei tempi di attesa delle coppie prodotti dal modello calcolato tramite legge di Little, indipendentemente dalla dimensione del lotto modellizzato, presenterà sempre un errore percentuale, rispetto ai dati prodotti dal simulatore, compreso tra -20% e +20% con una media pari al -1,76% ed una varianza di $3,09 * 10^{-3}$;

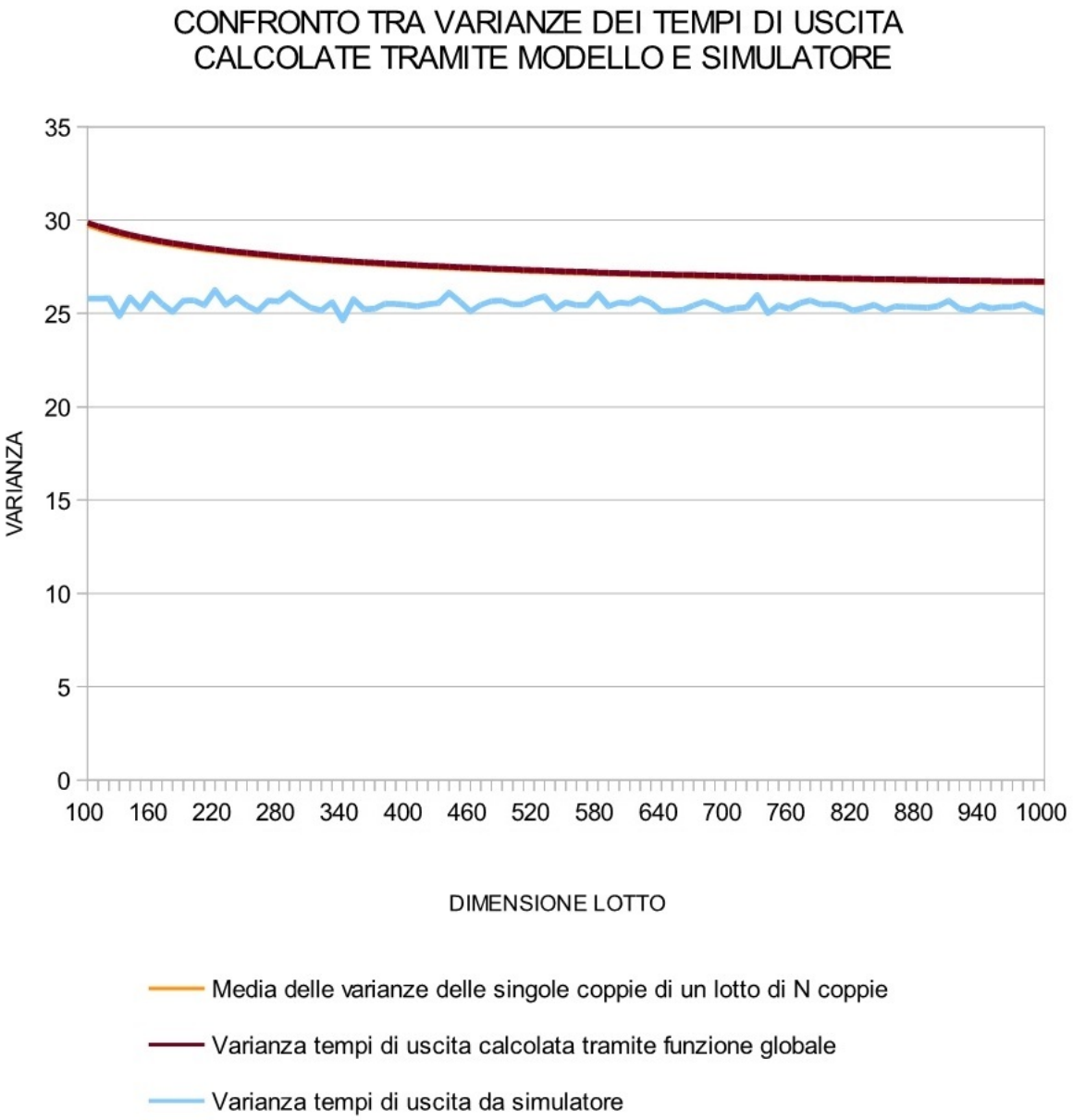
- riguardo il valore medio dei tempi di attesa delle coppie prodotti dal modello calcolato tramite funzione globale, indipendentemente dalla dimensione del lotto modellizzato, presenterà sempre un errore percentuale, rispetto ai dati prodotti dal simulatore, compreso tra -20% e +20% con una media pari al -0.028% ed una varianza di $3,16 * 10^{-3}$;

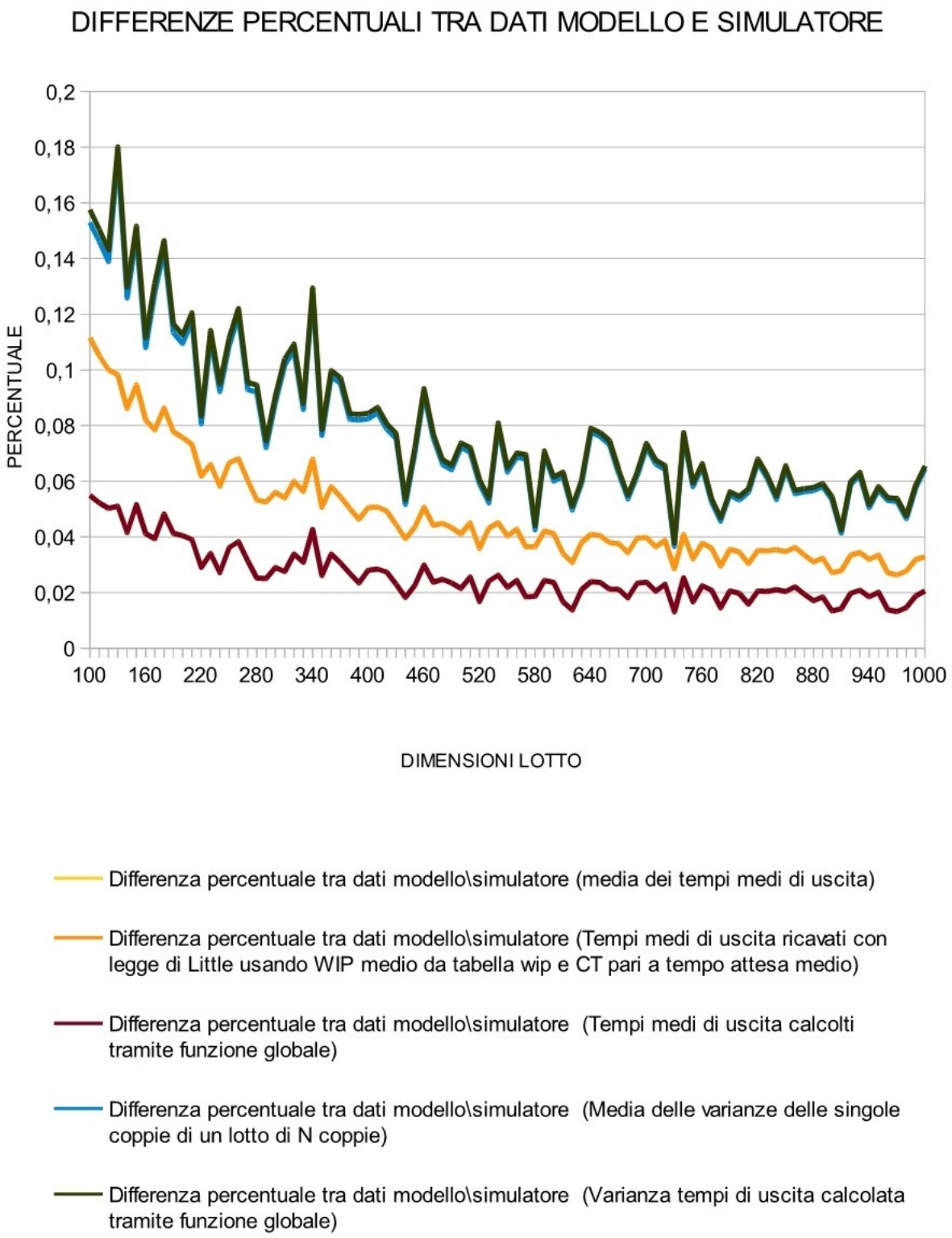
- riguardo il valore della varianza dei tempi di attesa delle coppie prodotti dal modello calcolato tramite media delle varianze delle singole coppie, indipendentemente dalla dimensione del lotto modellizzato, presenterà sempre un errore percentuale, rispetto ai dati prodotti dal simulatore, compreso tra +10% e +80% con una media pari al 40% ed una varianza di $2,045 * 10^{-2}$;

- riguardo il valore della varianza dei tempi di attesa delle coppie prodotti dal modello calcolato tramite funzione globale, indipendentemente dalla dimensione del lotto modellizzato, presenterà sempre un errore percentuale, rispetto ai dati prodotti dal simulatore, compreso tra +30% e +120% con una media pari al 75,49% ed una varianza di $2,89 * 10^{-2}$.

Rispetto alle previsioni del tempo medio di attesa, il modello una certa attendibilità. Purtroppo così non è con la varianza, dove la modellizzazione tramite media delle singole varianze ottiene risultati migliori rispetto a quella ottenuta tramite funzione globale. E' comunque utilizzabile in ambito di misurazione di performance come limite di paragone in un caso di Practical Worst Case.


CONFRONTO TRA TEMPI MEDI DI USCITA
CALCOLATI TRAMITE MODELLO E SIMULATORE
TEMPO
6
5,8
5,6
5,4
5,2
5
4,8
4,6
100 160 220 280 340 400 460 520 580 640 700 760 820 880 940 1000
DIMENSIONE LOTTO
Media dei tempi medi di uscita delle singole coppie su un lotto di N coppie
Tempi medi di uscita da calcoli teorici (ricavati con legge di Little usando WIP medio da tabella wip e CT pari a tempo attesa medio)
Tempi medi di uscita da simulatore
Tempi medi di uscita calcolti tramite funzione globale


CONFRONTO TRA VARIANZE DEI TEMPI DI USCITA
CALCOLATE TRAMITE MODELLO E SIMULATORE
35
30
25
20
15
10
5
0
VARIANZA
100 160 220 280 340 400 460 520 580 640 700 760 820 880 940 1000
DIMENSIONE LOTTO
Media delle varianze delle singole coppie di un lotto di N coppie
Varianza tempi di uscita calcolata tramite funzione globale
Varianza tempi di uscita da simulatore



Dai grafici precedenti si evince che:

- riguardo il valore medio degli intertempi di uscita delle coppie dalla stazione prodotti dal modello calcolato tramite media dei valori medi delle singole coppie, presenterà un errore percentuale, rispetto ai dati prodotti dal simulatore, avente un andamento decrescente al crescere delle dimensioni del lotto analizzato, comunque compreso tra +1,3% e +7% con una media pari al 2,6% ed una varianza di $9,53 * 10^{-5}$;
- riguardo il valore medio degli intertempi di uscita delle coppie dalla stazione prodotti dal modello calcolato tramite legge di Little, presenterà un errore percentuale, rispetto ai dati prodotti dal

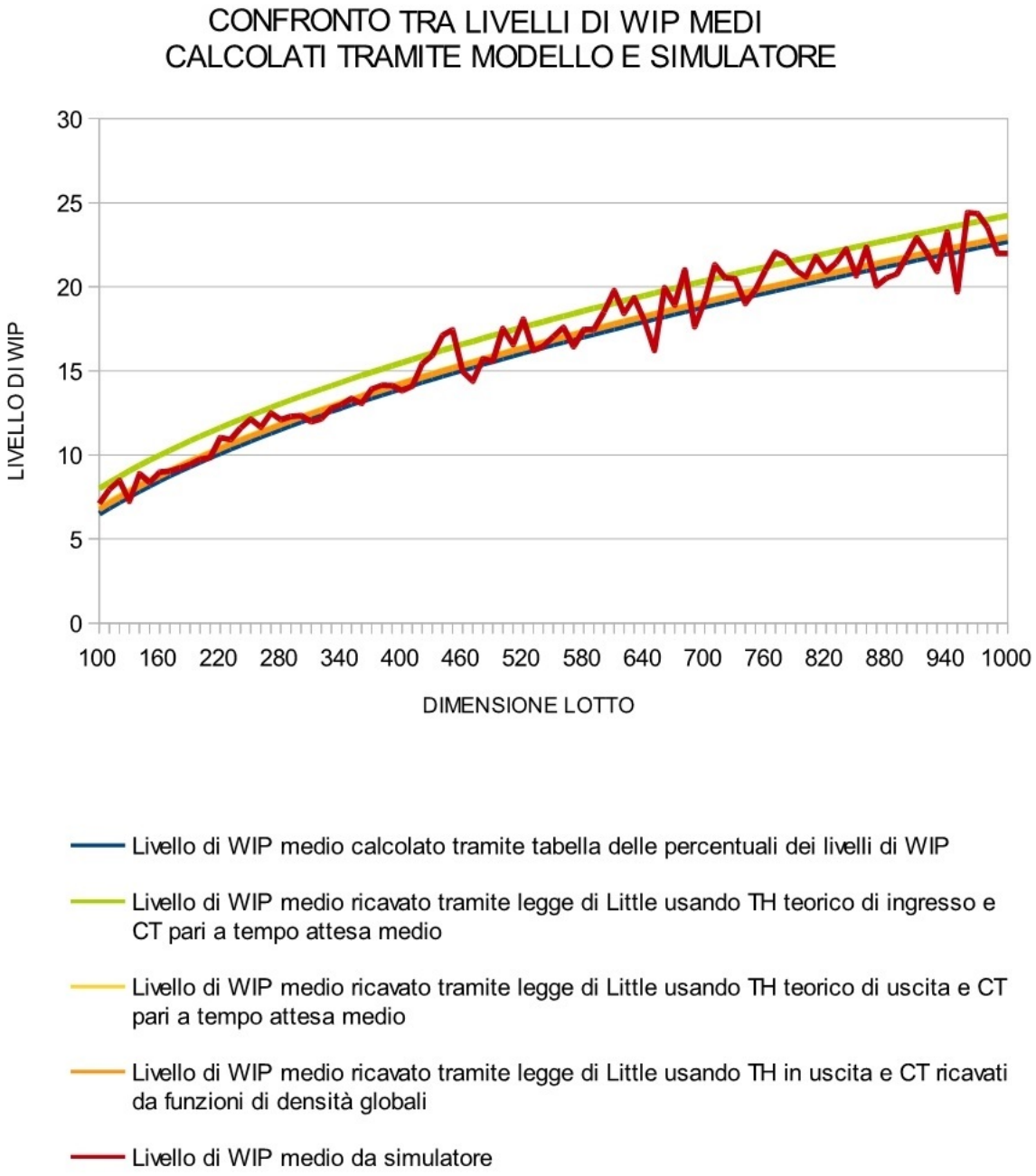
simulatore, avente un andamento decrescente al crescere delle dimensioni del lotto analizzato, comunque compreso tra +2,7% e +12% con una media pari al +4,8% ed una varianza di $3,7*10^{-4}$;

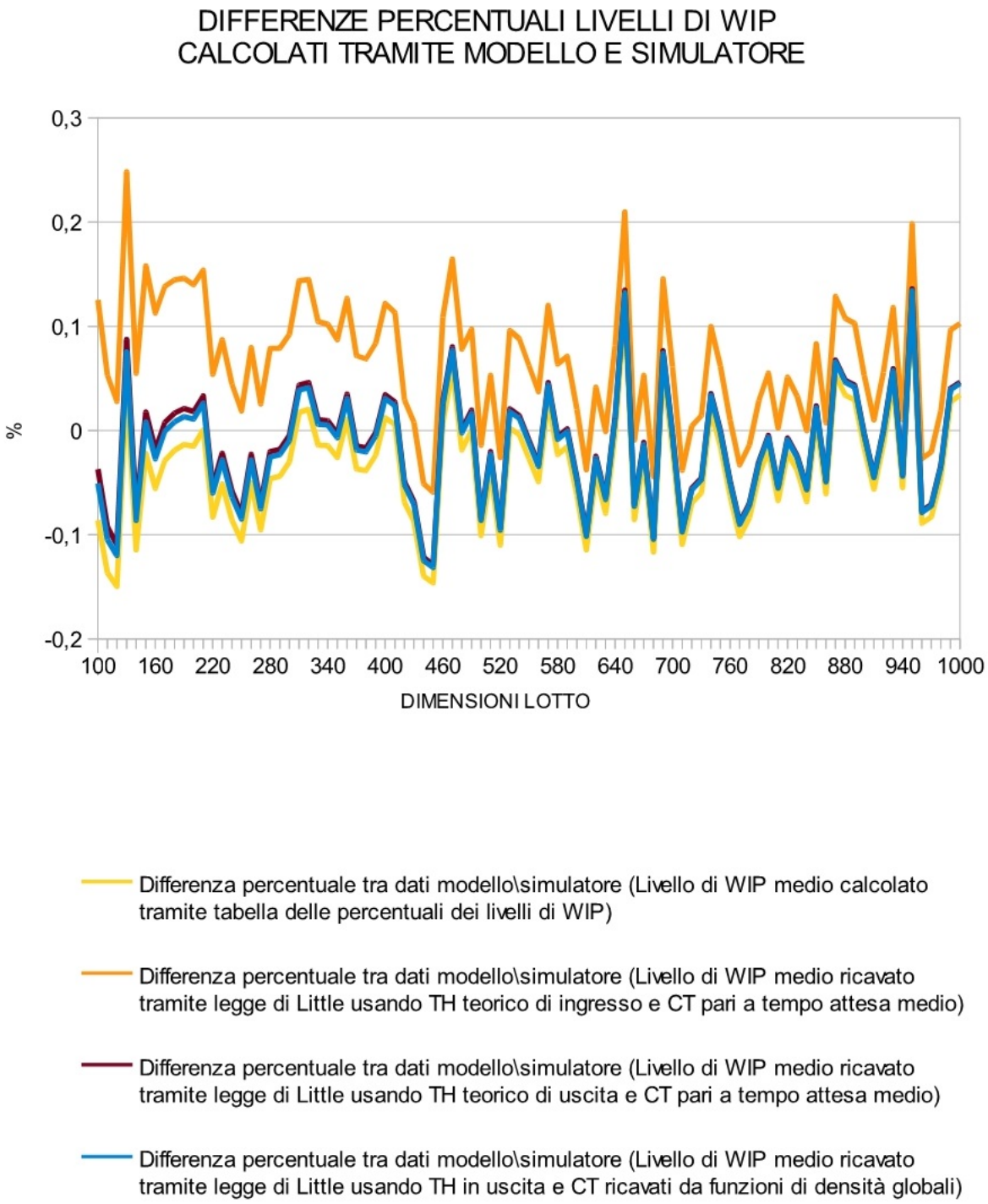
- riguardo il valore medio degli intertempi di uscita delle coppie dalla stazione prodotti dal modello calcolato tramite funzione globale, presenterà un errore percentuale, rispetto ai dati prodotti dal simulatore, avente un andamento decrescente al crescere delle dimensioni del lotto analizzato, comunque compreso tra +1,3% e +6% con una media pari al +2,6% ed una varianza di $39,53*10^{-5}$;

- riguardo il valore della varianza degli intertempi di uscita delle coppie dalla stazione prodotti dal modello calcolato tramite media delle varianze delle singole coppie, presenterà un errore percentuale, rispetto ai dati prodotti dal simulatore, avente un andamento decrescente al crescere delle dimensioni del lotto analizzato, comunque compreso tra +3,5% e +16% con una media pari al +7,8% ed una varianza di $8,13 * 10^{-4}$;

- riguardo il valore della varianza degli intertempi di uscita delle coppie dalla stazione prodotti dal modello calcolato tramite funzione globale, presenterà un errore percentuale, rispetto ai dati prodotti dal simulatore, avente un andamento decrescente al crescere delle dimensioni del lotto analizzato, comunque compreso tra +3,5% e +16% con una media pari al +8% ed una varianza di $8,59 * 10^{-4}$.

Negli intertempi di uscita il modello torna ad essere performante, con la modellizzazione tramite media delle medie e delle varianze leggermente più performante rispetto alle altrei. Essendo la differenza tra i valori modellizzati e quelli simulati sempre positiva, è consentito utilizzare i dati di previsione come limite superiore della performace attesa dal sistema.


CONFRONTO TRA LIVELLI DI WIP MEDI
CALCOLATI TRAMITE MODELLO E SIMULATORE
LIVELLO DI WIP
0
5
10
15
20
25
30
100
160
220
280
340
400
460
520
580
640
700
760
820
880
940
1000
DIMENSIONE LOTTO
Livello di WIP medio calcolato tramite tabella delle percentuali dei livelli di WIP
Livello di WIP medio ricavato tramite legge di Little usando TH teorico di ingresso e CT pari a tempo attesa medio
Livello di WIP medio ricavato tramite legge di Little usando TH teorico di uscita e CT pari a tempo attesa medio
Livello di WIP medio ricavato tramite legge di Little usando TH in uscita e CT ricavati da funzioni di densità globali
Livello di WIP medio da simulatore



Dai grafici precedenti si evince che:

- riguardo il valore del livello medio di WIP durante la fase di matching di un lotto di N coppie, calcolato tramite la tabella delle probabilità dei livelli di WIP all'uscita della singola coppia, rispetto ai dati prodotti dal simulatore presenterà un errore percentuale, indipendentemente dalle dimensioni del lotto, compreso tra -15% e +12% con una media pari a -3,5% ed una varianza di $2,89 * 10^{-3}$;

- riguardo il valore del livello medio di WIP durante la fase di matching di un lotto di N coppie, calcolato tramite legge di Little (con TH pari all'inverso dell'intertempo medio di ingresso modellizzato tramite media dei valori medi, con CT pari al tempo di attesa medio ricavato come media dei valori medi), rispetto ai dati prodotti dal simulatore presenterà un errore percentuale,

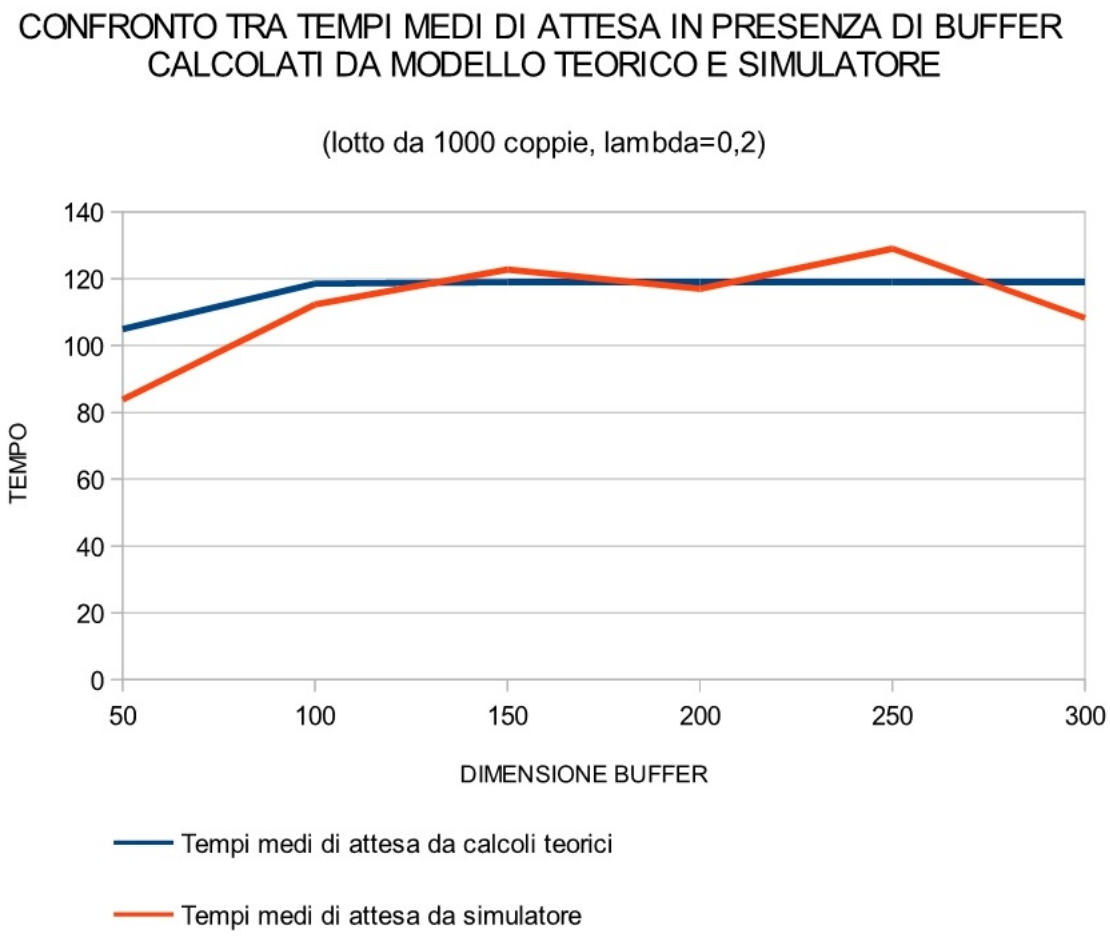
indipendentemente dalle dimensioni del lotto, compreso tra -6% e +26% con una media pari a +6,59% ed una varianza di $3,83 * 10^{-3}$;

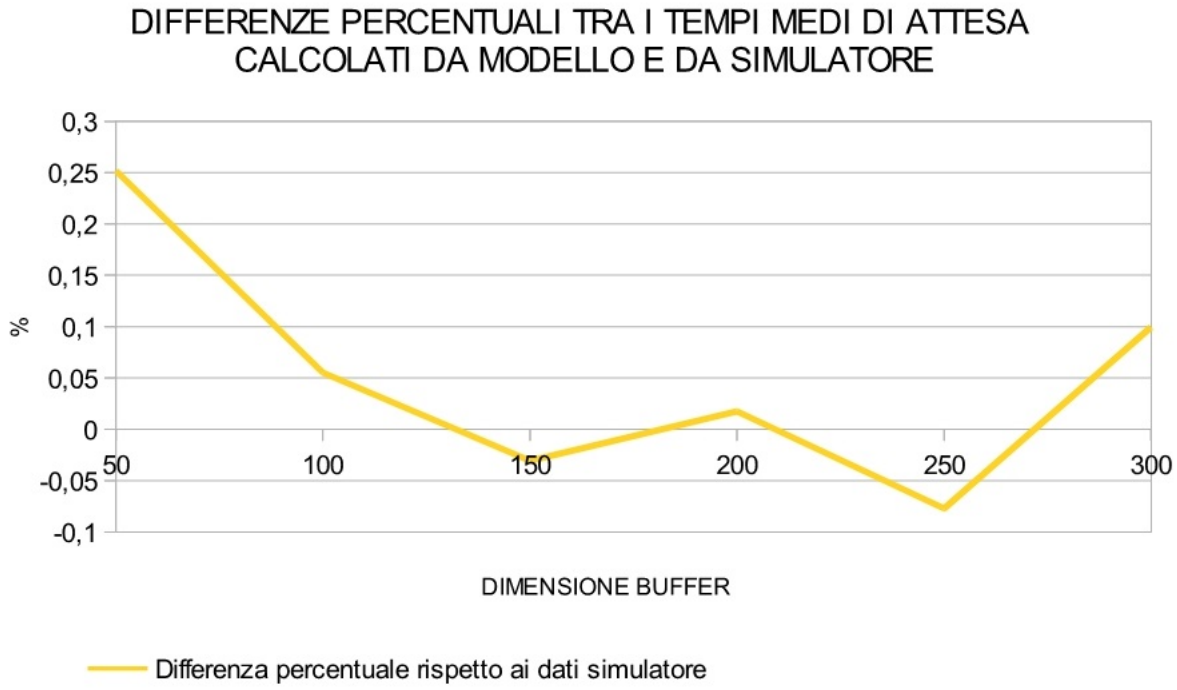
- riguardo il valore del livello medio di WIP durante la fase di matching di un lotto di N coppie, calcolato tramite legge di Little (con TH pari all'inverso dell'intertempo medio di uscita modellizzato tramite media dei valori medi, con CT pari al tempo di attesa medio ricavato come media dei valori medi), rispetto ai dati prodotti dal simulatore presenterà un errore percentuale, indipendentemente dalle dimensioni del lotto, compreso tra -11% e +15% con una media pari a -1,43% ed una varianza di $2,86 * 10^{-3}$;

- riguardo il valore del livello medio di WIP durante la fase di matching di un lotto di N coppie, calcolato tramite legge di Little (con TH pari all'inverso dell'intertempo medio di uscita modellizzato tramite media dei valori medi, con CT pari al tempo di attesa medio ricavato tramite funzione globale), rispetto ai dati prodotti dal simulatore presenterà un errore percentuale, indipendentemente dalle dimensioni del lotto, compreso tra -12% e +15% con una media pari a -1,8% ed una varianza di $2,87 * 10^{-3}$.

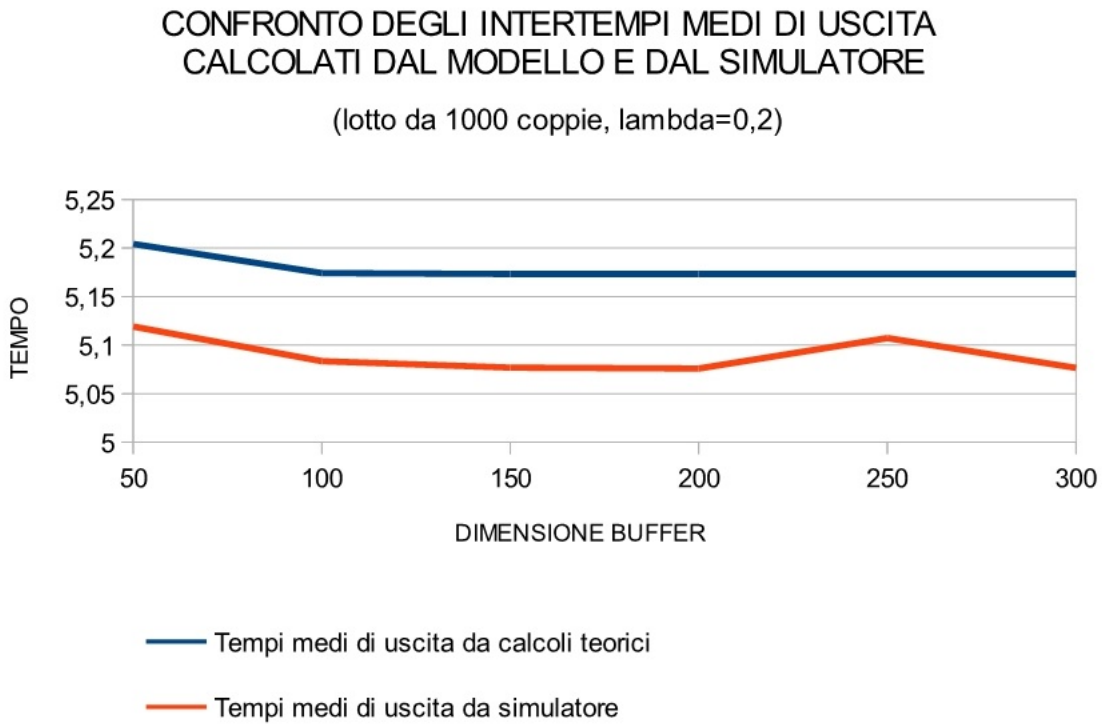
Ai fini della definizione di un limite superiore rispetto al quale parametrare la performance di un sistema, dai dati precedentemente elencati, risulta utile la modellizzazione utilizzando all'interno della legge di Little un TH pari al TH d'ingresso. Per tutti gli altri casi, può essere utilizzato uno qualsiasi dei rimanenti tre metodi di modellizzazione.

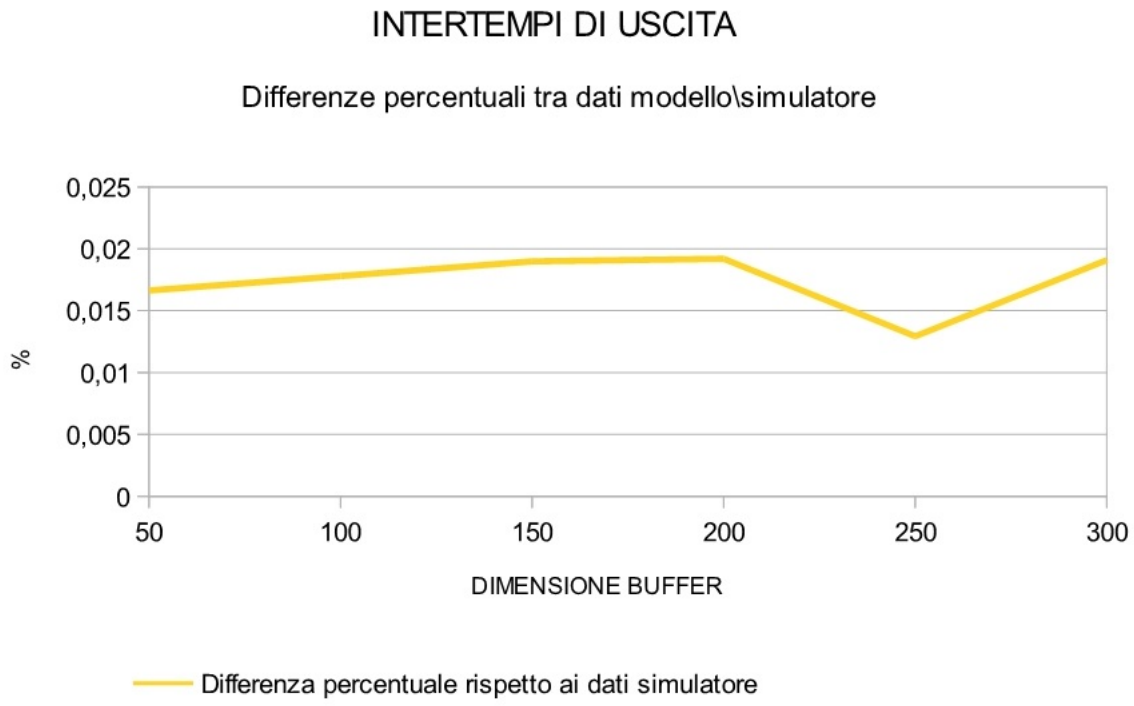
Come sistema di base, per valutare la prestazione del modello in presenza di un buffer, è stato utilizzato il medesimo usato in precedenza, con un lotto di dimensione costante pari a mille coppie. Il campione di casi su cui verrà effettuato il confronto è stato creato introducendo un buffer di dimensioni crescenti.



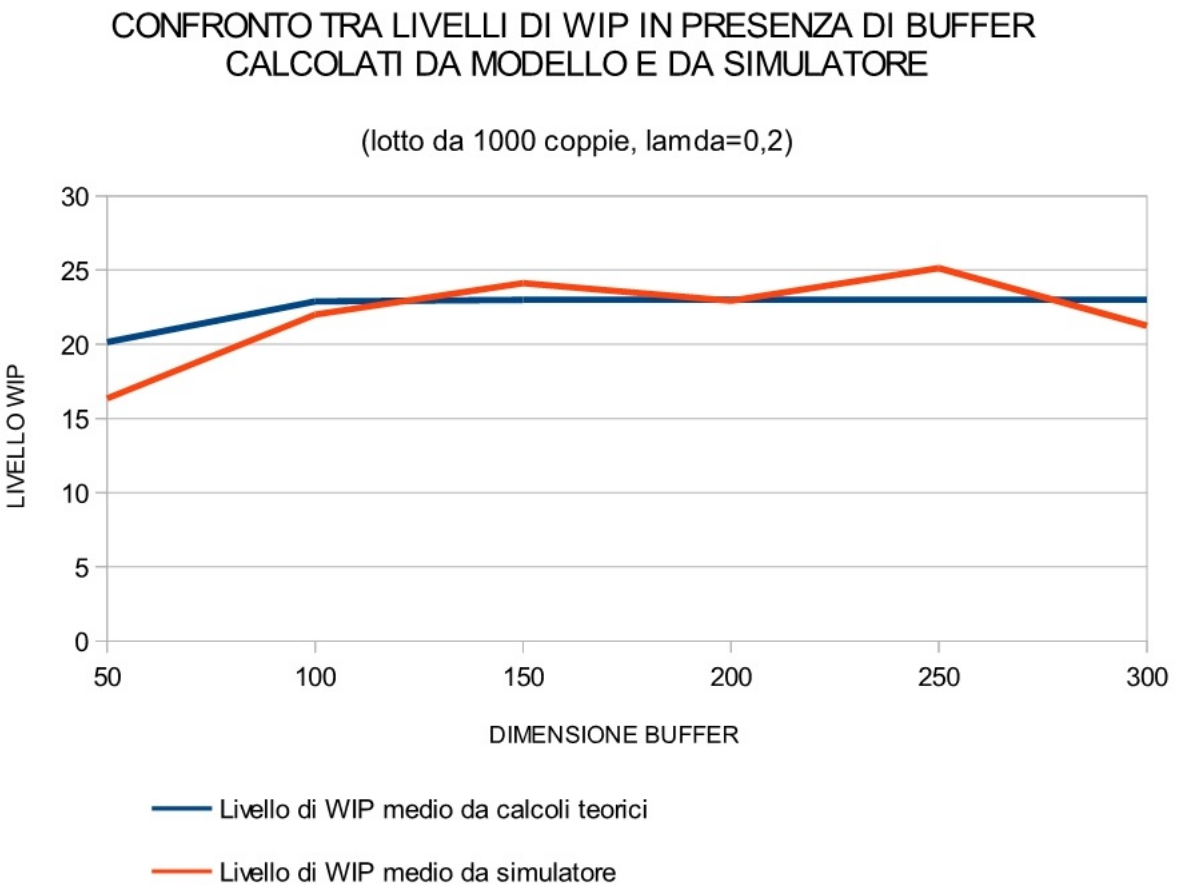


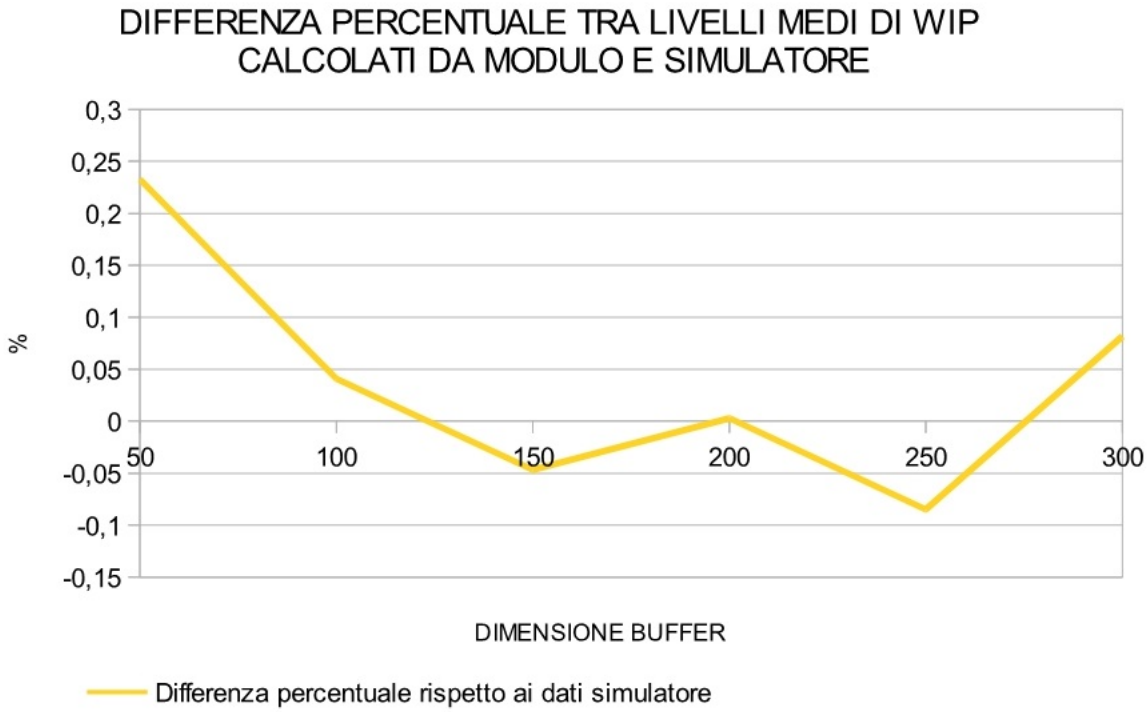
Dai grafici precedenti si evince che il dato modellizzato del tempo medio di attesa delle coppie all'interno della stazione di matching differisce dalla sua controparte simulata per un errore percentuale compreso tra -8% e +26% con una media pari a +5,27% ed una varianza di $1,11 * 10^{-2}$.





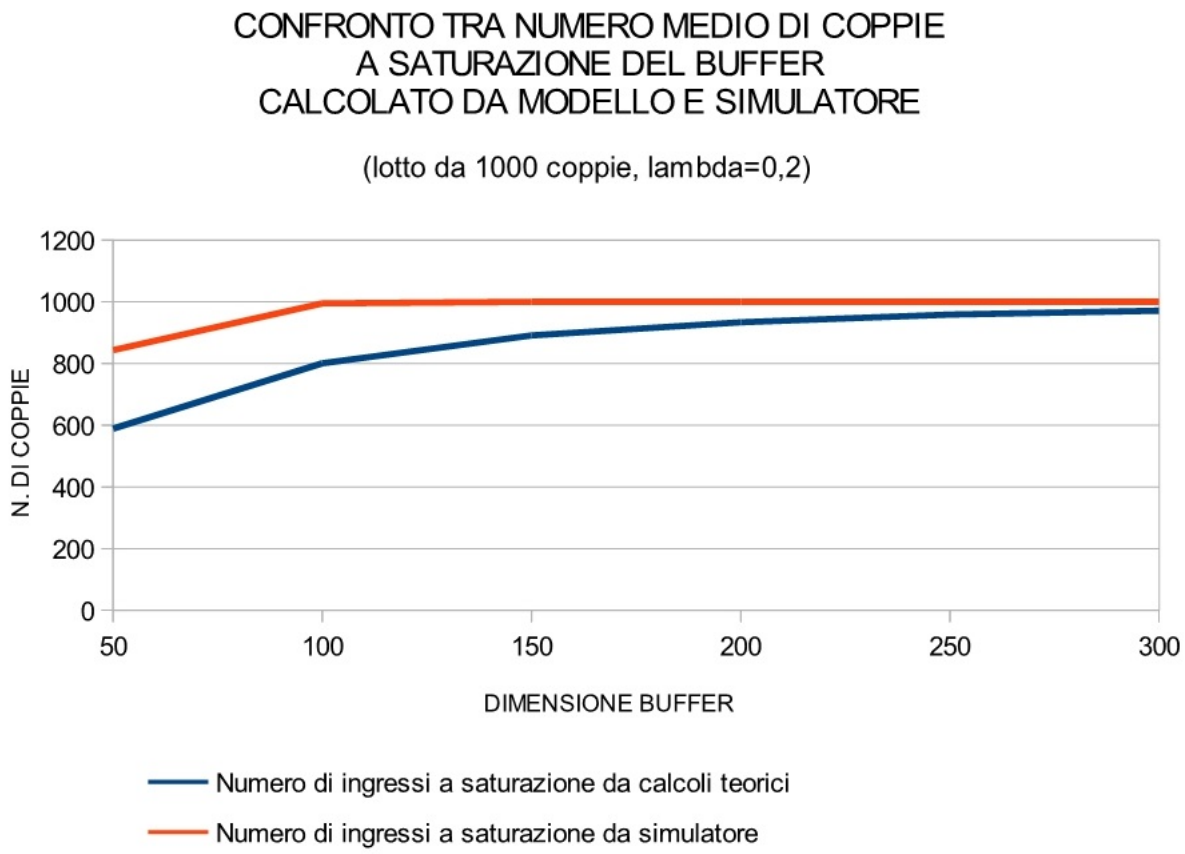
Dai grafici precedenti si evince che il dato modellizzato dell'intertempo medio di uscita delle coppie all'interno della stazione di matching differisce dalla sua controparte simulata per un errore percentuale compreso tra +1,2% e +2% con una media pari a +1,74% ed una varianza di $2,62 * 10^{-4}$.

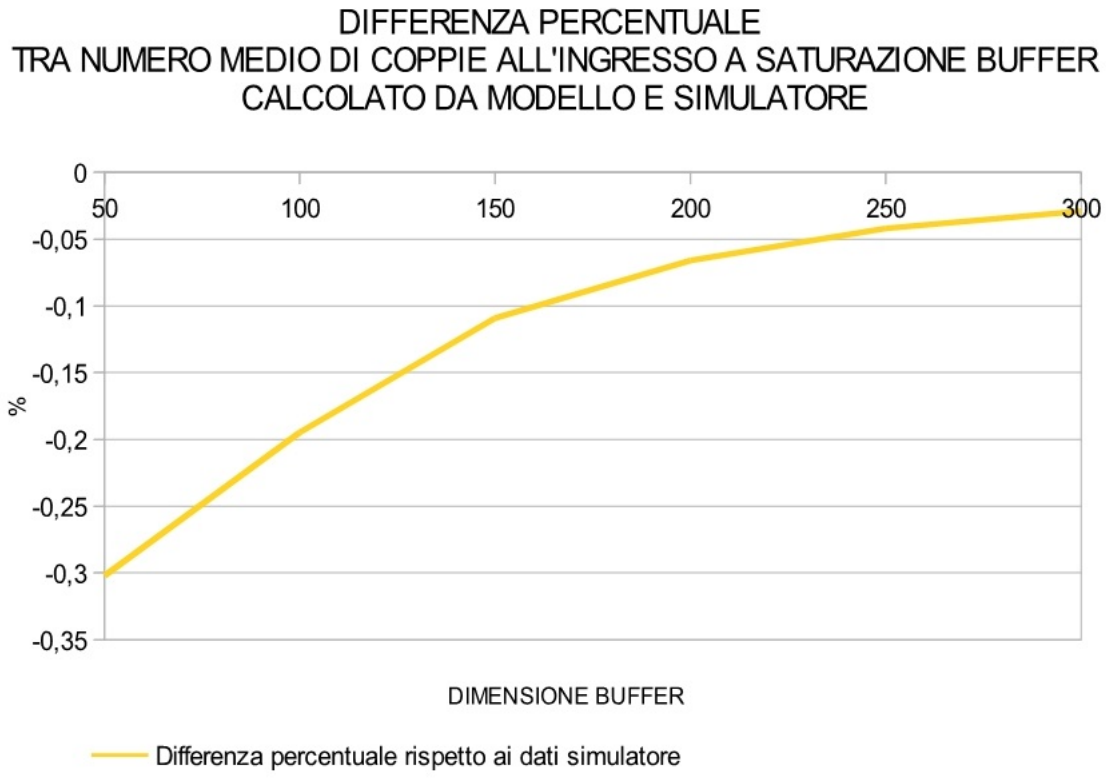




Dai grafici precedenti si evince che il dato modellizzato del livello medio di WIP all'interno della stazione di matching differisce dalla sua controparte simulata per un errore percentuale compreso tra -10% e +24% con una media pari a +3,78% ed una varianza di $1,06 * 10^{-2}$, con la fascia di errore massimo per

dimensioni del buffer inferiori al 5% della dimensione del lotto elaborato.





Dai grafici precedenti si evince che il dato modellizzato del numero medio di coppie giunte alla stazione di matching tra due successive saturazoni del buffer differisce dalla sua controparte simulata per un errore

percentuale compreso tra -35% e -3% con una media pari a -12,38% ed una varianza di $9,34 * 10^{-3}$, con andamento del valore assoluto della percentuale di errore decrescene al crescere della dimensione del buffer, con il picco massimo di errore per valori di quest'ultimo inferiori al 5% della dimensione del lotto di coppie elaborato.

Di seguito una tabella riassuntiva dei margini di errore tra modello e simulatore.

Riassunto dei dati delle percentuali di errore tra i dati modellizzati e simulati di un sistema in assenza di buffer					
Descrizione	**Limite inferiore**	**Limite superiore**	**Media**	**Varianza**	**Andamento**
Valore medio intertempi di uscita (media di valori medi)	-1,00%	1,00%	0,05%	$1,47*10^{-5}$	non presente
Valore medio intertempi di uscita (funzione globale)	-1,00%	1,00%	0,08%	$1,32*10^{-5}$	non presente
Varianza degli intertempi in ingresso (media delle varianze)	-2,50%	5,00%	0,33%	$1,27*10^{-4}$	non presente
Varianza degli intertempi in ingresso (funzione globale)	-2,50%	5,00%	0,58%	$1,35*10^{-5}$	non presente
Valore medio dei tempi di attesa (media dei valori medi)	-20,00%	20,00%	0.35%	$3,19*10^{-3}$	non presente
Valore medio dei tempi di attesa (Little, con WIP da tabella pertinente e TH ottenuto da media degli intertempi medi di uscita)	-20,00%	20,00%	-1,76%	$3,09*10^{-3}$	non presente
Valore medio dei tempi di attesa (funzione globale)	-20,00%	20,00%	-28,00%	$3,16*10^{-3}$	non presente
Valore medio degli intertempi di uscita (media dei valori medi)	10,00%	80,00%	40,00%	$2,045*10^{-2}$	non presente
Valore medio degli intertempi di uscita (Little, con WIP da tabella pertinente e CT pari al tempo medio di attesa)	30,00%	120,00%	75,49%	$2,89*10^{-2}$	non presente
Valore medio degli intertempi di uscita (media dei valori medi)	1,30%	7,00%	2,60%	$9,53*10^{-5}$	andamento decrescente al crescere delle dimensioni del lotto
Valore medio degli intertempi di uscita (Little, con WIP da tabella pertinente e CT pari al tempo medio di attesa)	2,70%	12,00%	4,80%	$39,53*10^{-5}$	andamento decrescente al crescere delle dimensioni del lotto

Riassunto dei dati delle percentuali di errore tra i dati modellizzati e simulati di un sistema in assenza di buffer					
Descrizione	**Limite inferiore**	**Limite superiore**	**Media**	**Varianza**	**Andamento**
Valore medio degli intertempi di uscita (funzione globale)	1,30%	6,00%	2,60%	$8,13*10^{-4}$	andamento decrescente al crescere delle dimensioni del lotto
Varianza degli intertempi di uscita (media delle varianze)	3,50%	16,00%	7,80%	$8,13* 10^{-4}$	andamento decrescente al crescere delle dimensioni del lotto
Varianza degli intertempi di uscita (funzione globale)	3,50%	16,00%	8,00%	$8,59* 10^{-4}$	andamento decrescente al crescere delle dimensioni del lotto
Valore del livello medio di WIP (tramite tabella probabilità livelli WIP)	-15,00%	12,00%	-3,50%	$2,89* 10^{-3}$	non presente
Valore del livello medio di WIP (Little, con TH ottenuto dall'intertempo medio di ingresso, CT pari al tempo di attesa medio)	-6,00%	26,00%	6,59%	$3,83* 10^{-3}$	non presente
Valore del livello medio di WIP (Little, con TH ottenuto dall'inertempo medio di uscita, CT pari al tempo di attesa medio)	-11,00%	15,00%	-1,43%	$2,86* 10^{-3}$	non presente
Valore del livello medio di WIP (Little, con TH ottenuto dall'intertempo medio di uscita, CT pari al tempo di attesa medio tramite funzione globale)	-12,00%	15,00%	-1,80%	$2,87* 10^{-3}$	non presente

Riassunto dei dati delle percentuali di errore tra i dati modellizzati e simulati di un sistema in presenza di buffer					
Descrizione	**Limite inferiore**	**Limite superiore**	**Media**	**Varianza**	**Andamento**
Valore del tempo medio di attesa	-8,00%	26,00%	5,27%	$1,11 * 10^{-2}$	non presente
Valore dell'intertempo medio di uscita	1,20%	2,00%	1,74%	$2,62 * 10^{-4}$	non presente
Valore del livello medio di WIP	-10,00%	24,00%	+3,78%	$1,06 * 10^{-2}$	errore massimo per buffer < 5% dimensione del lotto
Numero medio di coppie giunte alla stazione di matching tra due successive saturazoni del buffer	-35,00%	-3,00%	-12,38%	$9,34 * 10^{-3}$	valore assoluto della percentuale di errore tendente a 0 al crescere della dimensione del buffer, errore massimo per buffer < 5% dimensione lotto

UTILIZZI PLAUSIBILI				
Descrizione	**Utilizzo per...**	**Limite inferiore**	**Limite superiore**	**Valore centrale**
Valore medio intertempi di uscita (media di valori medi)	Stima attendibile durante attività ordinaria			X
Valore medio intertempi di uscita (funzione globale)	Stima attendibile durante attività ordinaria			X
Varianza degli intertempi in ingresso (media delle varianze)	Stima attendibile durante attività ordinaria			X
Varianza degli intertempi in ingresso (funzione globale)	Stima attendibile durante attività ordinaria			X
Valore medio dei tempi di attesa (media dei valori medi)	Stima di riferimento, mediamente precisa ma con alta varianza			X
Valore medio dei tempi di attesa (Little, con WIP da tabella pertinente e TH ottenuto da media degli intertempi medi di uscita)	Stima di riferimento, mediamente precisa ma con alta varianza			X
Valore medio dei tempi di attesa (funzione globale)	Stima di riferimento, mediamente precisa ma con alta varianza			X
Varianza dei tempi di attesa (media delle varianze)	Stima di practical worst case, dato modello sempre maggiore rispetto al simulato con media del 40%		X	
Varianza dei tempi di attesa (funzione globale)	Stima di worst case, dato modello sempre maggiore rispetto al simulato con media del 75%		X	
Valore medio degli intertempi di uscita (media dei valori medi)	Stima attendibile durante attività ordinaria		X	
Valore medio degli intertempi di uscita (Little, con WIP da tabella pertinente e CT pari al tempo medio di attesa)	Stima attendibile durante attività ordinaria		X	
Valore medio degli intertempi di uscita (funzione globale)	Stima attendibile durante attività ordinaria		X	
Varianza degli intertempi di uscita (media delle varianze)	Stima attendibile durante attività ordinaria		X	
Varianza degli intertempi di uscita (funzione globale)	Stima attendibile durante attività ordinaria		X	

UTILIZZI PLAUSIBILI				
Descrizione	**Utilizzo per...**	**Limite inferiore**	**Limite superiore**	**Valore centrale**
Valore del livello medio di WIP (tramite tabella probabilità livelli WIP)	Stima attendibile durante attività ordinaria			X
Valore del livello medio di WIP (Little, con TH ottenuto dall'intertempo medio di ingresso, CT pari al tempo di attesa medio)	Stima attendibile durante attività ordinaria		X	
Valore del livello medio di WIP (Little, con TH ottenuto dall'inertempo medio di uscita, CT pari al tempo di attesa medio)	Stima attendibile durante attività ordinaria			X
Valore del livello medio di WIP (Little, con TH ottenuto dall'intertempo medio di uscita, CT pari al tempo di attesa medio tramite funzione globale)	Stima attendibile durante attività ordinaria			X
Valore del tempo medio di attesa in presenza di buffer	Stima pessimistica di una performance ordinaria		X	
Valore dell'intertempo medio di uscita in presenza di buffer	Stima pessimistica di una performance ordinaria		X	
Valore del livello medio di WIP in presenza di buffer	Stima pessimistica di una performance ordinaria		X	
Numero medio di coppie giunte alla stazione di matching tra due successive saturazoni del buffer in presenza di buffer	Stima pessimistica di una performance ordinaria	X		

Conclusioni e ipotesi per studi successivi

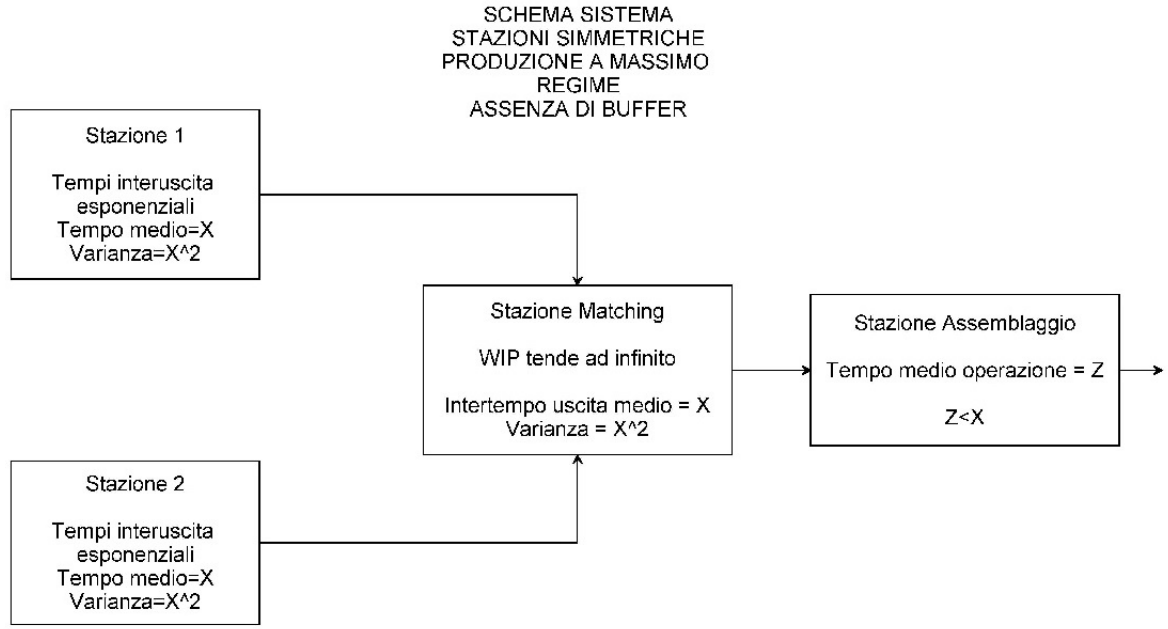
In generale, dai dati prodotti sia dal modello che dal simulatore riguardo al sistema preso in esame, in assenza di buffer si evince che:

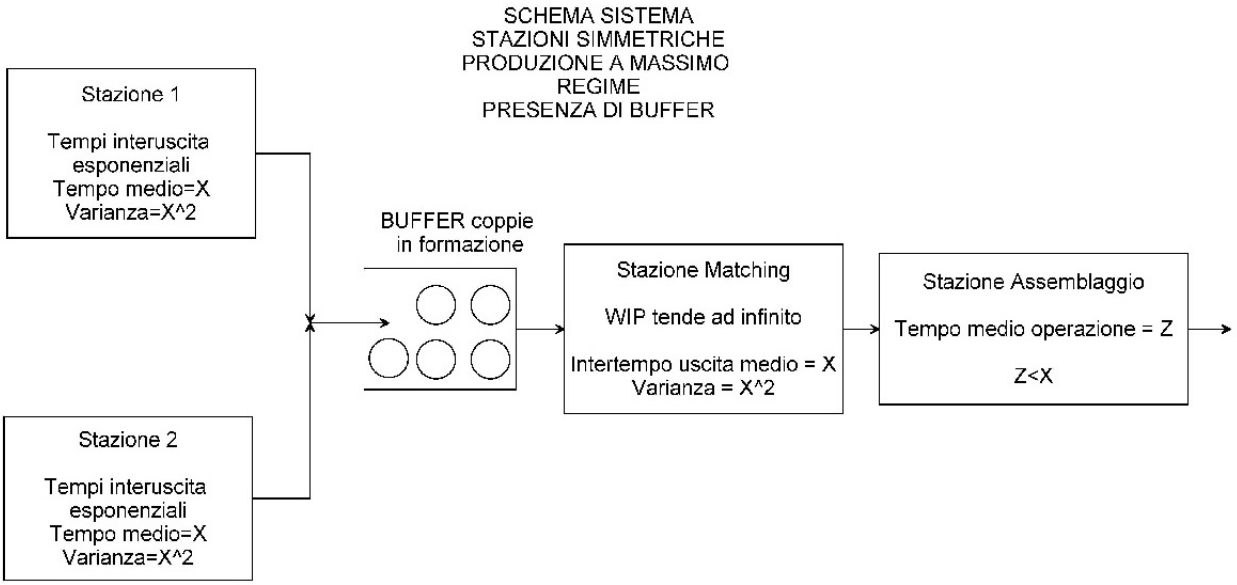
- il flusso di coppie all'ingresso della stazione di matching sarà sempre superiore a quello in uscita, generando un'utilizzazione superiore all'unità;
- la funzione di distribuzione di probabilità degli intertempi di uscita e di ingresso, al tendere ad infinito delle dimensioni del lotto, sono approssimabili con una esponenziale avente lambda pari a quella delle due stazioni a monte (la perdita di precisione conseguente è esigua);

- al tendere ad infinito delle dimensioni del lotto da produrre (produzione continua) i nostri livelli di WIP saranno destinati ad andare fuori controllo.

Urge quindi trovare delle soluzioni attive o preventive al fine del controllo dei livelli di works in progress.

Sul fronte delle prime c'è l'inserimento di un buffer all'ingresso della stazione di macthing, con i pro (livello di WIP sotto controllo) e i contro (riduzione del TH, tempi di inattività forzata di una delle due stazioni a monte) di questa soluzione.



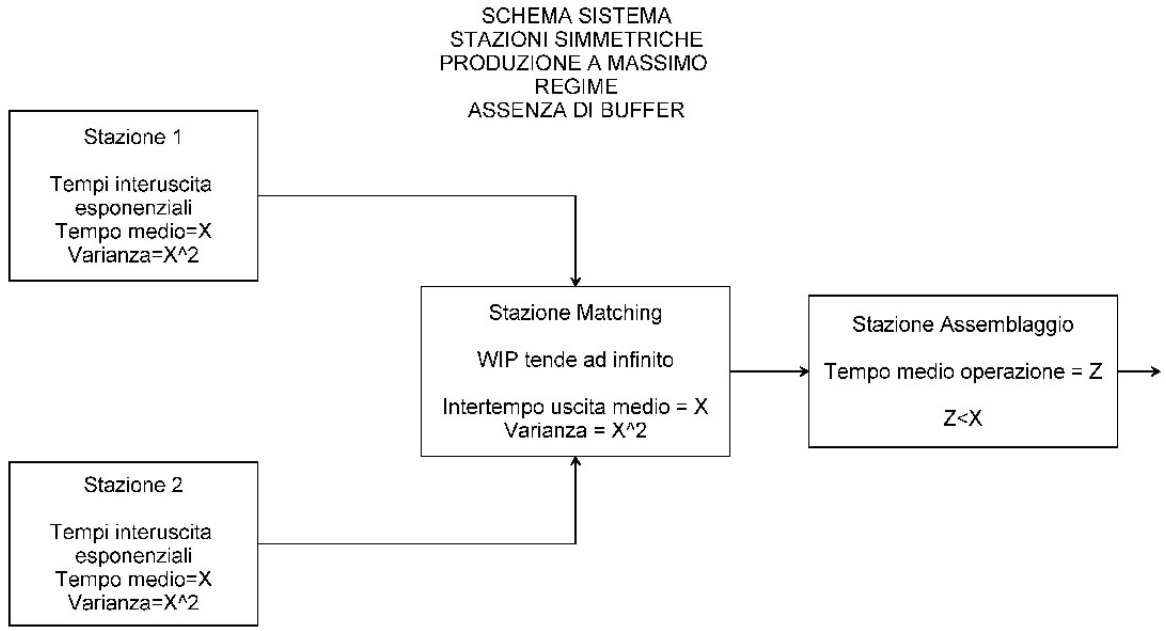


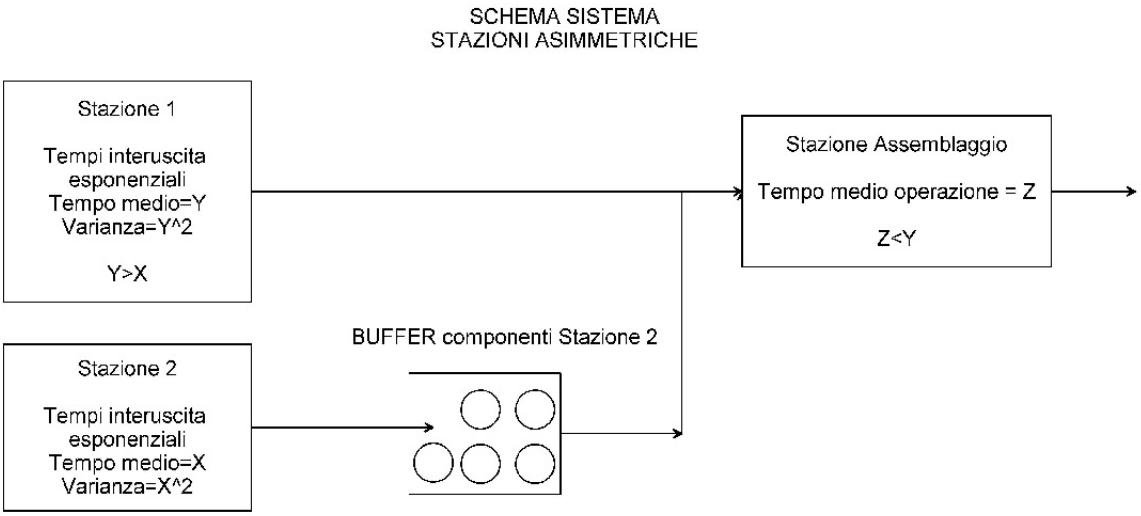
Da notare che, per grandi dimensioni dei lotti o in regime di produzione continua, essendo le distribuzioni degli intertempi di ingresso e di uscita convergenti ad una distribuzione esponenziale, si potrebbe semplificare il sistema come una linea composta da una singola stazione a monte ed una successiva di assemblaggio, con gli intertempi di uscita della prima e i tempi di processo dalla seconda definiti dalla medesima distribuzione esponenziale. Andando ad implementare un buffer e gestendolo con le formule standard si avrebbero delle discrete approssimazioni a bassissimo calcolo computazionale.

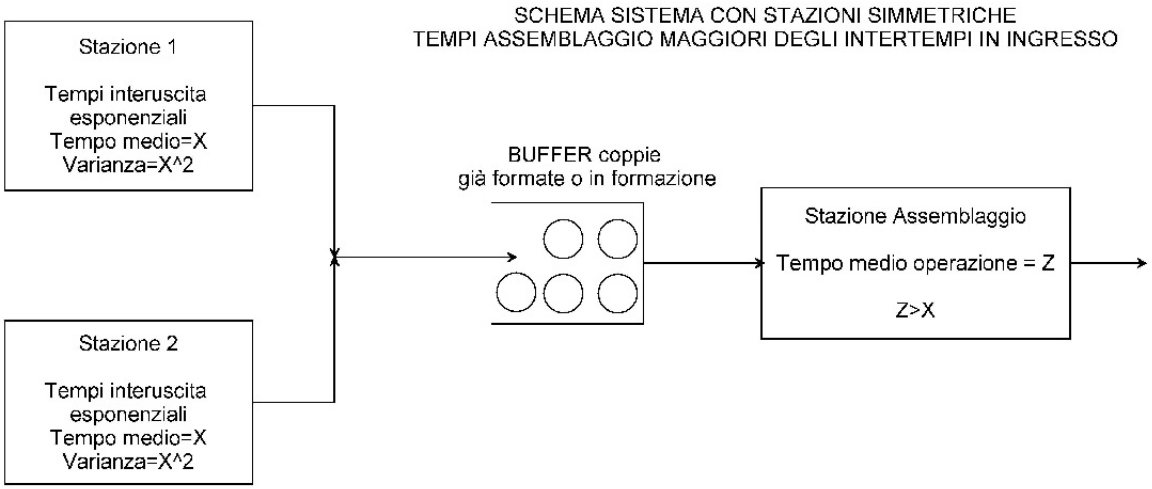
Le soluzioni preventive mirano invece a cambiare le caratteristiche del sistema in modo da prevenire il fattore scatenante della perdita di controllo sui livelli di WIP, causata dalla simmetria delle due linee a

monte assieme all'assenza di una strozzatura a valle della stazione di accoppiamento. Questa modifica può avvenire sia in senso fisico (strutturando fisicamente le stazioni a monte facendo sì che abbiano parametri diversi da quelli innescanti i problemi), sia in senso gestionale (governando il funzionamento delle linee a monte in base ad una politica diversa rispetto a «produzione a pieno regime»).

Lo scopo è rendere aprioristicamente nullo, o ininfluente ai fini del calcolo del CT complessivo della linea, il tempo di attesa tra i componenti per la formazione della coppia prima dell'assemblaggio.







Si può ottenere ciò riducendo creando delle differenze nella capacità di output delle linee a monte, annullando la loro simmetria, rimodellando concettualmente così:

- due stazioni a monte, aventi TH in uscita esponenziali con medie diverse, di cui la più lenta sarà la Master e la più veloce la Slave;
- un buffer per i soli componenti della linea Slave;
- una stazione di assemblaggio, caratterizzata dai parametri standard di una qualsiasi stazione di processo.

La fase di incontro potrà essere ignorata, in quanto la stazione Slave difficilmente sarà in ritardo rispetto alla Master. Per questo si dovrà introdurre un sistema di blocco, quale è il buffer, per prevenire alla linea Slave di generare nel tempo una coda di dimensioni tendenti ad infinito all'ingresso della stazione di assemblaggio.

Ipotetici vantaggi di questo approccio sono:

- facilità di calcolo e di gestione dei flussi (l'unica ramo da computare sarebbe quello della linea Master);
- blocchi di produzione per raggiunta saturazione del buffer solo su una delle due linee a monte (la Slave) con conseguente semplificazione dei sistemi di controllo;
- riduzione dei costi fissi inerenti il magazzino WIP a monte della stazione di assemblaggio, in quanto atto ad ospitare prevalentemente e sicuramente solo un modello di componente anziché entrambi;
- ampia possibilità di gestione delle operazioni di setup e manutenzione ordinaria e straordinaria sulla linea Slave, in quanto capace di produrre in anticipo il fabbisogno richiesto durante i periodi di inattività.

Ipotetici vantaggi:

- surplus di capacità produttiva e suo relativo costo sulla linea Slave;
- criticità della linea Master ai fini della produzione;
- limitata elasticità gestionale ed operativa.

E' plausibile la possibilità di raggiungere un risultato simile facendo sì che i tempi fisici di assemblaggio siano superiori a quelli di interuscita dalla stazione di matching, creando una coda di coppie in attesa del loro processamento. Ciò renderebbe superfluo il calcolo del tempo passato dai singoli componenti in attesa del proprio complementare in quanto, essendo il TH in ingresso superiore alla capacità di smaltimento, a regime i componenti si incontrerebbero all'interno della coda stessa. Il problema di gestione dei livelli di WIP, tramite buffer, verrebbe trasferito alla più gestibile stazione di assemblaggio. Si potrebbe così semplificare il sistema:

- una stazione a monte, rappresentante le stazioni a monte realmente esistenti e caratterizzata dalla loro stessa distribuzione esponenzialmente, il suo TH farebbe riferimento a delle coppie già pronte all'assemblaggio;
- un buffer per le coppie, adibito al blocco di entrambe le linee a monte;
- la stazione di assemblaggio.

Ipotetici vantaggi di questo approccio sono:

- facilità di calcolo e gestione dei flussi, tutte le formule standard sarebbero facilmente applicabili;
- ampia possibilità di gestione delle operazioni di setup e manutenzione ordinaria e straordinaria su entrambe le linee a monte, in quanto capaci di produrre in anticipo il necessario alla copertura del fabbisogno richiesto durante i periodi di inattività;
- mancanza di criticità a monte della stazione di assemblaggio.

Ipotetici svantaggi sono:

- surplus di capacità produttiva e suo relativo costo su entrambe le linee a monte;
- costi fissi di magazzino superiori alla prima soluzione proposta, in quanto predisposto allo stoccaggio di due tipologie di componenti diversi anziché ad una sola di esse;
- criticità della stazione di assemblaggio.

Una soluzione ibrida potrebbe essere il sovradimensionamento della capacità produttiva di entrambe le stazioni a monte e della stazione di assemblaggio rispetto al flusso richiesto, permettendo una maggiore elasticità ed eliminando ogni fattore di criticità a fronte di maggiori costi fissi.

Ipotetici vantaggi:

- ogni linea o stazione può permettersi di affrontare un ritardo produttivo, in quanto capace di recuperarlo sfruttando la capacità in eccesso;
- ampia possibilità delle stazioni a monte di programmare lavori di manutenzione ordinaria o straordinaria, producendo in anticipo il fabbisogno richiesto per il loro periodo di inattività;
- discreto ventaglio di opzioni gestionali per far fronte ad eventuali imprevisti;
- possibilità di variare il funzionamento reale modellandolo su una delle precedenti soluzioni proposte in base alle necessità di produzione;
- assenza di criticità.

Ipotetici svantaggi:

- costi fissi delle linee a monte e della stazione di assemblaggio decisamente superiori allo stretto necessario;
- costi fissi di magazzino superiori alla prima soluzione proposta, in quanto deve essere predisposto allo stoccaggio di due tipologie di componenti diversi anziché ad una sola di esse;
- maggiori costi gestionali dovuti ad un lavoro di monitoraggio continuo della performace del sistema per mantenerlo nello stato desiderato ed evitare che torni al suo stato naturale (saturazione del buffer delle coppie casuale, con conseguente blocco della linea a monte che più performante, per frutto del caso e non di una scelta specifica).

Un'ultima possibilità è trasformare il tutto in un sistema kanban, risolvendo a monte ogni possibile problema di coordinamento. Ricordiamo che nei sistemi PULL la produzione in un centro si attiva solo su specifiche richieste giunte tramite cartellino dalle stazioni a valle, quando consumano le loro scorte dedicate di componenti. Di conseguenza la dimensione del lotto da produrre (da 1 a X) è prestabilita, così come la dimensione di tutti i magazzini dedicati delle stazioni del sistema. Essendo la capacità

massima produttiva, le sue tempistiche e dinamiche predeterminate nella fase di progettazione del sistema e mantenute tramite rigide, quanto semplici, regole di funzionamento, il sistema di produzione non può essere messo sotto stress, in quanto darebbe sempre la stessa risposta: «Il prodotto sarà pronto quando l'avremo finito». Questa rigidità fornisce delle caratteristiche quasi deterministiche alle tempistiche richieste alla consegna di un'eventuale ordine, al costo dell'impossibilità di forzare i ritmi produttivi (una qualsiasi stazione non può anticipare una produzione futura se non in casi eccezionali). Il tutto risulta in una divisione fisiologica dei compiti ben precisa in cui la produzione, in ogni sua stazione, sarà responsabile di evadere gli ordini ricevuti da valle nel minor tempo possibile necessario al mantenimento della qualità, e in cui il settore vendite e marketing dovrà gestire la domanda stabilendo date di consegna coerenti con le timeline fornite dalla produzione.

APPENDICE

Flow chart della struttura del programma

Il programma inizia con un'elaborazione preventiva delle produttorie fino ad un valore X preimpostato al suo interno. Come visto nella sezione teorica riguardante la modellizzazione, queste sono spesso presenti nelle formule utilizzate, quindi il loro calcolo preventivo ridurrà drasticamente i tempi computazionali successivi a fronte di un'ulteriore ma esigua quantità di memoria RAM occupata dal programma.

Il primo menù permette di scegliere se eseguire le operazioni in modalità manuale o di creare/eseguire un file di schedulazione lavori.

La modalità utente permette, tramite una serie di sub-menù annidati, la scelta diretta e l'esecuzione immediata di una qualsiasi delle operazioni implementate.

Queste sono divise in tre famiglie:

- modellizzazione teorica;
- simulazione;
- comparazione tra i dati modellizzati e simulati.

Tutte le operazioni implementate nel modellizzatore sono riferite ad un sistema avente due linee a monte, aventi entrambe flussi di uscita caratterizzati dalla medesima distribuzione esponenziale dei tempi, ognuna adibita esclusivamente alla produzione di uno dei due componenti che riforniranno la seguente stazione di matching a valle.

In questa sezione, causa la complessità dei calcoli richiesta, il programma prevede la creazione di due livelli di matrici di base, contenenti i dati necessari allo studio di tutti i lotti aventi una dimensione minore o uguale a quella da loro coperta (N), tutte impostate su $\lambda = 1$.

Vi sono:

- le matrici dei coeficienti delle funzioni di distribuzione di probabilità dei tempi di interarrivo, interuscita e di attesa locali e globali;
- le matrici dei coeficienti delle funzioni di probabilità cumulata dei tempi di interarrivo, interuscita e di attesa locali;
- le matrici dei coeficienti delle funzioni di probabilità cumulata dei tempi di interarrivo e di attesa globali;
- la tabella delle probabilità dei livelli di WIP all'ingresso dell'i-esima coppia in assenza di buffer;
- matrice dei valori medi e delle varianze adimensionali (sempre calcolati per $\lambda = 1$) delle distribuzioni locali precedentemente elencate;
- matrice contenente, per ogni lotto fino ad una dimensione N, le media dei valori medi e delle varianze delle distribuzioni locali precedentemente elencate;
- matrice contenente i valori medi e le varianze globali per ogni lotto avente dimensione inferiore o uguale a N.

Queste matrici, una volta calcolate, vengono salvate su appositi file che possono esser caricati in memoria all'occorrenza.

Questo permette, a fronte di un maggior utilizzo di memoria su disco rigido, di concentrare i tempi

di calcolo in un unico momento iniziale che sarà utile a coprire preventivamente un numero elevato di casi specifici in assenza di buffer, riducendo il tempo di elaborazione totale (caricamento dei dati incluso) del 90% circa.

In egual modo lo stesso principio è stato applicato a questi ultimi, permettendo all'utente di salvarne i risultati su file ricaricabili dal programma per la modellizzazione degli stessi in presenza di buffer di varie dimensioni.

I files di output dal programma in questa fase si dividono in due categorie:

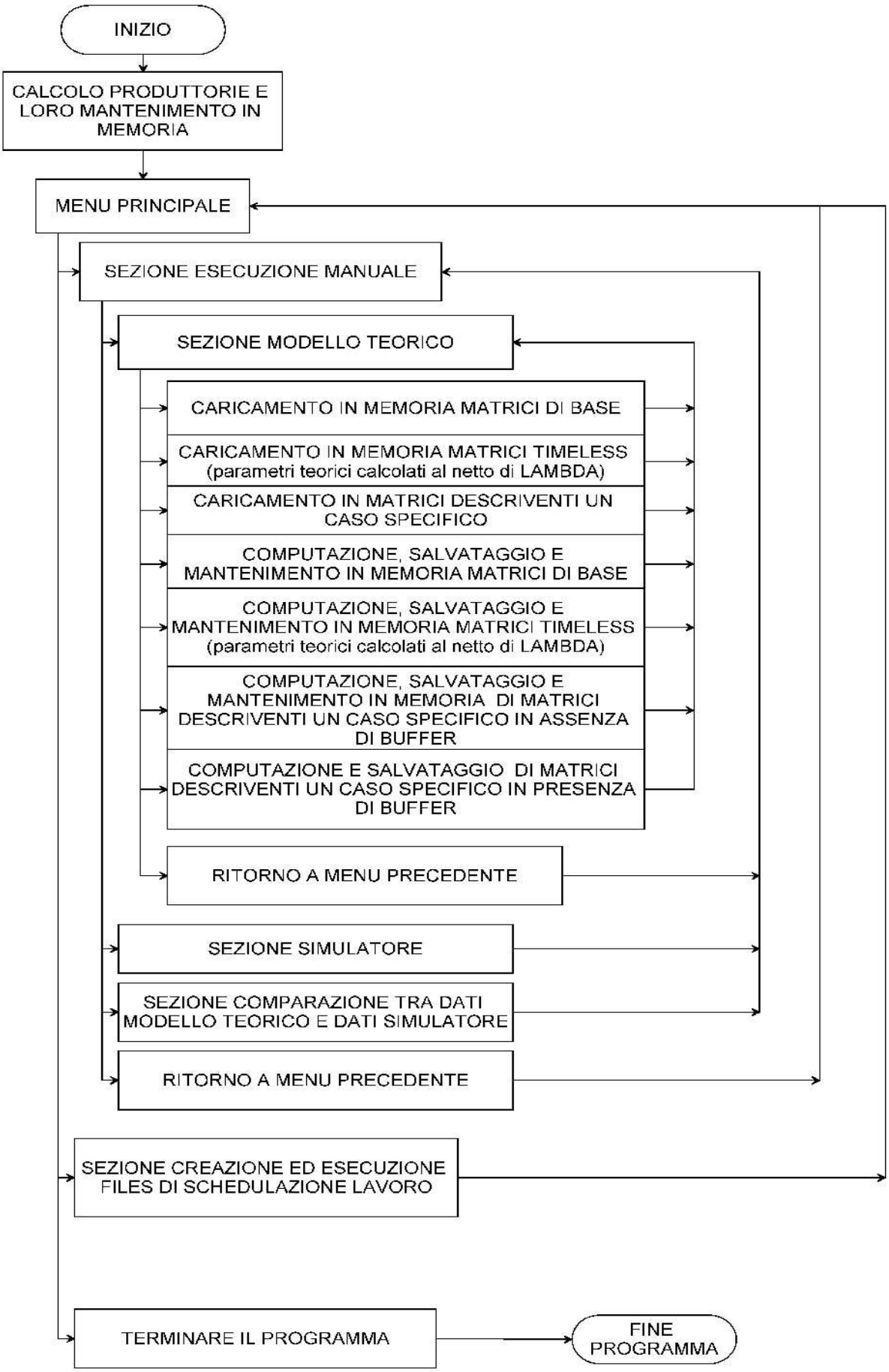
- files dati destinati a rielaborazioni successive;
- files utente.

I primi possiedono le seguenti caratteristiche:

- presenza di una prima linea contenente la chiave di lettura dei dati, al fine di un caricamento più rapido dei dati in fase di upload dati da parte del programma;
- assenza di campi esplicativi della natura dei dati comprensibili ad un utente che sia ignaro di essa;
- possibilità di lettura di visualizzazione dei dati attraverso Excel o compatibile, ai fini di eventuali controlli sul funzionamento del programma;
- i loro nomi sono un codice generato automaticamente dal programma che indica la loro natura (file dati interni al programma), il tipo di dati in essi contenuti, il range ai quali saranno applicabili e il contesto che li ha generati.

Gli altri:

- non verranno mai più ricaricati in memoria dal programma, per nessun fine;
- presentano campi esplicativi della natura dei dati ivi contenuti;
- sono caricabili da Excel o compatibile;
- i loro dati sono trascritti in un formato che ne permetta l'utilizzo diretto tramite Excel o compatibile;
- i loro nomi sono un codice generato automaticamente dal programma che indica la loro natura (file utente), il tipo di dati in essi contenuti e il contesto che li ha generati.


INIZIO
CALCOLO PRODUTTORIE E LORO MANTENIMENTO IN MEMORIA
MENU PRINCIPALE
SEZIONE ESECUZIONE MANUALE
SEZIONE MODELLO TEORICO
CARICAMENTO IN MEMORIA MATRICI DI BASE
CARICAMENTO IN MEMORIA MATRICI TIMELESS (parametri teorici calcolati al netto di LAMBDA)
CARICAMENTO IN MATRICI DESCRIVENTI UN CASO SPECIFICO
COMPUTAZIONE, SALVATAGGIO E MANTENIMENTO IN MEMORIA MATRICI DI BASE
COMPUTAZIONE, SALVATAGGIO E MANTENIMENTO IN MEMORIA MATRICI TIMELESS (parametri teorici calcolati al netto di LAMBDA)
COMPUTAZIONE, SALVATAGGIO E MANTENIMENTO IN MEMORIA DI MATRICI DESCRIVENTI UN CASO SPECIFICO IN ASSENZA DI BUFFER
COMPUTAZIONE E SALVATAGGIO DI MATRICI DESCRIVENTI UN CASO SPECIFICO IN PRESENZA DI BUFFER
RITORNO A MENU PRECEDENTE
SEZIONE SIMULATORE
SEZIONE COMPARAZIONE TRA DATI MODELLO TEORICO E DATI SIMULATORE
RITORNO A MENU PRECEDENTE
SEZIONE CREAZIONE ED ESECUZIONE FILES DI SCHEDULAZIONE LAVORO
TERMINARE IL PROGRAMMA
FINE PROGRAMMA

La sezione di simulazione consente quattro operazioni:

- simulazione singola in assenza di buffer;
- simulazione singola in presenza di buffer;
- esecuzione di un blocco di simulazioni singole in assenza di buffer e calcolo della media dei loro parametri;
- esecuzione di un blocco di simulazioni singole in presenza di buffer e calcolo della media dei loro parametri.

A differenza delle operazioni di modellizzazione, in questa sezione il numero di linee a monte non è fissato a due, ma è una variabile governabile dall'utente. Permane invece il vincolo di identicità della funzione esponenziale governante l'output delle stazioni a monte.

Le due simulazioni singole sono il motore principale di questa sezione, in quanto su di loro si basano i blocchi di simulazioni.
L'accesso diretto alla singola simulazione è stato previsto, nonostante il loro scarso valore ai fini statistici, per consentire un maggior controllo sulla correttezza delle operazioni da loro svolte, il tutto tramite delle opzioni di output su file a loro specifiche che rendono consultabili su Excel, o compatibile, i dati prodotti in esecuzione (le timelines, i valori medi di tempi ed intertempi, le variazioni dei livelli di WIP e il livello medio di questo, numero di coppie medio in ingresso tra una saturazione e la successiva del buffer quando presente).

I due esecutori di blocchi di simulazioni singole, con o senza buffer, sono basati su un ciclo interno che ad ogni passaggio esegue la simulazione singola correlata, raccogliendone i dati medi finali. Al termine di questo verranno elaborate le medie e generati i file di output.
La rilevanza statistica dei dati prodotti da un esecutori di blocco è direttamente proporzionale al numero di simulazioni che gli si è ordinato di eseguire.

Come nella sezione precedente, i nomi dei files di output rappresentano un codice, generato automaticamente dal programma, contenente un riassunto dei dati in questi contenuti.

L'introduzione della sezione dedicata alla comparazione nasce dalla necessità di valutare i dati ottenuti tramite modellizzazione rispetto a quelli simulati. Pur essendo possibile svolgere manualmente il compito all'esterno del programma attraverso i files utente, questa specifica parte, svolgendo in sequenza il modellizzatore e l'esecutore di blocchi a lui specifico e riorganizzando i dati all'interno di specifici files di output, permette una consultazione degli elaborati più rapida ed efficiente.

Entrambi i comparatori, in realtà, sono studiati per elaborare in sequenza un range di casi possibili, al fine di una più rapida visualizzazione degli eventuali trend di errore del modellizzatore rispetto a determinate dimensioni.
I due comparatori implementati sono:

- comparatore su un range di casi specifici (basato sulla dimensione del lotto in analisi) in assenza di buffer;
- comparatore su un range di casi specifici (basato sulla dimensione del buffer) in presenza di buffer.

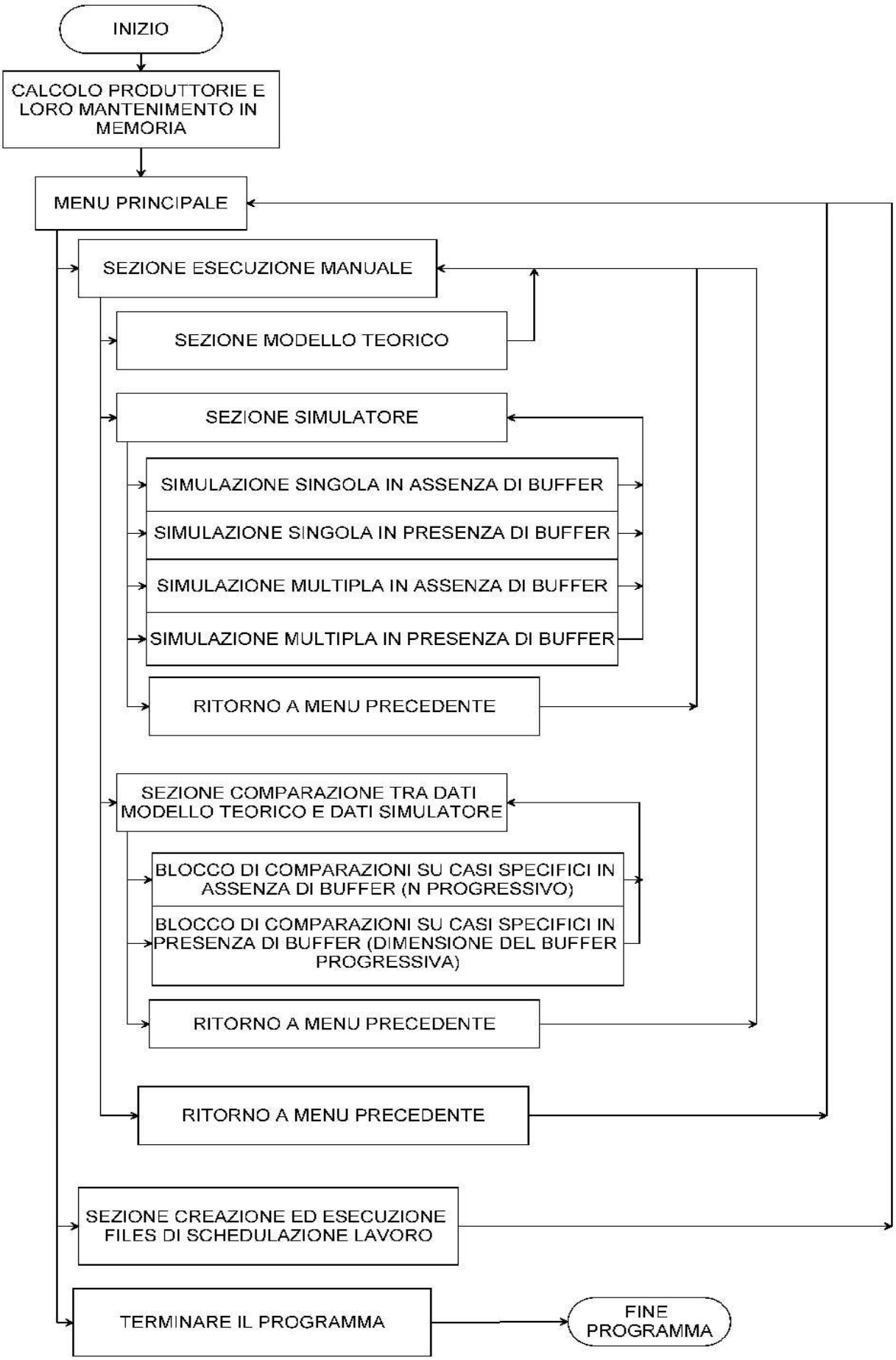
Entrambi, durante la loro esecuzione, richiamando le funzioni di esecuzione di blocchi di simulazioni, ne fisseranno automaticamente il numero di macchine a monte a due.

I files di output relativi a questa sezione presentano le seguenti caratteristiche:

- presenza di una famiglia di files, ognuno contentente un tipo di dati specifico;
- ogni blocco di comparazioni genererà un codice che diventerà parte integrante del nome di

tutti i files da esso prodotti

- nomi rappresentante un codice, generato automaticamente dal programma, contenente il codice il blocco a cui si riferiscono e il tipo di dati contenuti ;
- per ogni tipo di dato, saranno presenti i confronti tra quelli modellizzati e simulati e la percentuale di errore tra i primi ed i secondi.


INIZIO
CALCOLO PRODUTTORIE E LORO MANTENIMENTO IN MEMORIA
MENU PRINCIPALE
SEZIONE ESECUZIONE MANUALE
SEZIONE MODELLO TEORICO
SEZIONE SIMULATORE
SIMULAZIONE SINGOLA IN ASSENZA DI BUFFER
SIMULAZIONE SINGOLA IN PRESENZA DI BUFFER
SIMULAZIONE MULTIPLA IN ASSENZA DI BUFFER
SIMULAZIONE MULTIPLA IN PRESENZA DI BUFFER
RITORNO A MENU PRECEDENTE
SEZIONE COMPARAZIONE TRA DATI MODELLO TEORICO E DATI SIMULATORE
BLOCCO DI COMPARAZIONI SU CASI SPECIFICI IN ASSENZA DI BUFFER (N PROGRESSIVO)
BLOCCO DI COMPARAZIONI SU CASI SPECIFICI IN PRESENZA DI BUFFER (DIMENSIONE DEL BUFFER PROGRESSIVA)
RITORNO A MENU PRECEDENTE
RITORNO A MENU PRECEDENTE
SEZIONE CREAZIONE ED ESECUZIONE FILES DI SCHEDULAZIONE LAVORO
TERMINARE IL PROGRAMMA
FINE PROGRAMMA

La possibilità di creare ed eseguire dei files di schedulazione è stata implementata per permette elaborazioni dati di lunga durata, anche nell'ordine dei giorni, senza la presenza di un operatore.

La fase di creazione viene guidata dal programma tramite una serie di menù adibiti alla cernita delle operazioni che si desidera eseguire, con conseguente input guidato dei dati base necessari per il corretto funzionamento, e controlli inabilitanti la scelta di operazioni che non possano essere eseguite senza previo inserimento di ulteriori istruzioni. Alla fine di questa procedura, se giunta a buon fine, verrà generato un file di schedulazione contente le istruzioni in un linguaggio specifico del programma.
La sezione "crea file di schedulazione" è stata implementata per:

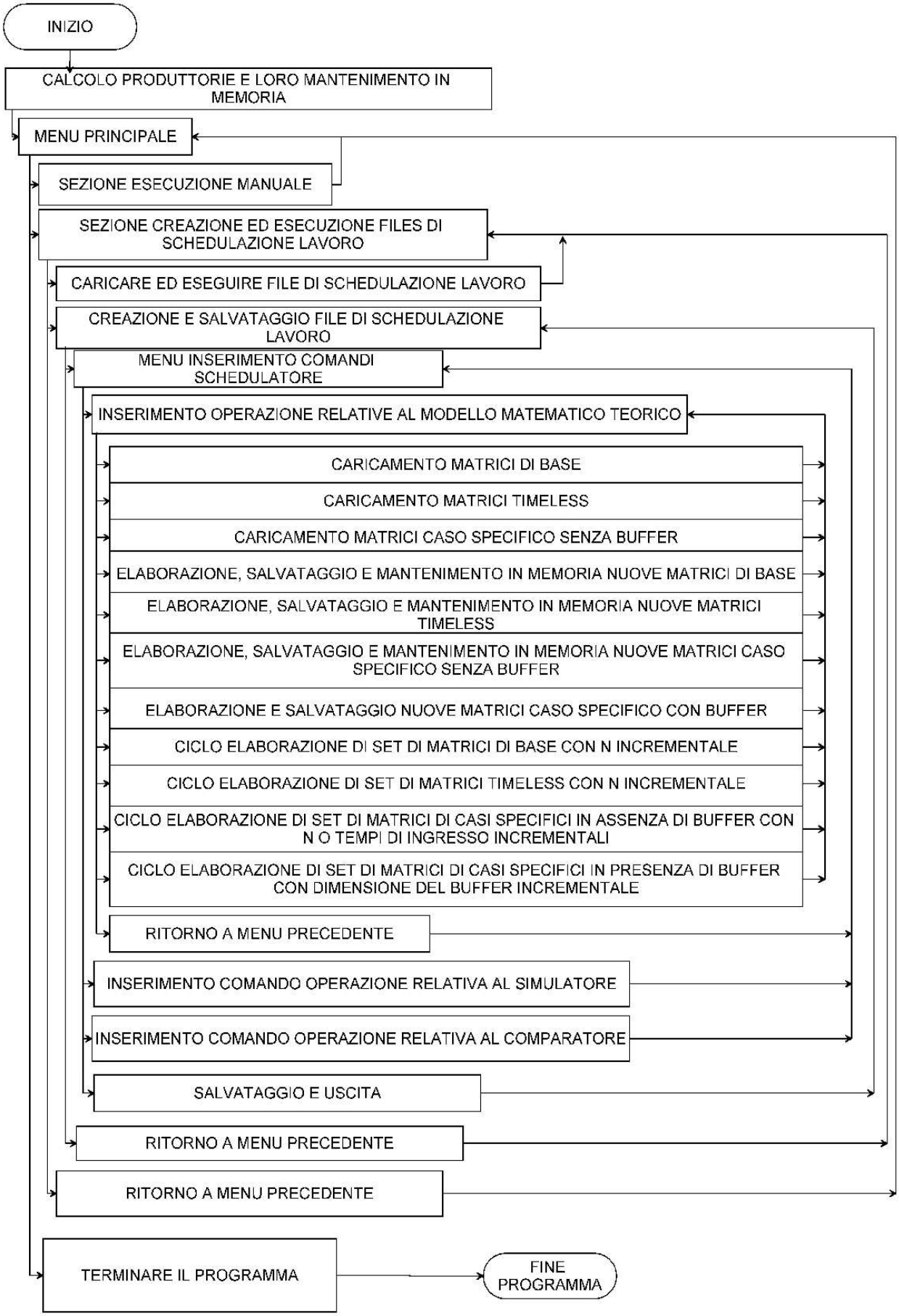
- evitare all'utente di dover memorizzare un linguaggio utile solo all'interno di questo programma;
- ridurre al minimo la possibilità di errori nei dati delle istruzioni, potenziale (se non sicura) causa di successivi crash del programma.

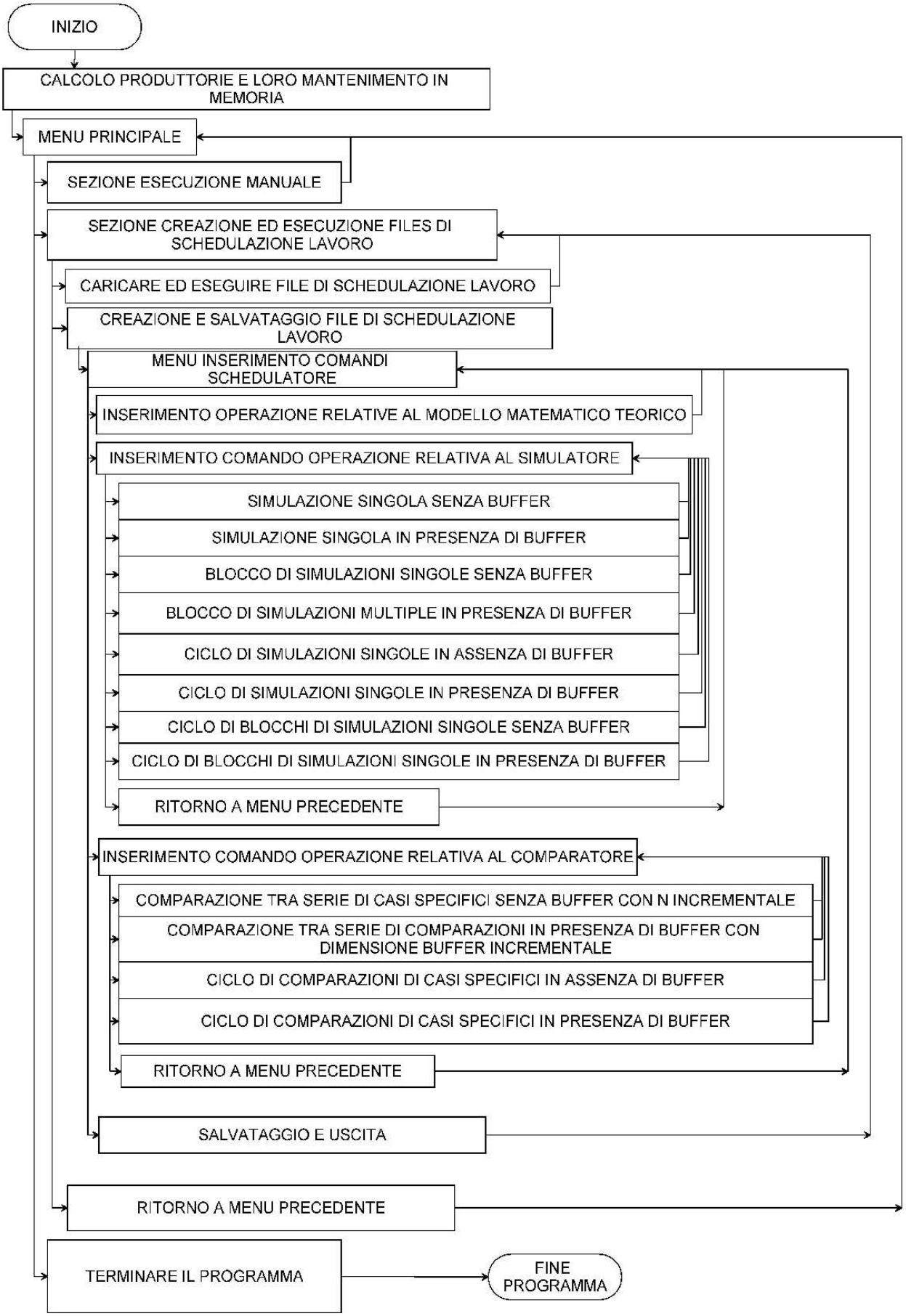
Le operazioni selezionabili per l'esecuzione in modalità schedulazione rispecchiano tendenzialmente quelle già presenti in modalità manuale, con l'aggiunta di qualche scorciatoia (tutte le voci «Creazione ciclo di...»).

Da quanto illustrato finora risulta palese che all'interno del programma esistono delle variabili di stato che indicano alle varie funzioni custom, coinvolte nelle varie operazioni, in quale modalità si trova la richiamante.
Gli stati possibili sono:

- chiamata "manuale diretta", quando la funzione è richiamata direttamente dal menù della sezione manuale
- chiamata "schedulatore diretta", quando la funzione è richiamata esplicitamente da una linea di comando contenuta nel file di schedulazione in fase di esecuzione;
- chiamata "da altra funzione, modalità ininfluente", quando una funzione è richiamata da una gerarchicamente superiore il cui stato (eseguita in modalità manuale, schedulatore o altro) è ininfluente ai fini del lavoro che deve svolgere;
- chiamata "comparatore manuale", quando una funzione è richiamata dal comparatore, che è stato richiamato a sua volta da comando diretto in modalità manuale;
- chiamata da "comparatore in schedulazione", quando una funzione è richiamata dal comparatore, che è stato richiamato a sua volta da un comando diretto contenuto nel file di schedulazione in fase di esecuzione.

In base a queste variabili il modo in cui le varie funzioni custom operano può variare anche drasticamente.


INIZIO
CALCOLO PRODUTTORIE E LORO MANTENIMENTO IN MEMORIA
MENU PRINCIPALE
SEZIONE ESECUZIONE MANUALE
SEZIONE CREAZIONE ED ESECUZIONE FILES DI SCHEDULAZIONE LAVORO
CARICARE ED ESEGUIRE FILE DI SCHEDULAZIONE LAVORO
CREAZIONE E SALVATAGGIO FILE DI SCHEDULAZIONE LAVORO
MENU INSERIMENTO COMANDI SCHEDULATORE
INSERIMENTO OPERAZIONE RELATIVE AL MODELLO MATEMATICO TEORICO
CARICAMENTO MATRICI DI BASE
CARICAMENTO MATRICI TIMELESS
CARICAMENTO MATRICI CASO SPECIFICO SENZA BUFFER
ELABORAZIONE, SALVATAGGIO E MANTENIMENTO IN MEMORIA NUOVE MATRICI DI BASE
ELABORAZIONE, SALVATAGGIO E MANTENIMENTO IN MEMORIA NUOVE MATRICI TIMELESS
ELABORAZIONE, SALVATAGGIO E MANTENIMENTO IN MEMORIA NUOVE MATRICI CASO SPECIFICO SENZA BUFFER
ELABORAZIONE E SALVATAGGIO NUOVE MATRICI CASO SPECIFICO CON BUFFER
CICLO ELABORAZIONE DI SET DI MATRICI DI BASE CON N INCREMENTALE
CICLO ELABORAZIONE DI SET DI MATRICI TIMELESS CON N INCREMENTALE
CICLO ELABORAZIONE DI SET DI MATRICI DI CASI SPECIFICI IN ASSENZA DI BUFFER CON N O TEMPI DI INGRESSO INCREMENTALI
CICLO ELABORAZIONE DI SET DI MATRICI DI CASI SPECIFICI IN PRESENZA DI BUFFER CON DIMENSIONE DEL BUFFER INCREMENTALE
RITORNO A MENU PRECEDENTE
INSERIMENTO COMANDO OPERAZIONE RELATIVA AL SIMULATORE
INSERIMENTO COMANDO OPERAZIONE RELATIVA AL COMPARATORE
SALVATAGGIO E USCITA
RITORNO A MENU PRECEDENTE
RITORNO A MENU PRECEDENTE
TERMINARE IL PROGRAMMA
FINE PROGRAMMA


INIZIO
CALCOLO PRODUTTORIE E LORO MANTENIMENTO IN MEMORIA
MENU PRINCIPALE
SEZIONE ESECUZIONE MANUALE
SEZIONE CREAZIONE ED ESECUZIONE FILES DI SCHEDULAZIONE LAVORO
CARICARE ED ESEGUIRE FILE DI SCHEDULAZIONE LAVORO
CREAZIONE E SALVATAGGIO FILE DI SCHEDULAZIONE LAVORO
MENU INSERIMENTO COMANDI SCHEDULATORE
INSERIMENTO OPERAZIONE RELATIVE AL MODELLO MATEMATICO TEORICO
INSERIMENTO COMANDO OPERAZIONE RELATIVA AL SIMULATORE
SIMULAZIONE SINGOLA SENZA BUFFER
SIMULAZIONE SINGOLA IN PRESENZA DI BUFFER
BLOCCO DI SIMULAZIONI SINGOLE SENZA BUFFER
BLOCCO DI SIMULAZIONI MULTIPLE IN PRESENZA DI BUFFER
CICLO DI SIMULAZIONI SINGOLE IN ASSENZA DI BUFFER
CICLO DI SIMULAZIONI SINGOLE IN PRESENZA DI BUFFER
CICLO DI BLOCCHI DI SIMULAZIONI SINGOLE SENZA BUFFER
CICLO DI BLOCCHI DI SIMULAZIONI SINGOLE IN PRESENZA DI BUFFER
RITORNO A MENU PRECEDENTE
INSERIMENTO COMANDO OPERAZIONE RELATIVA AL COMPARATORE
COMPARAZIONE TRA SERIE DI CASI SPECIFICI SENZA BUFFER CON N INCREMENTALE
COMPARAZIONE TRA SERIE DI COMPARAZIONI IN PRESENZA DI BUFFER CON DIMENSIONE BUFFER INCREMENTALE
CICLO DI COMPARAZIONI DI CASI SPECIFICI IN ASSENZA DI BUFFER
CICLO DI COMPARAZIONI DI CASI SPECIFICI IN PRESENZA DI BUFFER
RITORNO A MENU PRECEDENTE
SALVATAGGIO E USCITA
RITORNO A MENU PRECEDENTE
TERMINARE IL PROGRAMMA
FINE PROGRAMMA

FLOW CHART SEMPLIFICATI DEI SIMULATORI

Tutti i simulatori si basano su una costante RANGE, modificabile solo dal programmatore, indicante la granularità con cui verrà riprodotta la distribuzione esponenziale.

Simulatore singolo in assenza di buffer

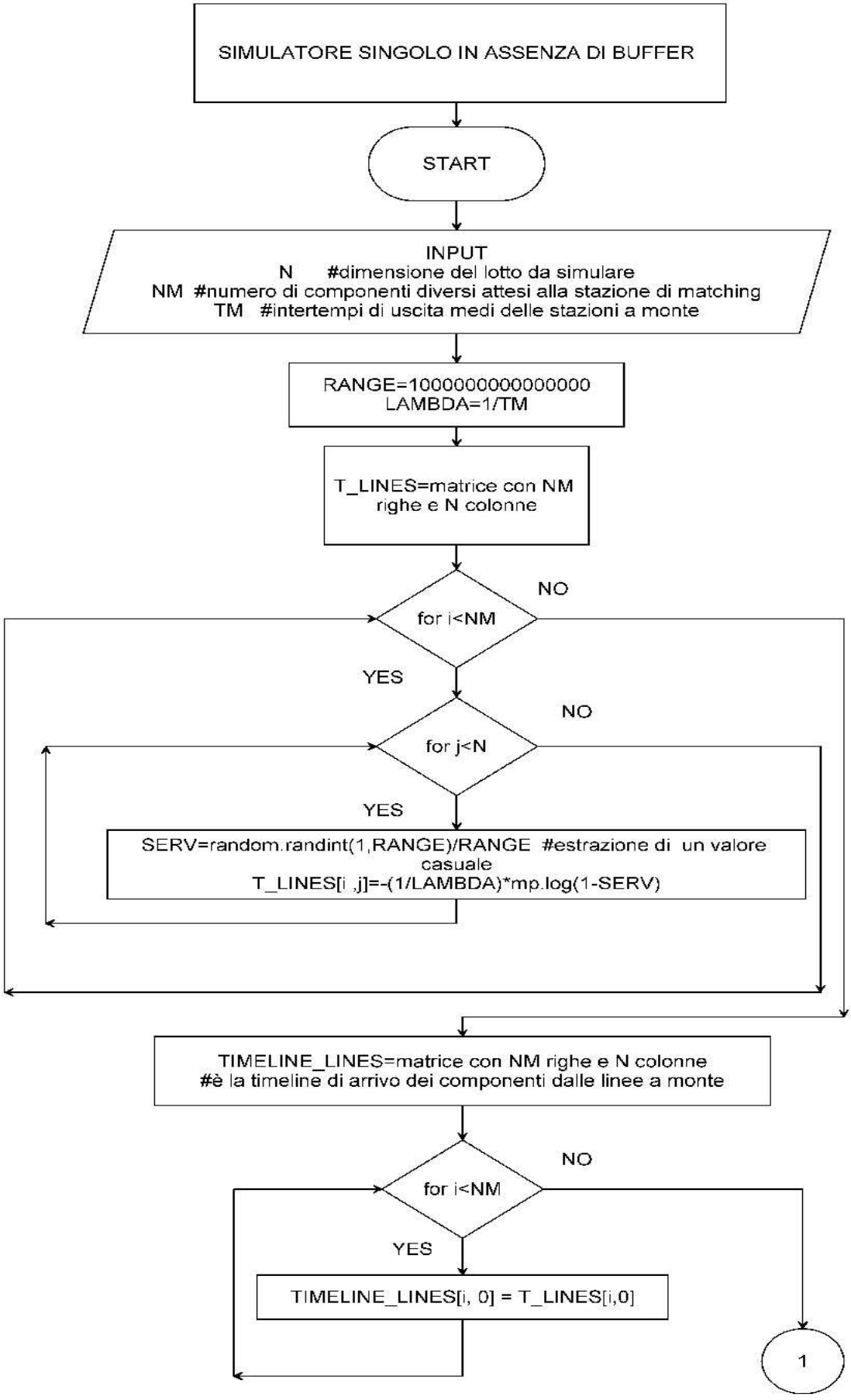
E' atto all'analisi del comportamento di una stazione di matching posta a valle di X linee, caratterizzate dalla medesima distribuzione esponenziale degli intertempi di uscita, producenti i componenti, tra loro complementari, che in questa verranno riuniti in pacchetti da destinarsi poi ad una successiva stazione di assemblaggio. Non vi è buffer al suo ingresso.

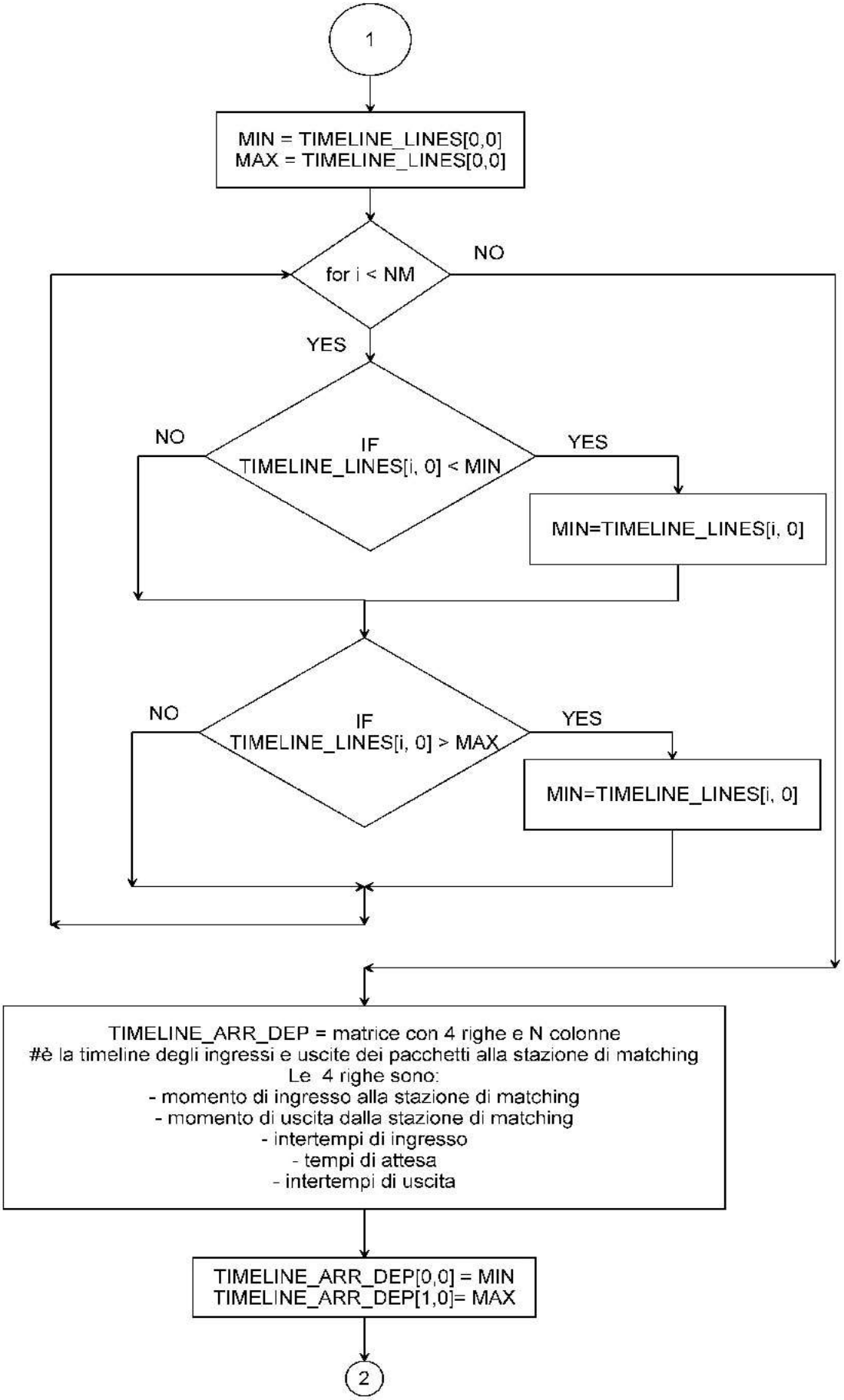
Il primo passo saranno le N * X estrazioni casuali dei tempi di interuscita dalle stazioni a monte.

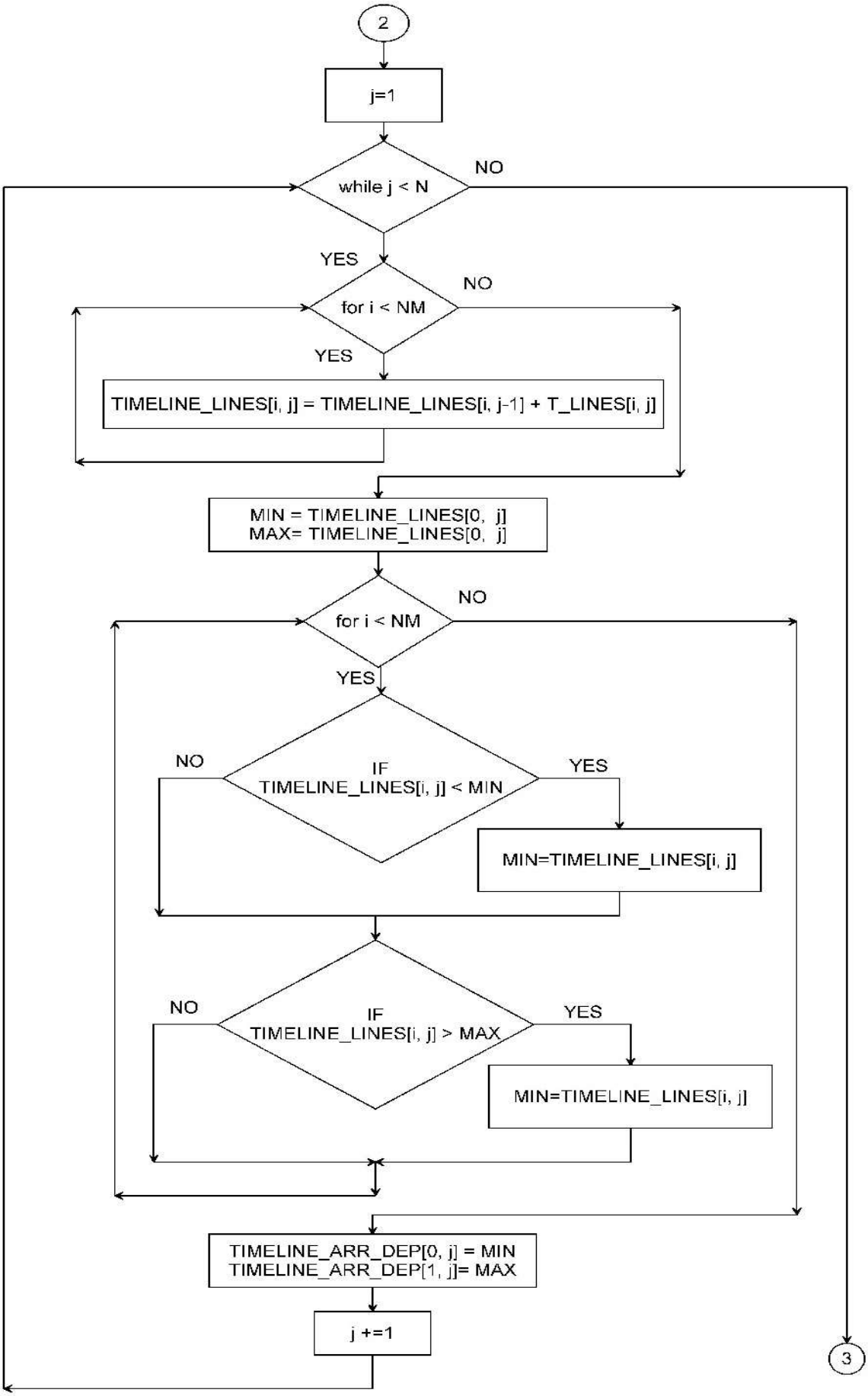
Successivamente il programma creerà X timelines, ognuna delle quali avente N istanti di arrivo alla stazione di matching.

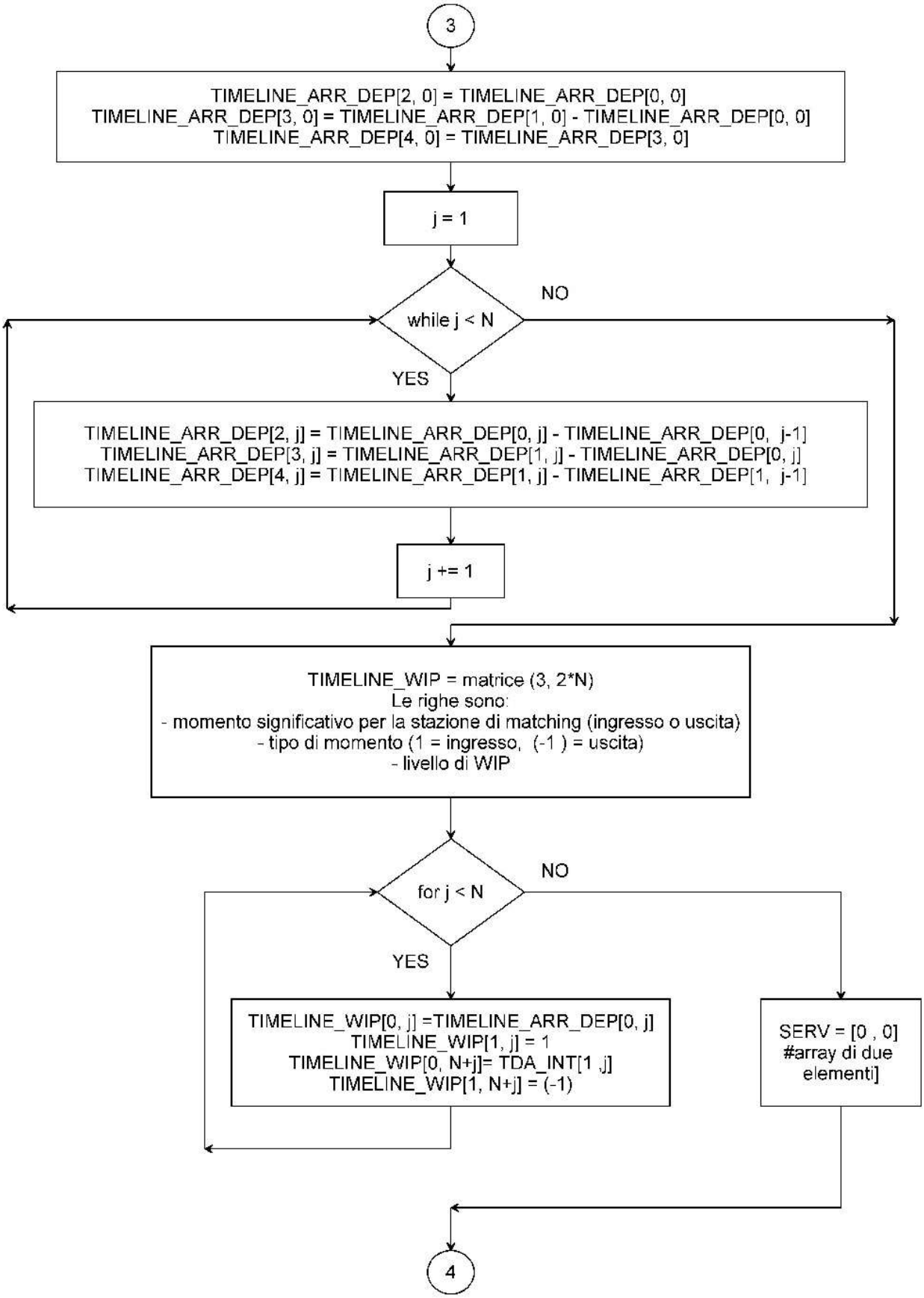
Da queste ne estrapolerà altre due, contenenti rispettivamente i momenti di arrivo alla stazione del primo e dell'ultimo componente di ogni pacchetto. Queste saranno la timeline rappresentanti gli ingressi e i rilasci dei pacchetti dalla stazione di matching. Tramite loro il programmma calcolerà gli intertempi di ingresso e di uscita ed i tempi di attesa, le loro medie e varianze. Per ottenere i livelli di WIP verrà creata una timeline specifica riassuntiva contenente, in un'unica sequenza, sia gli arrivi che i rilasci, caratterizzati rispettivamente da un «+1» e da un «-1», e dal livello di WIP raggiunto allo scoccare di ognuno di essi. Su questa timeline verrà calcolato il WIP medio.

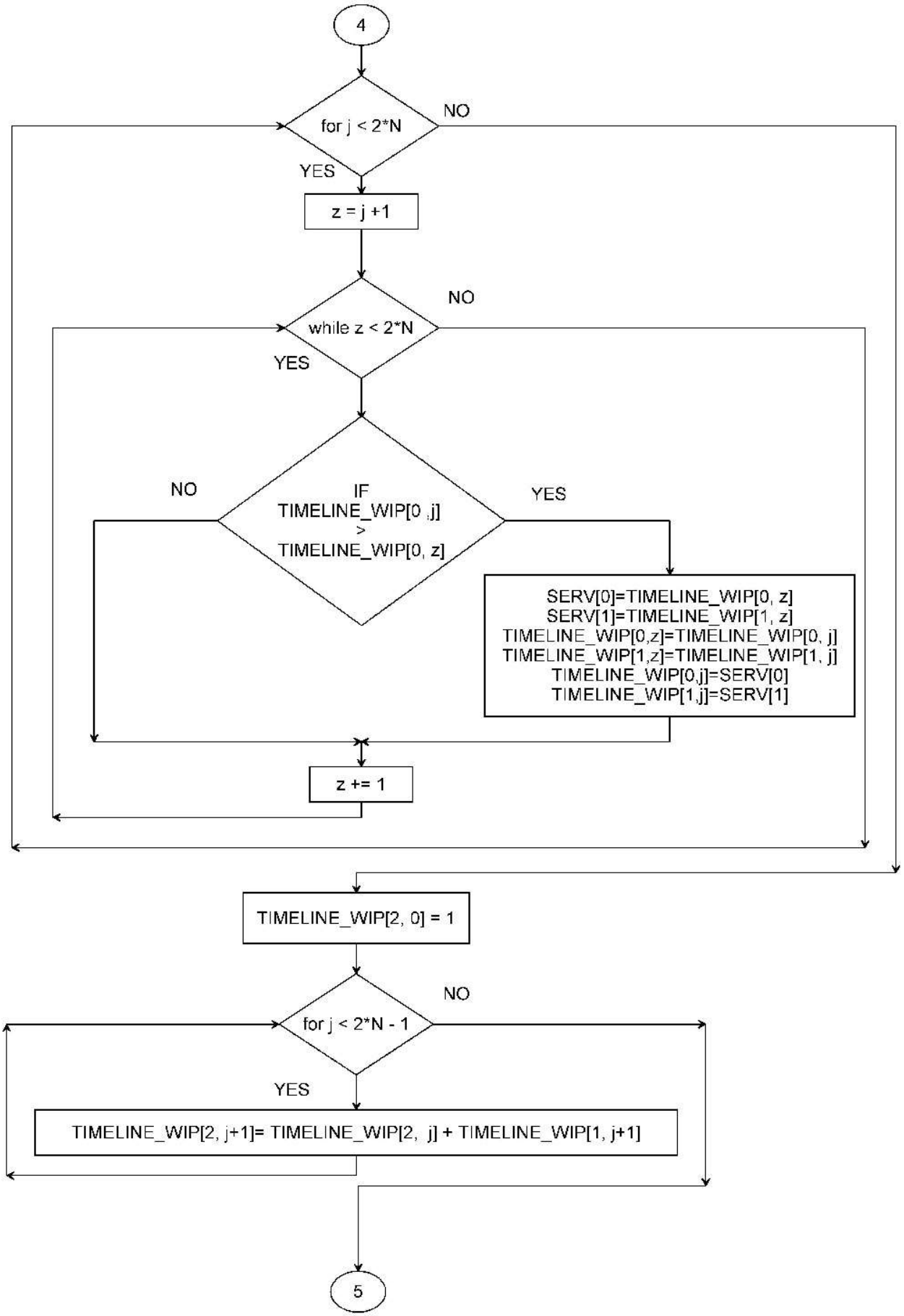
Ogni fase verrà eseguita sull'intero lotto.

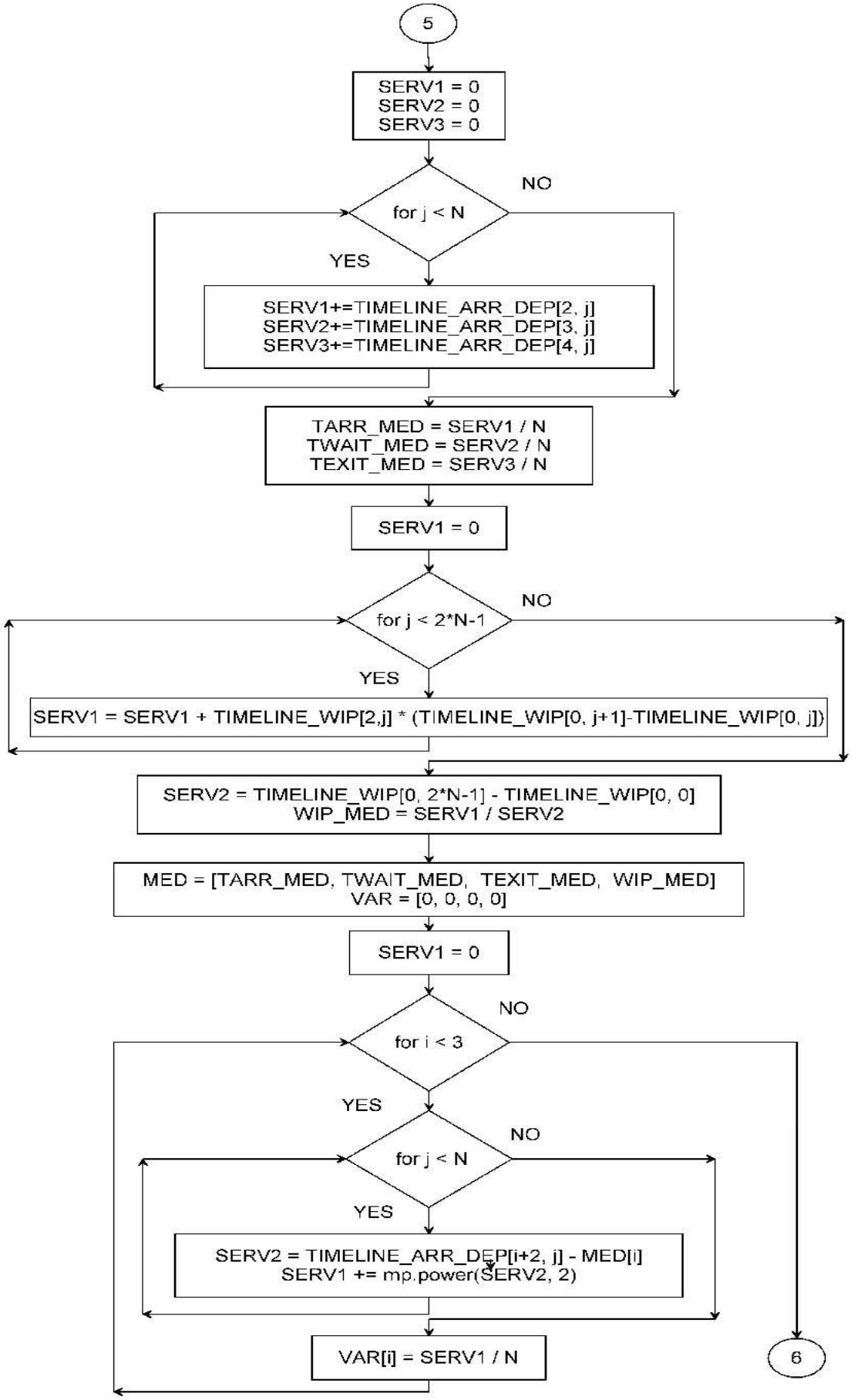

SIMULATORE SINGOLO IN ASSENZA DI BUFFER
START
INPUT
N #dimensione del lotto da simulare
NM #numero di componenti diversi attesi alla stazione di matching
TM #intertempi di uscita medi delle stazioni a monte
RANGE=1000000000000000
LAMBDA=1/TM
T_LINES=matrice con NM righe e N colonne
for i<NM
NO
YES
for j<N
NO
YES
SERV=random.randint(1,RANGE)/RANGE #estrazione di un valore casuale
T_LINES[i ,j]=-(1/LAMBDA)*mp.log(1-SERV)
TIMELINE_LINES=matrice con NM righe e N colonne
#è la timeline di arrivo dei componenti dalle linee a monte
for i<NM
NO
YES
TIMELINE_LINES[i, 0] = T_LINES[i,0]
1

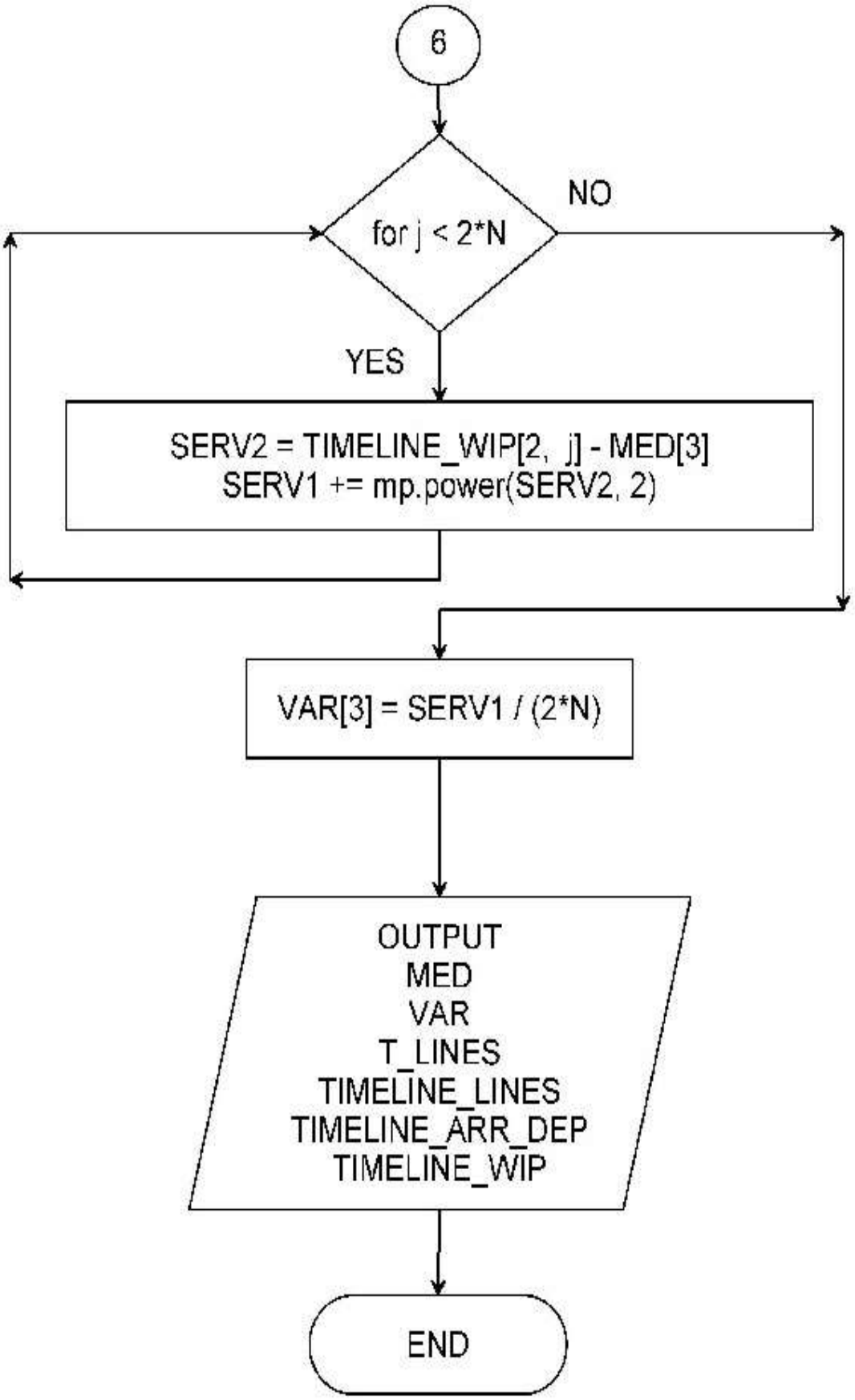

1
MIN = TIMELINE_LINES[0,0]
MAX = TIMELINE_LINES[0,0]
for i < NM
NO
YES
IF
TIMELINE_LINES[i, 0] < MIN
NO
YES
MIN=TIMELINE_LINES[i, 0]
IF
TIMELINE_LINES[i, 0] > MAX
NO
YES
MIN=TIMELINE_LINES[i, 0]
TIMELINE_ARR_DEP = matrice con 4 righe e N colonne
#è la timeline degli ingressi e uscite dei pacchetti alla stazione di matching
Le 4 righe sono:
- momento di ingresso alla stazione di matching
- momento di uscita dalla stazione di matching
- intertempi di ingresso
- tempi di attesa
- intertempi di uscita
TIMELINE_ARR_DEP[0,0] = MIN
TIMELINE_ARR_DEP[1,0]= MAX
2


2
j=1
while j < N
NO
YES
for i < NM
NO
YES
TIMELINE_LINES[i, j] = TIMELINE_LINES[i, j-1] + T_LINES[i, j]
MIN = TIMELINE_LINES[0, j]
MAX= TIMELINE_LINES[0, j]
for i < NM
NO
YES
IF
TIMELINE_LINES[i, j] < MIN
NO
YES
MIN=TIMELINE_LINES[i, j]
IF
TIMELINE_LINES[i, j] > MAX
NO
YES
MIN=TIMELINE_LINES[i, j]
TIMELINE_ARR_DEP[0, j] = MIN
TIMELINE_ARR_DEP[1, j]= MAX
j +=1
3


3
TIMELINE_ARR_DEP[2, 0] = TIMELINE_ARR_DEP[0, 0]
TIMELINE_ARR_DEP[3, 0] = TIMELINE_ARR_DEP[1, 0] - TIMELINE_ARR_DEP[0, 0]
TIMELINE_ARR_DEP[4, 0] = TIMELINE_ARR_DEP[3, 0]
j = 1
while j < N
NO
YES
TIMELINE_ARR_DEP[2, j] = TIMELINE_ARR_DEP[0, j] - TIMELINE_ARR_DEP[0, j-1]
TIMELINE_ARR_DEP[3, j] = TIMELINE_ARR_DEP[1, j] - TIMELINE_ARR_DEP[0, j]
TIMELINE_ARR_DEP[4, j] = TIMELINE_ARR_DEP[1, j] - TIMELINE_ARR_DEP[1, j-1]
j += 1
TIMELINE_WIP = matrice (3, 2*N)
Le righe sono:
- momento significativo per la stazione di matching (ingresso o uscita)
- tipo di momento (1 = ingresso, (-1) = uscita)
- livello di WIP
for j < N
NO
YES
TIMELINE_WIP[0, j] =TIMELINE_ARR_DEP[0, j]
TIMELINE_WIP[1, j] = 1
TIMELINE_WIP[0, N+j]= TDA_INT[1 ,j]
TIMELINE_WIP[1, N+j] = (-1)
SERV = [0 , 0]
#array di due
elementi]
4


4
for j < 2*N
NO
YES
z = j +1
while z < 2*N
NO
YES
IF
TIMELINE_WIP[0 ,j]
>
TIMELINE_WIP[0, z]
NO
YES
SERV[0]=TIMELINE_WIP[0, z]
SERV[1]=TIMELINE_WIP[1, z]
TIMELINE_WIP[0,z]=TIMELINE_WIP[0, j]
TIMELINE_WIP[1,z]=TIMELINE_WIP[1, j]
TIMELINE_WIP[0,j]=SERV[0]
TIMELINE_WIP[1,j]=SERV[1]
z += 1
TIMELINE_WIP[2, 0] = 1
for j < 2*N - 1
NO
YES
TIMELINE_WIP[2, j+1]= TIMELINE_WIP[2, j] + TIMELINE_WIP[1, j+1]
5

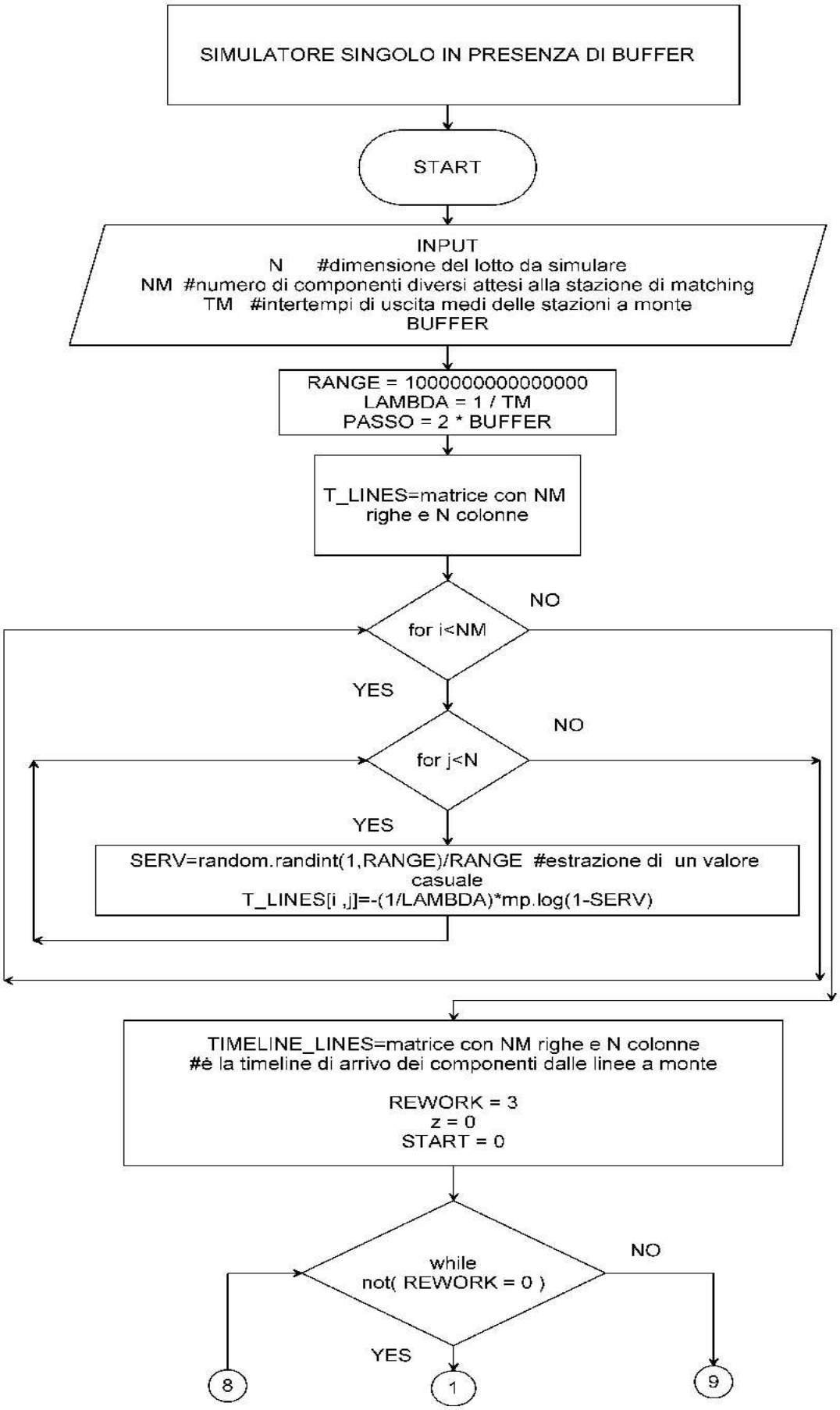

5
SERV1 = 0
SERV2 = 0
SERV3 = 0
for j < N
NO
YES
SERV1+=TIMELINE_ARR_DEP[2, j]
SERV2+=TIMELINE_ARR_DEP[3, j]
SERV3+=TIMELINE_ARR_DEP[4, j]
TARR_MED = SERV1 / N
TWAIT_MED = SERV2 / N
TEXIT_MED = SERV3 / N
SERV1 = 0
for j < 2*N-1
NO
YES
SERV1 = SERV1 + TIMELINE_WIP[2,j] * (TIMELINE_WIP[0, j+1]-TIMELINE_WIP[0, j])
SERV2 = TIMELINE_WIP[0, 2*N-1] - TIMELINE_WIP[0, 0]
WIP_MED = SERV1 / SERV2
MED = [TARR_MED, TWAIT_MED, TEXIT_MED, WIP_MED]
VAR = [0, 0, 0, 0]
SERV1 = 0
for i < 3
NO
YES
for j < N
NO
YES
SERV2 = TIMELINE_ARR_DEP[i+2, j] - MED[i]
SERV1 += mp.power(SERV2, 2)
VAR[i] = SERV1 / N
6

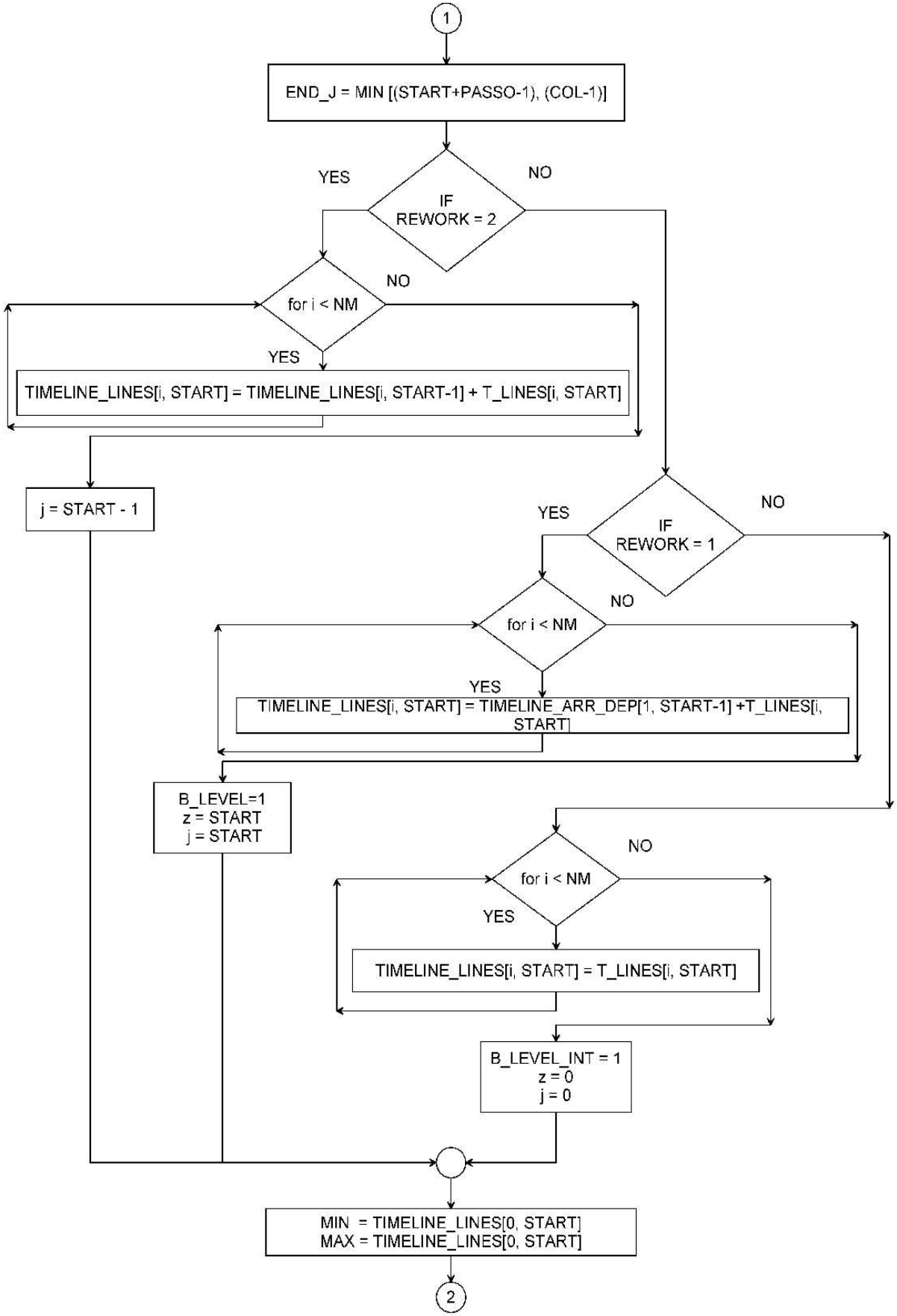

6
for j < 2*N
NO
YES
SERV2 = TIMELINE_WIP[2, j] - MED[3]
SERV1 += mp.power(SERV2, 2)
VAR[3] = SERV1 / (2*N)
OUTPUT
MED
VAR
T_LINES
TIMELINE_LINES
TIMELINE_ARR_DEP
TIMELINE_WIP
END

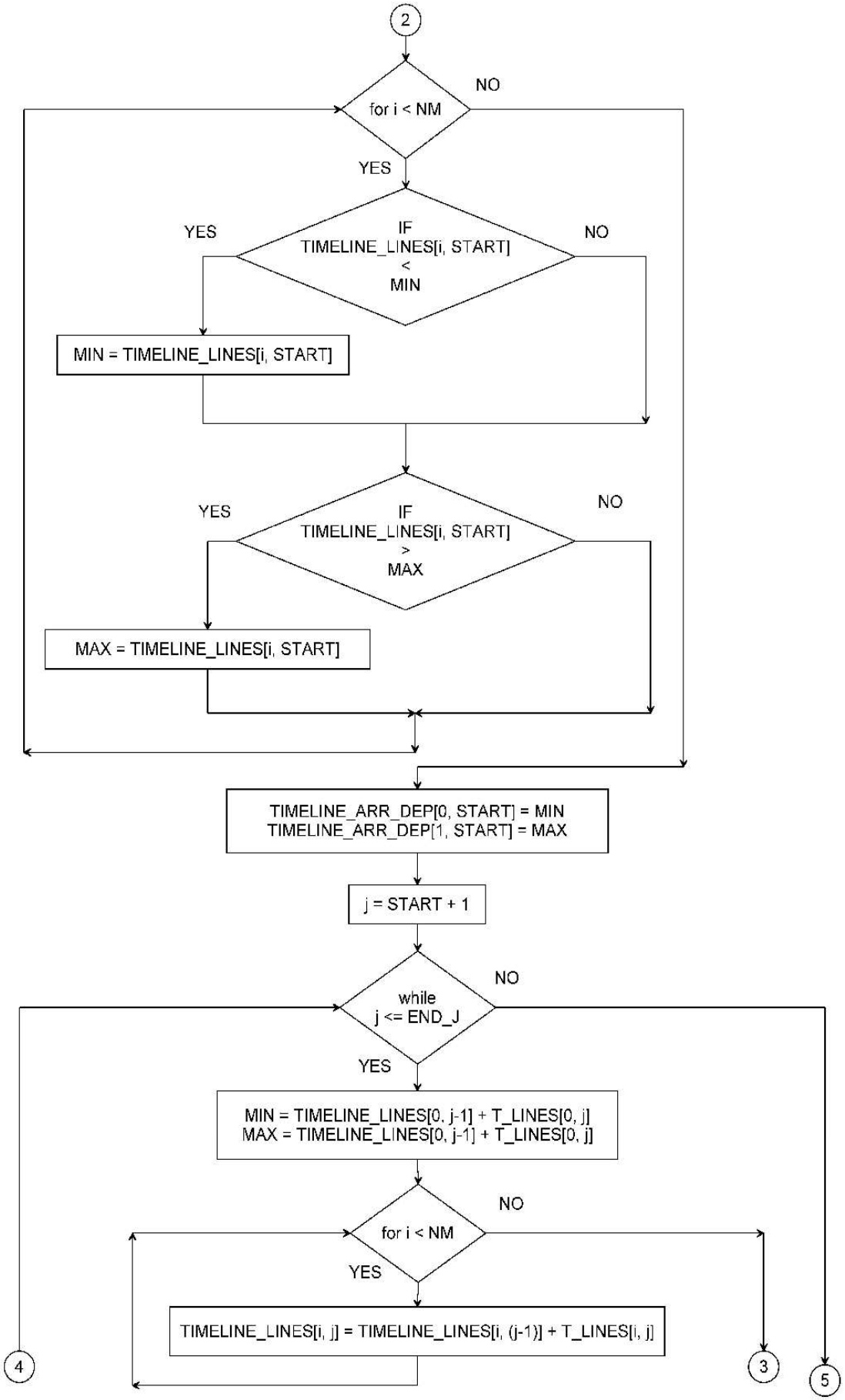
Simulatore singolo in presenza di buffer

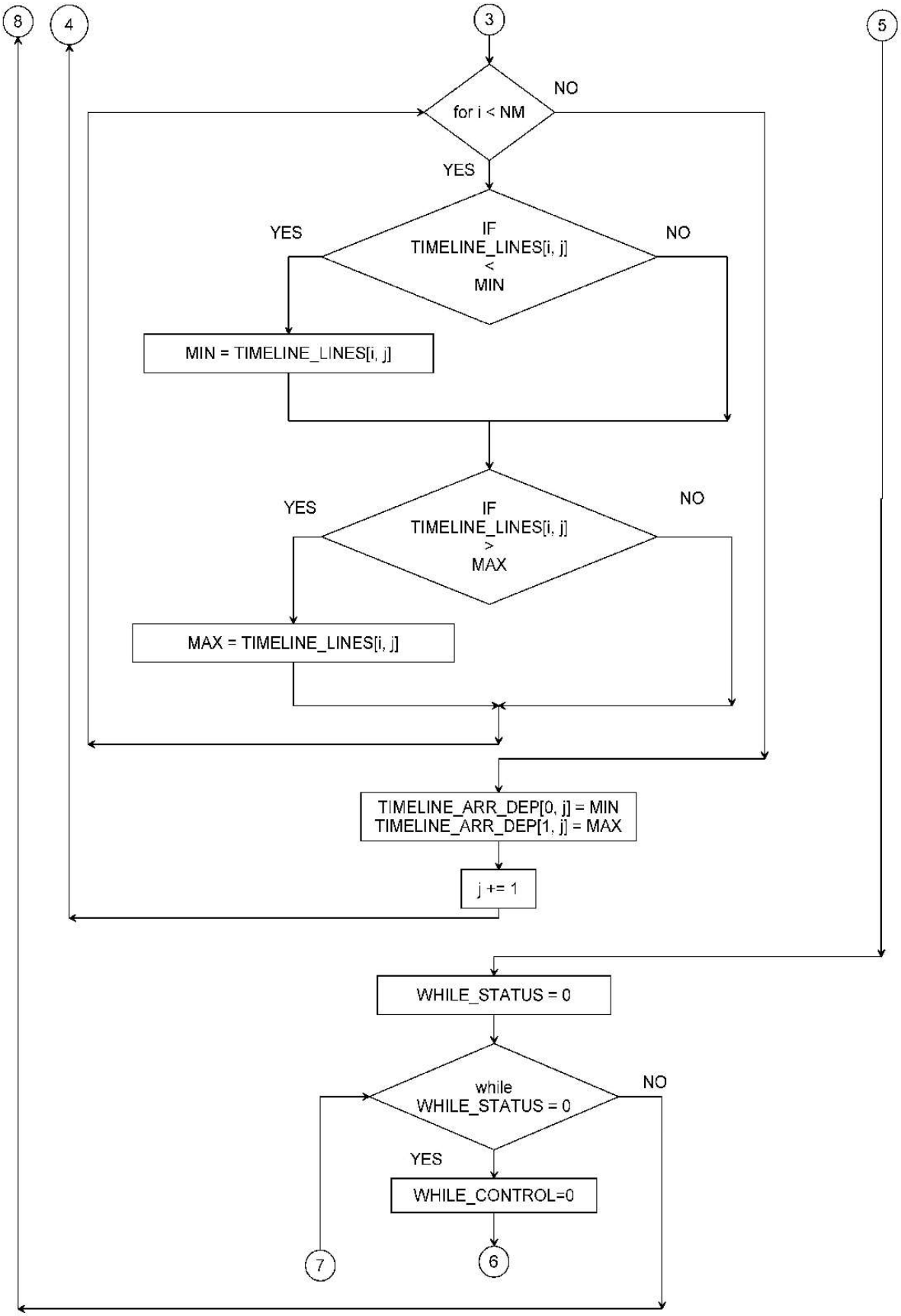
I casi simulabili con questo strumento sono gli stessi del precedente con la differenza della presenza di un buffer a limitare il numero di pacchetti in coda alla stazione di matching.

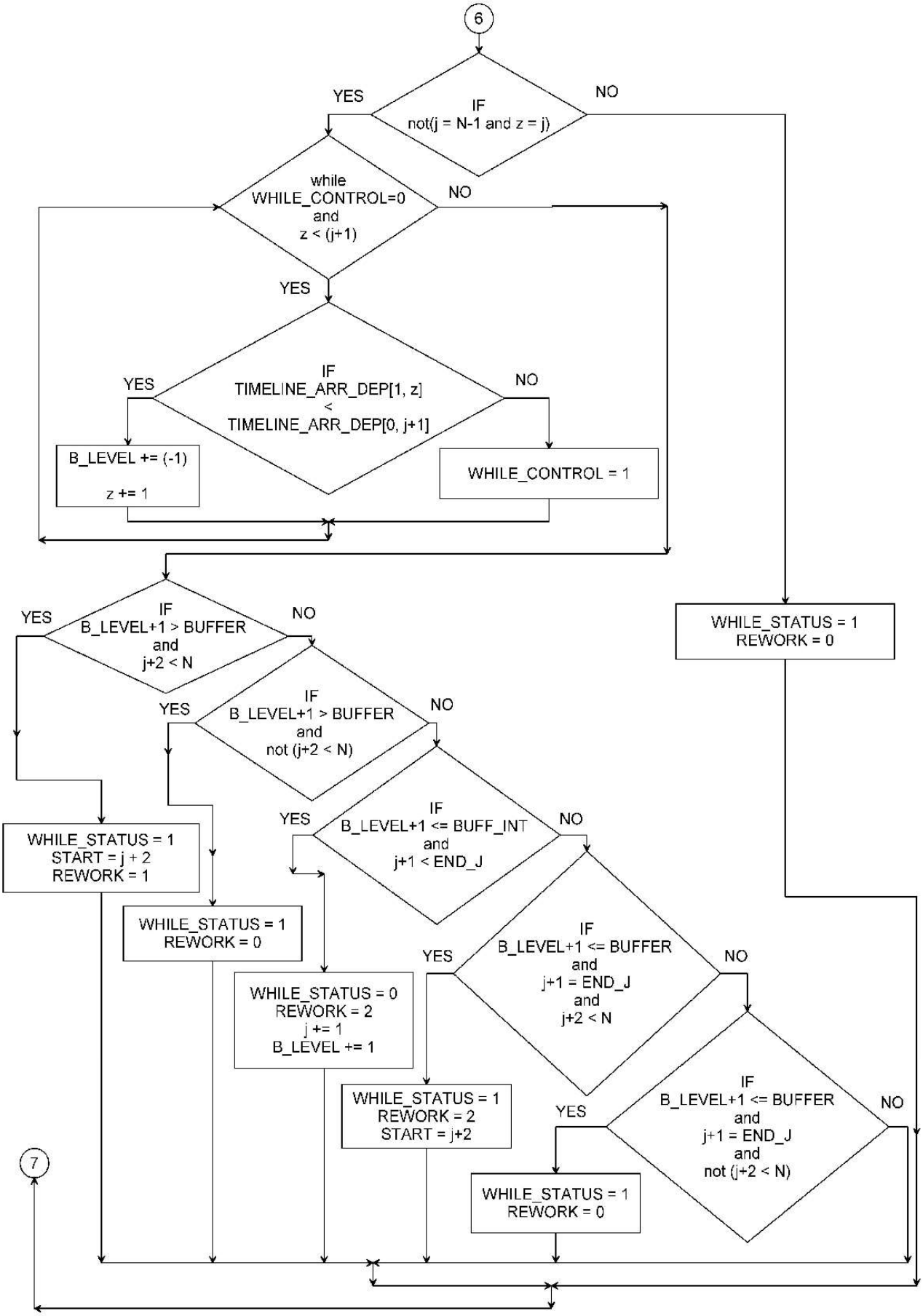
Come il precedente eseguirà le estrazioni degli N*X tempi delle stazione a monte in un unica operazione all'inizio dell'elaborazione. La costruzione delle timelines avverrà invece tramite un ciclo che elaborerà solo Y pacchetti alla volta. Il contatore del ciclo verrà resettato solo in caso di saturazione del buffer o, in caso contrario, di esaurimento del contatore stesso. I punti di partenza da cui riprenderà la creazione delle timelines varrieranno a seconda della condizione scaturente.

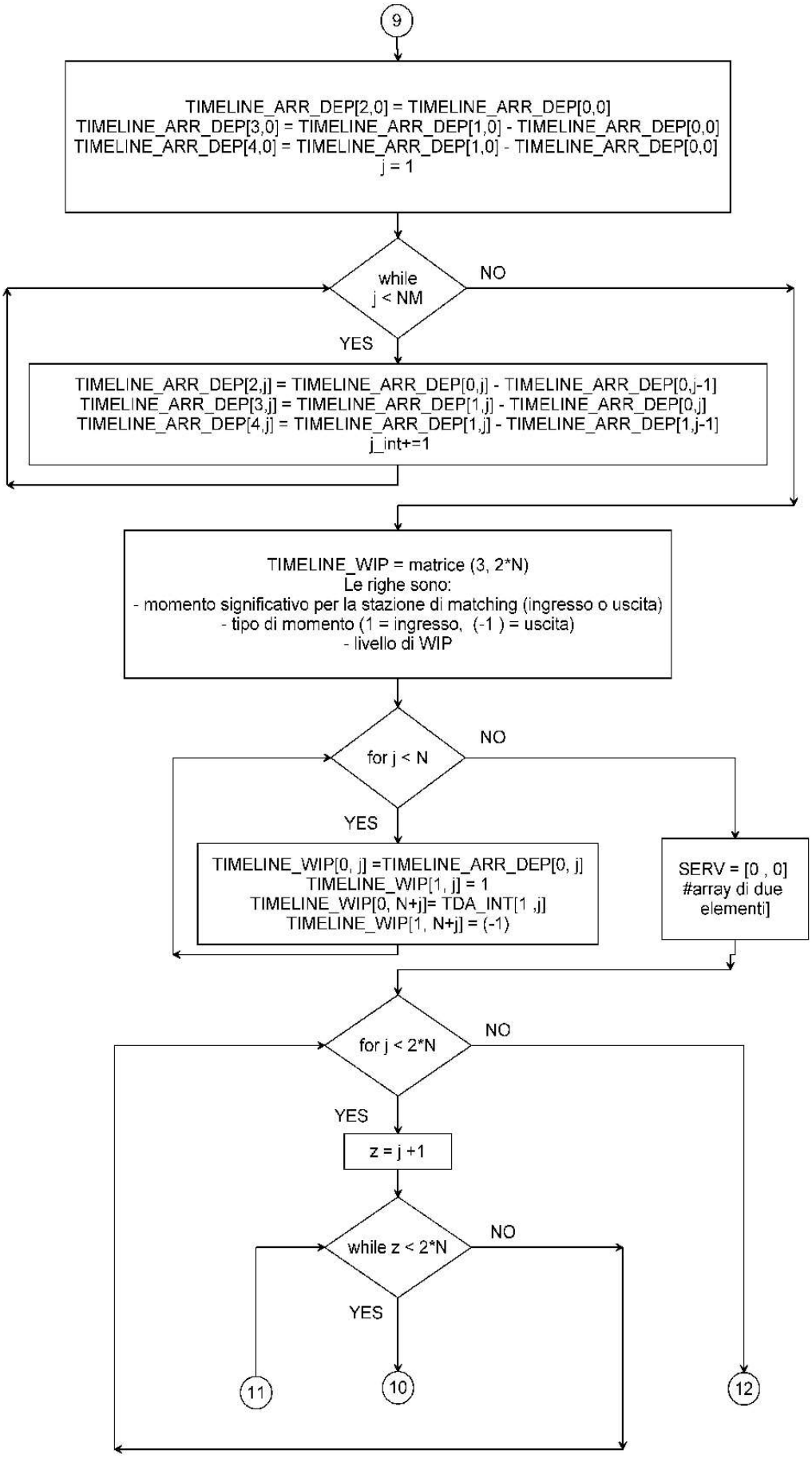

SIMULATORE SINGOLO IN PRESENZA DI BUFFER
START
INPUT
N #dimensione del lotto da simulare
NM #numero di componenti diversi attesi alla stazione di matching
TM #intertempi di uscita medi delle stazioni a monte
BUFFER
RANGE = 1000000000000000
LAMBDA = 1 / TM
PASSO = 2 * BUFFER
T_LINES=matrice con NM righe e N colonne
for i<NM
NO
YES
for j<N
NO
YES
SERV=random.randint(1,RANGE)/RANGE #estrazione di un valore casuale
T_LINES[i ,j]=-(1/LAMBDA)*mp.log(1-SERV)
TIMELINE_LINES=matrice con NM righe e N colonne
#è la timeline di arrivo dei componenti dalle linee a monte
REWORK = 3
z = 0
START = 0
while not(REWORK = 0)
NO
YES
8
1
9

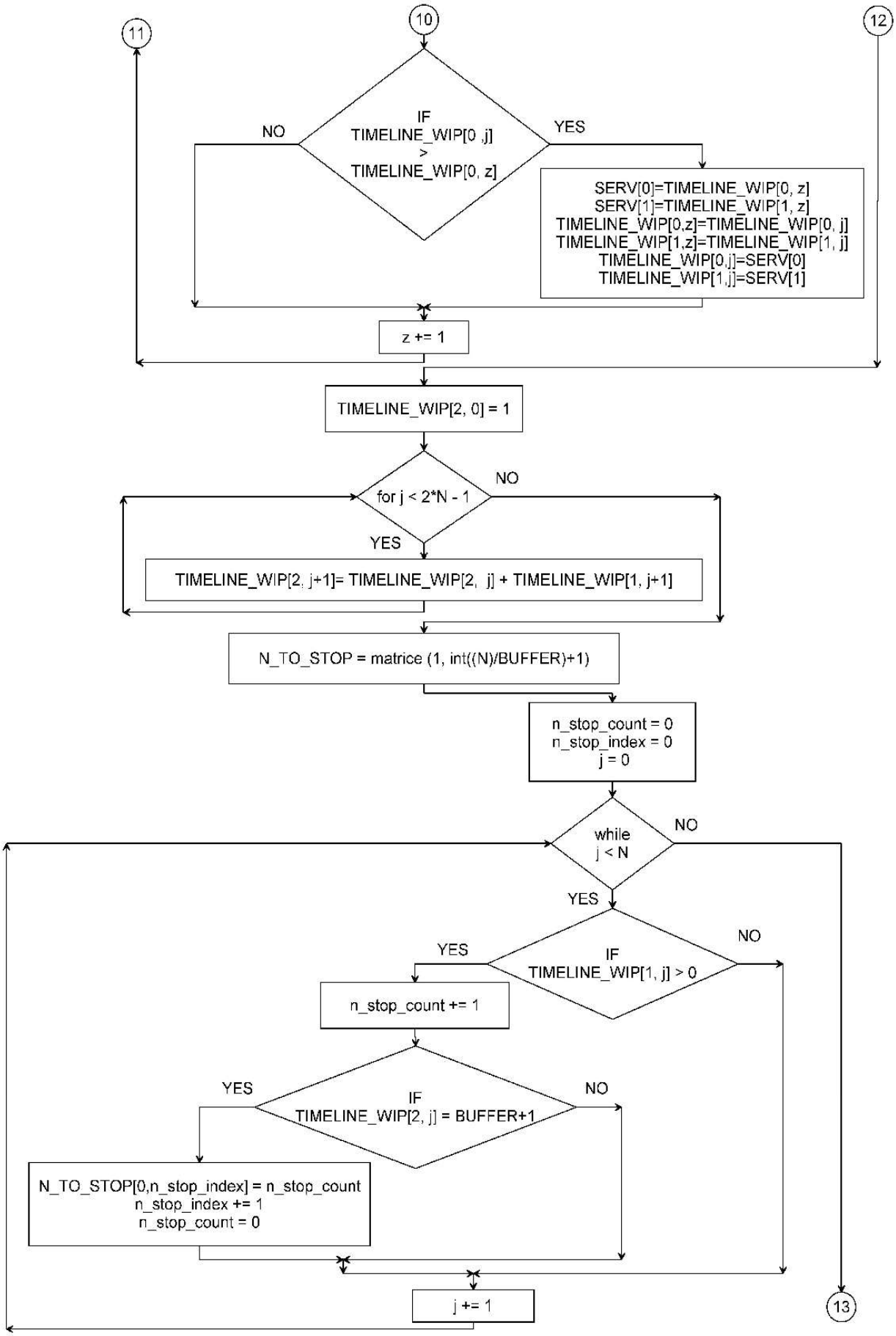

1
END_J = MIN [(START+PASSO-1), (COL-1)]
YES
IF
REWORK = 2
NO
NO
for i < NM
YES
TIMELINE_LINES[i, START] = TIMELINE_LINES[i, START-1] + T_LINES[i, START]
j = START - 1
YES
IF
REWORK = 1
NO
NO
for i < NM
YES
TIMELINE_LINES[i, START] = TIMELINE_ARR_DEP[1, START-1] +T_LINES[i, START]
B_LEVEL=1
z = START
j = START
NO
for i < NM
YES
TIMELINE_LINES[i, START] = T_LINES[i, START]
B_LEVEL_INT = 1
z = 0
j = 0
MIN = TIMELINE_LINES[0, START]
MAX = TIMELINE_LINES[0, START]
2

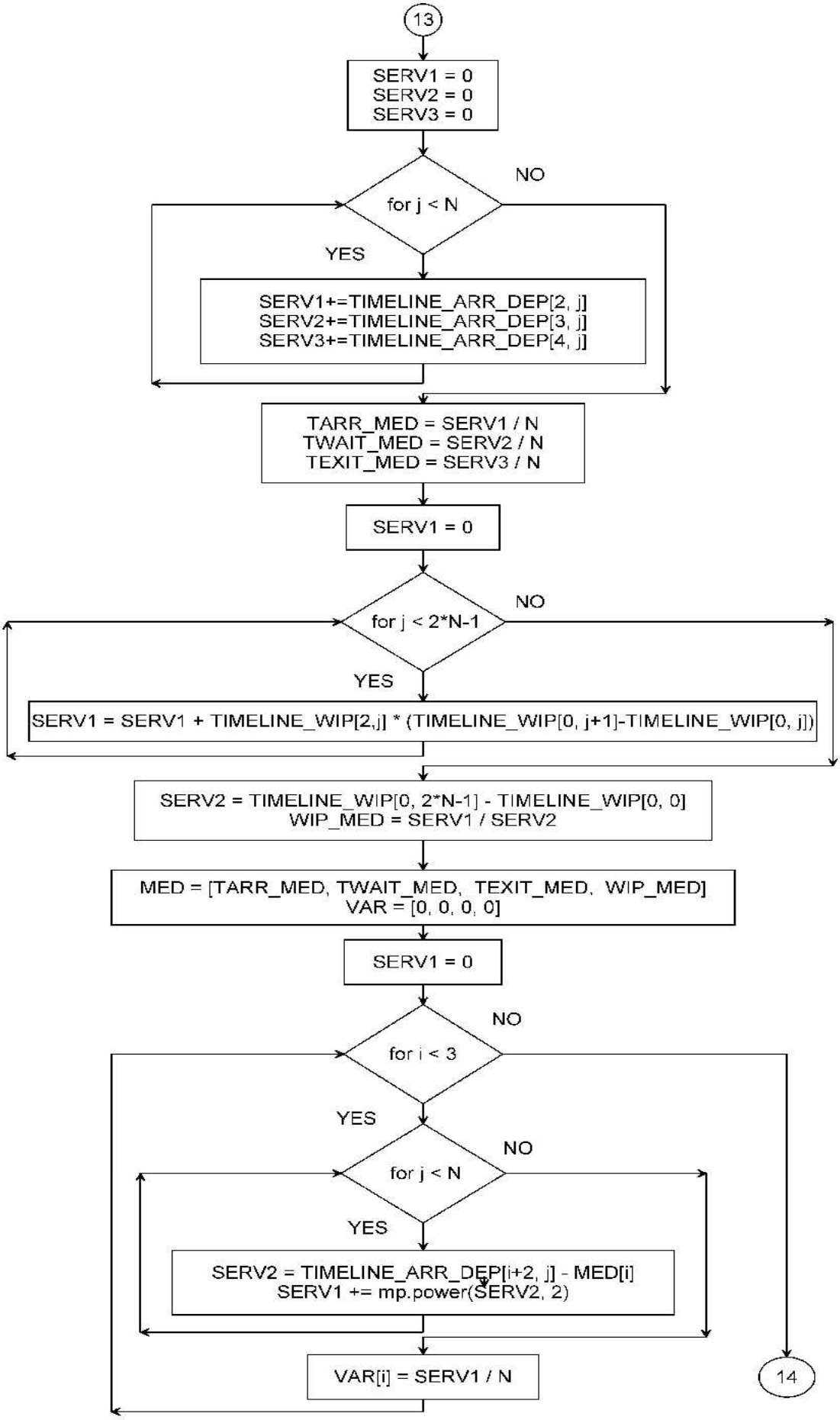

2
for i < NM
NO
YES
IF
TIMELINE_LINES[i, START]
<
MIN
YES
NO
MIN = TIMELINE_LINES[i, START]
IF
TIMELINE_LINES[i, START]
>
MAX
YES
NO
MAX = TIMELINE_LINES[i, START]
TIMELINE_ARR_DEP[0, START] = MIN
TIMELINE_ARR_DEP[1, START] = MAX
j = START + 1
while
j <= END_J
NO
YES
MIN = TIMELINE_LINES[0, j-1] + T_LINES[0, j]
MAX = TIMELINE_LINES[0, j-1] + T_LINES[0, j]
for i < NM
NO
YES
TIMELINE_LINES[i, j] = TIMELINE_LINES[i, (j-1)] + T_LINES[i, j]
4
3
5

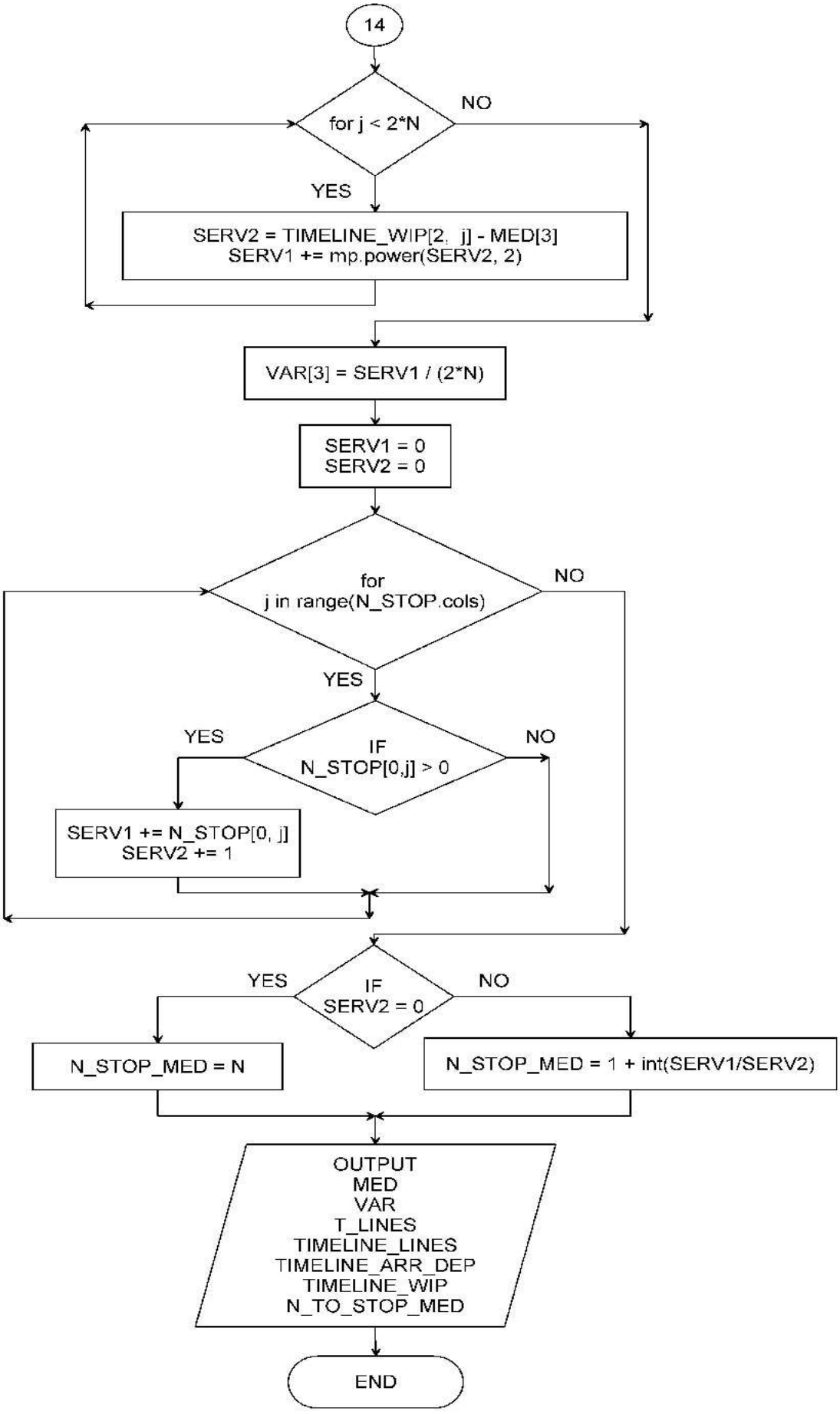

8
4
3
5
for i < NM
NO
YES
IF
TIMELINE_LINES[i, j]
<
MIN
YES
NO
MIN = TIMELINE_LINES[i, j]
IF
TIMELINE_LINES[i, j]
>
MAX
YES
NO
MAX = TIMELINE_LINES[i, j]
TIMELINE_ARR_DEP[0, j] = MIN
TIMELINE_ARR_DEP[1, j] = MAX
j += 1
WHILE_STATUS = 0
while
WHILE_STATUS = 0
NO
YES
WHILE_CONTROL=0
7
6


6
IF
not(j = N-1 and z = j)
YES
NO
while
WHILE_CONTROL=0
and
z < (j+1)
NO
YES
IF
TIMELINE_ARR_DEP[1, z]
<
TIMELINE_ARR_DEP[0, j+1]
YES
NO
B_LEVEL += (-1)
z += 1
WHILE_CONTROL = 1
WHILE_STATUS = 1
REWORK = 0
IF
B_LEVEL+1 > BUFFER
and
j+2 < N
YES
NO
IF
B_LEVEL+1 > BUFFER
and
not (j+2 < N)
YES
NO
IF
B_LEVEL+1 <= BUFF_INT
and
j+1 < END_J
YES
NO
WHILE_STATUS = 1
START = j + 2
REWORK = 1
WHILE_STATUS = 1
REWORK = 0
WHILE_STATUS = 0
REWORK = 2
j += 1
B_LEVEL += 1
IF
B_LEVEL+1 <= BUFFER
and
j+1 = END_J
and
j+2 < N
YES
NO
WHILE_STATUS = 1
REWORK = 2
START = j+2
IF
B_LEVEL+1 <= BUFFER
and
j+1 = END_J
and
not (j+2 < N)
YES
NO
WHILE_STATUS = 1
REWORK = 0
7


9
TIMELINE_ARR_DEP[2,0] = TIMELINE_ARR_DEP[0,0]
TIMELINE_ARR_DEP[3,0] = TIMELINE_ARR_DEP[1,0] - TIMELINE_ARR_DEP[0,0]
TIMELINE_ARR_DEP[4,0] = TIMELINE_ARR_DEP[1,0] - TIMELINE_ARR_DEP[0,0]
j = 1
while
j < NM
NO
YES
TIMELINE_ARR_DEP[2,j] = TIMELINE_ARR_DEP[0,j] - TIMELINE_ARR_DEP[0,j-1]
TIMELINE_ARR_DEP[3,j] = TIMELINE_ARR_DEP[1,j] - TIMELINE_ARR_DEP[0,j]
TIMELINE_ARR_DEP[4,j] = TIMELINE_ARR_DEP[1,j] - TIMELINE_ARR_DEP[1,j-1]
j_int+=1
TIMELINE_WIP = matrice (3, 2*N)
Le righe sono:
- momento significativo per la stazione di matching (ingresso o uscita)
- tipo di momento (1 = ingresso, (-1) = uscita)
- livello di WIP
for j < N
NO
YES
TIMELINE_WIP[0, j] =TIMELINE_ARR_DEP[0, j]
TIMELINE_WIP[1, j] = 1
TIMELINE_WIP[0, N+j]= TDA_INT[1 ,j]
TIMELINE_WIP[1, N+j] = (-1)
SERV = [0 , 0]
#array di due
elementi]
for j < 2*N
NO
YES
z = j +1
while z < 2*N
NO
YES
11
10
12


10
11
12
IF
TIMELINE_WIP[0 ,j]
>
TIMELINE_WIP[0, z]
NO
YES
SERV[0]=TIMELINE_WIP[0, z]
SERV[1]=TIMELINE_WIP[1, z]
TIMELINE_WIP[0,z]=TIMELINE_WIP[0, j]
TIMELINE_WIP[1,z]=TIMELINE_WIP[1, j]
TIMELINE_WIP[0,j]=SERV[0]
TIMELINE_WIP[1,j]=SERV[1]
z += 1
TIMELINE_WIP[2, 0] = 1
for j < 2*N - 1
NO
YES
TIMELINE_WIP[2, j+1]= TIMELINE_WIP[2, j] + TIMELINE_WIP[1, j+1]
N_TO_STOP = matrice (1, int((N)/BUFFER)+1)
n_stop_count = 0
n_stop_index = 0
j = 0
while
j < N
NO
YES
IF
TIMELINE_WIP[1, j] > 0
YES
NO
n_stop_count += 1
IF
TIMELINE_WIP[2, j] = BUFFER+1
YES
NO
N_TO_STOP[0,n_stop_index] = n_stop_count
n_stop_index += 1
n_stop_count = 0
j += 1
13

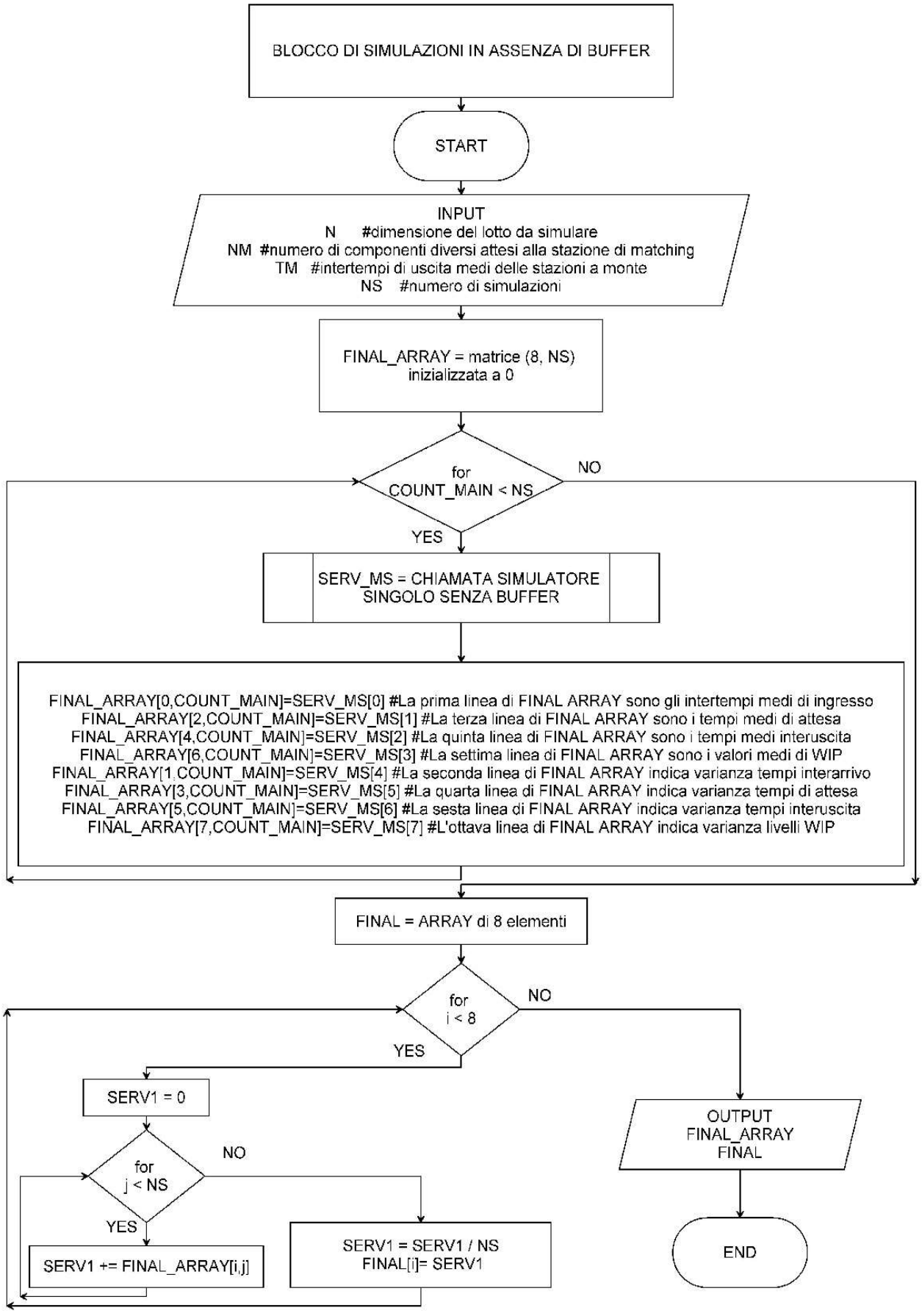

13
SERV1 = 0
SERV2 = 0
SERV3 = 0
for j < N
NO
YES
SERV1+=TIMELINE_ARR_DEP[2, j]
SERV2+=TIMELINE_ARR_DEP[3, j]
SERV3+=TIMELINE_ARR_DEP[4, j]
TARR_MED = SERV1 / N
TWAIT_MED = SERV2 / N
TEXIT_MED = SERV3 / N
SERV1 = 0
for j < 2*N-1
NO
YES
SERV1 = SERV1 + TIMELINE_WIP[2,j] * (TIMELINE_WIP[0, j+1]-TIMELINE_WIP[0, j])
SERV2 = TIMELINE_WIP[0, 2*N-1] - TIMELINE_WIP[0, 0]
WIP_MED = SERV1 / SERV2
MED = [TARR_MED, TWAIT_MED, TEXIT_MED, WIP_MED]
VAR = [0, 0, 0, 0]
SERV1 = 0
for i < 3
NO
YES
for j < N
NO
YES
SERV2 = TIMELINE_ARR_DEP[i+2, j] - MED[i]
SERV1 += mp.power(SERV2, 2)
VAR[i] = SERV1 / N
14

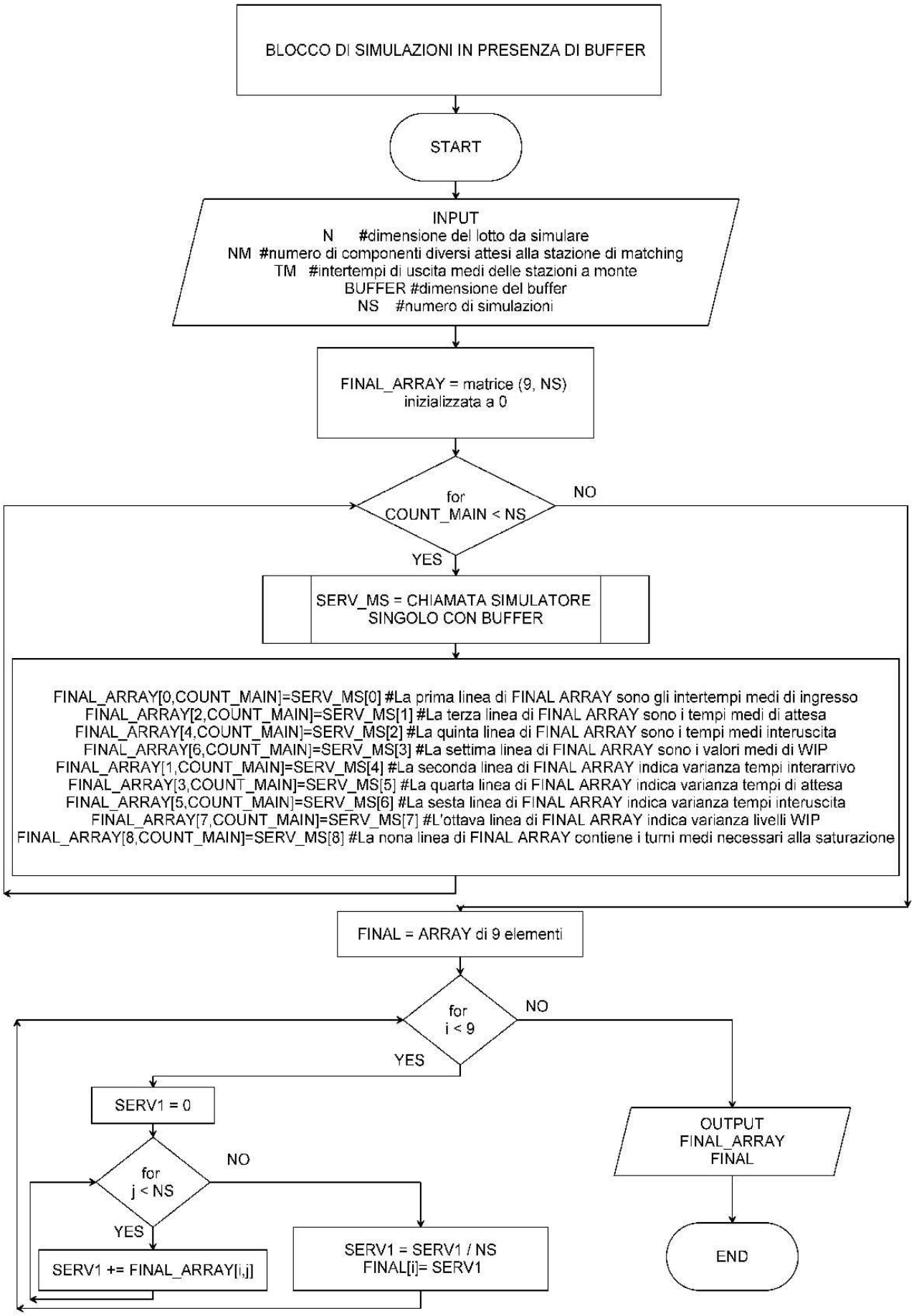

14
for j < 2*N
NO
YES
SERV2 = TIMELINE_WIP[2, j] - MED[3]
SERV1 += mp.power(SERV2, 2)
VAR[3] = SERV1 / (2*N)
SERV1 = 0
SERV2 = 0
for
j in range(N_STOP.cols)
NO
YES
YES
IF
N_STOP[0,j] > 0
NO
SERV1 += N_STOP[0, j]
SERV2 += 1
YES
IF
SERV2 = 0
NO
N_STOP_MED = N
N_STOP_MED = 1 + int(SERV1/SERV2)
OUTPUT
MED
VAR
T_LINES
TIMELINE_LINES
TIMELINE_ARR_DEP
TIMELINE_WIP
N_TO_STOP_MED
END

Simulazioni a blocchi

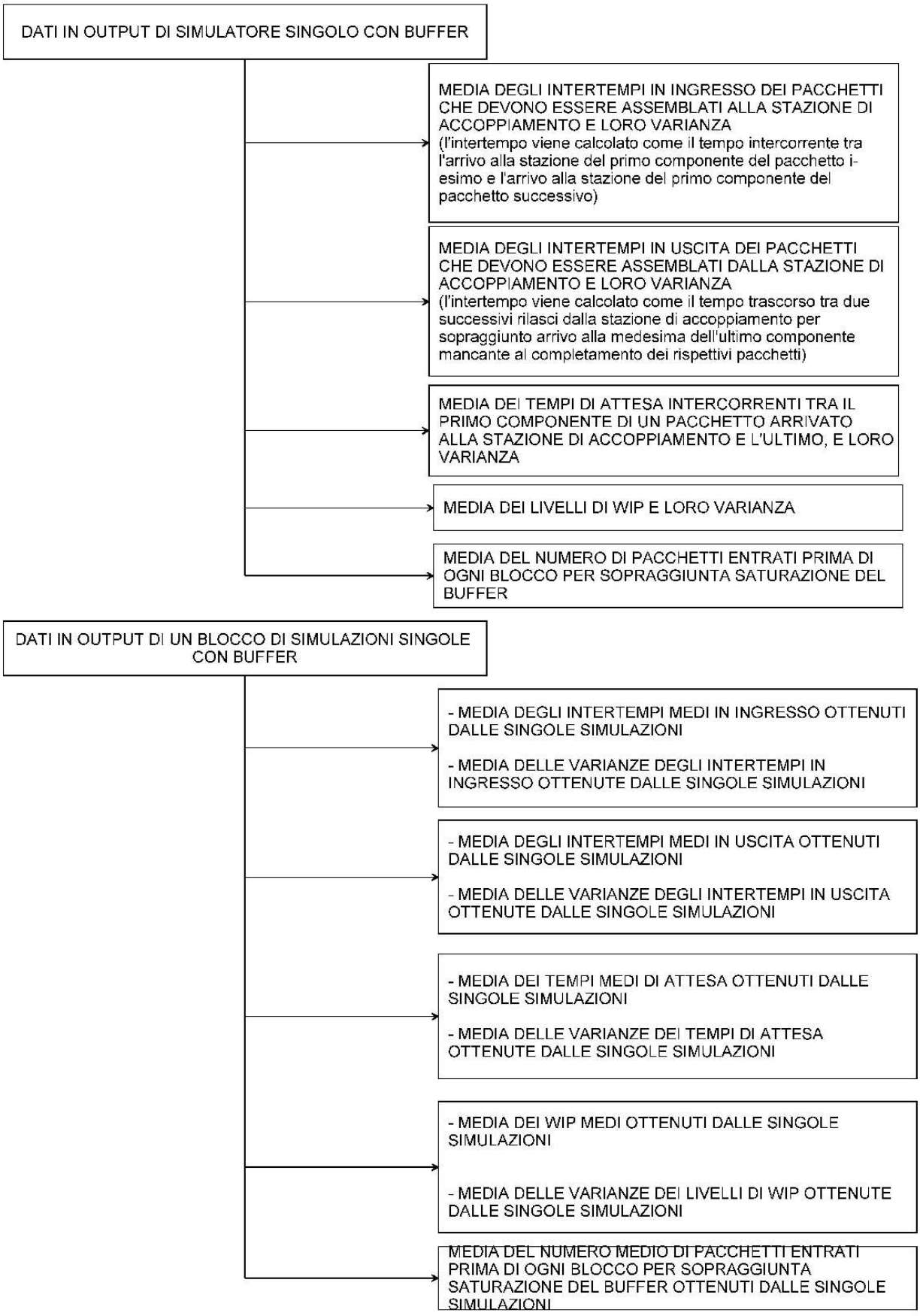
Scopo del blocco di simulazioni, con o senza buffer, è quello di produrre delle statistiche attendibili tramite il calcolo delle medie dei valori medi prodotti dalle singole simuzioni. Non vi sono particolari differenze logiche tra la versione in assenza ed in presenza di buffer.

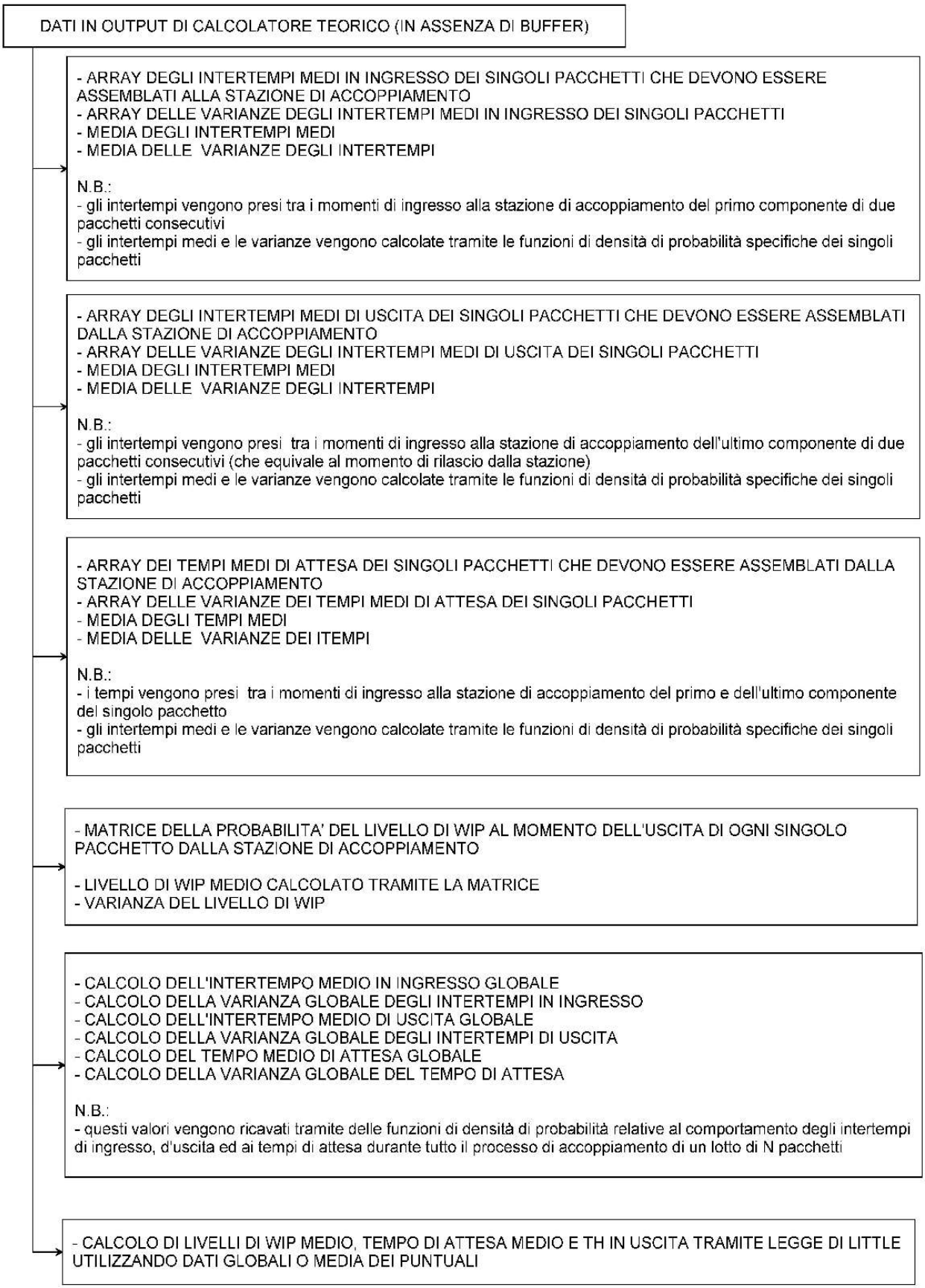
Entrambe eseguiranno richiameranno per il numero di volte richiesto (dall'utente, dallo schedulatore o dal comparatore) la simulazione singola a loro coerente, raccogliendone i dati ed eseguendo una media finale.

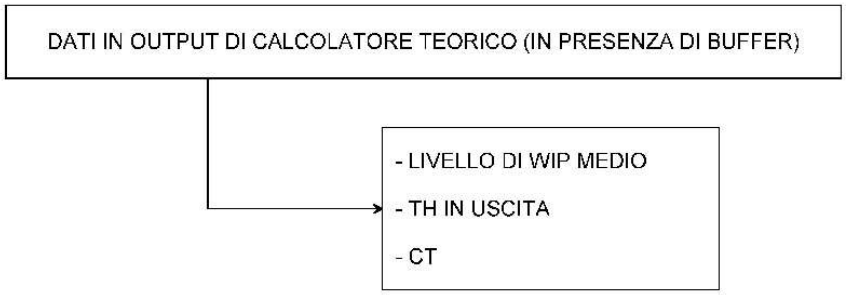

BLOCCO DI SIMULAZIONI IN ASSENZA DI BUFFER
START
INPUT
N #dimensione del lotto da simulare
NM #numero di componenti diversi attesi alla stazione di matching
TM #intertempi di uscita medi delle stazioni a monte
NS #numero di simulazioni
FINAL_ARRAY = matrice (8, NS)
inizializzata a 0
for
COUNT_MAIN < NS
NO
YES
SERV_MS = CHIAMATA SIMULATORE
SINGOLO SENZA BUFFER
FINAL_ARRAY[0,COUNT_MAIN]=SERV_MS[0] #La prima linea di FINAL ARRAY sono gli intertempi medi di ingresso
FINAL_ARRAY[2,COUNT_MAIN]=SERV_MS[1] #La terza linea di FINAL ARRAY sono i tempi medi di attesa
FINAL_ARRAY[4,COUNT_MAIN]=SERV_MS[2] #La quinta linea di FINAL ARRAY sono i tempi medi interuscita
FINAL_ARRAY[6,COUNT_MAIN]=SERV_MS[3] #La settima linea di FINAL ARRAY sono i valori medi di WIP
FINAL_ARRAY[1,COUNT_MAIN]=SERV_MS[4] #La seconda linea di FINAL ARRAY indica varianza tempi interarrivo
FINAL_ARRAY[3,COUNT_MAIN]=SERV_MS[5] #La quarta linea di FINAL ARRAY indica varianza tempi di attesa
FINAL_ARRAY[5,COUNT_MAIN]=SERV_MS[6] #La sesta linea di FINAL ARRAY indica varianza tempi interuscita
FINAL_ARRAY[7,COUNT_MAIN]=SERV_MS[7] #L'ottava linea di FINAL ARRAY indica varianza livelli WIP
FINAL = ARRAY di 8 elementi
for
i < 8
NO
YES
SERV1 = 0
for
j < NS
NO
YES
SERV1 += FINAL_ARRAY[i,j]
SERV1 = SERV1 / NS
FINAL[i]= SERV1
OUTPUT
FINAL_ARRAY
FINAL
END


BLOCCO DI SIMULAZIONI IN PRESENZA DI BUFFER
START
INPUT
N #dimensione del lotto da simulare
NM #numero di componenti diversi attesi alla stazione di matching
TM #intertempi di uscita medi delle stazioni a monte
BUFFER #dimensione del buffer
NS #numero di simulazioni
FINAL_ARRAY = matrice (9, NS)
inizializzata a 0
for
COUNT_MAIN < NS
NO
YES
SERV_MS = CHIAMATA SIMULATORE
SINGOLO CON BUFFER
FINAL_ARRAY[0,COUNT_MAIN]=SERV_MS[0] #La prima linea di FINAL ARRAY sono gli intertempi medi di ingresso
FINAL_ARRAY[2,COUNT_MAIN]=SERV_MS[1] #La terza linea di FINAL ARRAY sono i tempi medi di attesa
FINAL_ARRAY[4,COUNT_MAIN]=SERV_MS[2] #La quinta linea di FINAL ARRAY sono i tempi medi interuscita
FINAL_ARRAY[6,COUNT_MAIN]=SERV_MS[3] #La settima linea di FINAL ARRAY sono i valori medi di WIP
FINAL_ARRAY[1,COUNT_MAIN]=SERV_MS[4] #La seconda linea di FINAL ARRAY indica varianza tempi interarrivo
FINAL_ARRAY[3,COUNT_MAIN]=SERV_MS[5] #La quarta linea di FINAL ARRAY indica varianza tempi di attesa
FINAL_ARRAY[5,COUNT_MAIN]=SERV_MS[6] #La sesta linea di FINAL ARRAY indica varianza tempi interuscita
FINAL_ARRAY[7,COUNT_MAIN]=SERV_MS[7] #L'ottava linea di FINAL ARRAY indica varianza livelli WIP
FINAL_ARRAY[8,COUNT_MAIN]=SERV_MS[8] #La nona linea di FINAL ARRAY contiene i turni medi necessari alla saturazione
FINAL = ARRAY di 9 elementi
for
i < 9
NO
YES
SERV1 = 0
for
j < NS
NO
YES
SERV1 += FINAL_ARRAY[i,j]
SERV1 = SERV1 / NS
FINAL[i]= SERV1
OUTPUT
FINAL_ARRAY
FINAL
END

Riassunto dati in output del modello e dei simulatore




DATI IN OUTPUT DI CALCOLATORE TEORICO (IN ASSENZA DI BUFFER)
- ARRAY DEGLI INTERTEMPI MEDI IN INGRESSO DEI SINGOLI PACCHETTI CHE DEVONO ESSERE ASSEMBLATI ALLA STAZIONE DI ACCOPPIAMENTO
- ARRAY DELLE VARIANZE DEGLI INTERTEMPI MEDI IN INGRESSO DEI SINGOLI PACCHETTI
- MEDIA DEGLI INTERTEMPI MEDI
- MEDIA DELLE VARIANZE DEGLI INTERTEMPI
N.B.:
- gli intertempi vengono presi tra i momenti di ingresso alla stazione di accoppiamento del primo componente di due pacchetti consecutivi
- gli intertempi medi e le varianze vengono calcolate tramite le funzioni di densità di probabilità specifiche dei singoli pacchetti
- ARRAY DEGLI INTERTEMPI MEDI DI USCITA DEI SINGOLI PACCHETTI CHE DEVONO ESSERE ASSEMBLATI DALLA STAZIONE DI ACCOPPIAMENTO
- ARRAY DELLE VARIANZE DEGLI INTERTEMPI MEDI DI USCITA DEI SINGOLI PACCHETTI
- MEDIA DEGLI INTERTEMPI MEDI
- MEDIA DELLE VARIANZE DEGLI INTERTEMPI
N.B.:
- gli intertempi vengono presi tra i momenti di ingresso alla stazione di accoppiamento dell'ultimo componente di due pacchetti consecutivi (che equivale al momento di rilascio dalla stazione)
- gli intertempi medi e le varianze vengono calcolate tramite le funzioni di densità di probabilità specifiche dei singoli pacchetti
- ARRAY DEI TEMPI MEDI DI ATTESA DEI SINGOLI PACCHETTI CHE DEVONO ESSERE ASSEMBLATI DALLA STAZIONE DI ACCOPPIAMENTO
- ARRAY DELLE VARIANZE DEI TEMPI MEDI DI ATTESA DEI SINGOLI PACCHETTI
- MEDIA DEGLI TEMPI MEDI
- MEDIA DELLE VARIANZE DEI ITEMPI
N.B.:
- i tempi vengono presi tra i momenti di ingresso alla stazione di accoppiamento del primo e dell'ultimo componente del singolo pacchetto
- gli intertempi medi e le varianze vengono calcolate tramite le funzioni di densità di probabilità specifiche dei singoli pacchetti
- MATRICE DELLA PROBABILITA' DEL LIVELLO DI WIP AL MOMENTO DELL'USCITA DI OGNI SINGOLO PACCHETTO DALLA STAZIONE DI ACCOPPIAMENTO
- LIVELLO DI WIP MEDIO CALCOLATO TRAMITE LA MATRICE
- VARIANZA DEL LIVELLO DI WIP
- CALCOLO DELL'INTERTEMPO MEDIO IN INGRESSO GLOBALE
- CALCOLO DELLA VARIANZA GLOBALE DEGLI INTERTEMPI IN INGRESSO
- CALCOLO DELL'INTERTEMPO MEDIO DI USCITA GLOBALE
- CALCOLO DELLA VARIANZA GLOBALE DEGLI INTERTEMPI DI USCITA
- CALCOLO DEL TEMPO MEDIO DI ATTESA GLOBALE
- CALCOLO DELLA VARIANZA GLOBALE DEL TEMPO DI ATTESA
N.B.:
- questi valori vengono ricavati tramite delle funzioni di densità di probabilità relative al comportamento degli intertempi di ingresso, d'uscita ed ai tempi di attesa durante tutto il processo di accoppiamento di un lotto di N pacchetti
- CALCOLO DI LIVELLI DI WIP MEDIO, TEMPO DI ATTESA MEDIO E TH IN USCITA TRAMITE LEGGE DI LITTLE UTILIZZANDO DATI GLOBALI O MEDIA DEI PUNTUALI


DATI IN OUTPUT DI CALCOLATORE TEORICO (IN PRESENZA DI BUFFER)
- LIVELLO DI WIP MEDIO
- TH IN USCITA
- CT

Programma Phyton

```
import mpmath as mp
import random

def LIBRARY_SELECTOR(TYPE_LS,DATA_LS):
#Funzione custom dedicata alla definizione delle linee di testo e dei campi di input.
#Viene richiamata dalla funzione custom INPUT_MENU.
#TYPE_LS contiene il codice della libreria da richiamare
#DATA_LS contiene alcuni dati in base ai quali cambiano alcuni parametri della libreria selezionata
#La selezione avviene attraverso una "IF" multipla.
#La prima linea che verrà fornita in output nel vettore LIBRARY_LS contiene un array di due
#numeri. Il primo indica il #numero di linee di testo "inerti", il secondo le linee che diventeranno
#dei campi per l'inserimento
#Nel vettore INPUT_LS, anch'esso destinato all'output, verranno immessi i parametri di valutazione
#di corretto inserimento dell'input richiesto
    LIBRARY_LS=[]
    INPUT_LS=[]
    if TYPE_LS=='main':
        LIBRARY_LS.append([5,0])
        LIBRARY_LS.append('\n')
        LIBRARY_LS.append('SELETTORE MODALITA')
        LIBRARY_LS.append('Inserire (1) per accedere alle schermate utente')
        LIBRARY_LS.append('Inserire (2) per accedere alla sezione creazione\esecuzione
            schedulazione elaborazioni')
        LIBRARY_LS.append('Inserire (0) per uscire')
        LIBRARY_LS.append(["","",""])
        INPUT_LS.append([0,1,'I','a',[0,1,2]])

    elif TYPE_LS=='user_main':
        LIBRARY_LS.append([6,0])
        LIBRARY_LS.append('\n')
        LIBRARY_LS.append('MENU UTENTE')
        LIBRARY_LS.append('Inserire (1) per effettuare operazioni di calcolo teorico')
        LIBRARY_LS.append('Inserire (2) per effettuare delle simulazioni')
        LIBRARY_LS.append('Inserire (3) per effettuare delle comparazioni teoria/simulazione')
        LIBRARY_LS.append('Inserire (0) per tornare al menù principale')
        LIBRARY_LS.append(["","",""])
        INPUT_LS.append([0,1,'I','a',[0,1,2,3]])

    elif TYPE_LS=='user_theory':
        LIBRARY_LS.append([10,0])
        LIBRARY_LS.append('\n')
        LIBRARY_LS.append('MENU UTENTE')
        LIBRARY_LS.append('Inserire (1) per caricare in memoria matrici di base preesistenti al fine
            di successive operazioni')
        LIBRARY_LS.append('Inserire (2) per caricare in memoria tabulati TIMELESS')
        LIBRARY_LS.append('Inserire (3) per caricare in memoria parametri di un caso specifico già
```

```
        calcolato al fine di successive operazioni')
    LIBRARY_LS.append('Inserire (4) per generare nuove tabelle di base delle probabilità dei
        tempi di ingresso e attesa')
    LIBRARY_LS.append('Inserire (5) per generare nuovi tabulati timeless ')
    LIBRARY_LS.append('Inserire (6) per calcolare dati teorici di un caso specifico in assenza di
        buffer')
    LIBRARY_LS.append('Inserire (7) per calcolare dati teorici di un caso specifico in presenza di
        buffer')
    LIBRARY_LS.append('Inserire (0) per
        tornare al menù precedente')
    LIBRARY_LS.append(['',','])
    INPUT_LS.append([0,1,'I','a',[0,1,2,3,4,5,6,7,8]])

elif TYPE_LS=='buff1_sub':
    LIBRARY_LS.append([2,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('Si desidera calcolare i parametri della stazione in presenza di buffer?')
    LIBRARY_LS.append(['','Sì=1/No=2',''])
    INPUT_LS.append([0,1,'I','a',[1,2]])

elif TYPE_LS=='buff2_sub':
    LIBRARY_LS.append([2,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('Si desidera ripetere la computazione con un buffer diverso?')
    LIBRARY_LS.append(['','Sì=1/No=2',''])
    INPUT_LS.append([0,1,'I','a',[1,2]])

elif TYPE_LS=='mtx_sav':
    LIBRARY_LS.append([2,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('Si desidera mantenere in memoria matrici appena elaborate?')
    LIBRARY_LS.append(['','Sì=1/No=2',''])
    INPUT_LS.append([0,1,'I','a',[1,2]])

elif TYPE_LS=='timeless_sav':
    LIBRARY_LS.append([2,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('Si desidera mantenere in memoria gli ultimi tabulati timeless
        elaborati?')
    LIBRARY_LS.append(['','Sì=1/No=2',''])
    INPUT_LS.append([0,1,'I','a',[1,2]])

elif TYPE_LS=='tab_sav':
    LIBRARY_LS.append([2,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('Si desidera mantenere in memoria gli ultimi tabulati elaborati?')
    LIBRARY_LS.append(['','Sì=1/No=2',''])
    INPUT_LS.append([0,1,'I','a',[1,2]])

elif TYPE_LS=='buff_sav':
```

```
    LIBRARY_LS.append([3,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('Si desidera mantenere in memoria tabulati con questo livello di
      buffer?')
    LIBRARY_LS.append('Verranno salvati anche i tabulati di base sui quali il corrente BUFFER
      viene applicato')
    LIBRARY_LS.append(['','Sì=1/No=2',''])
    INPUT_LS.append([0,1,'I','a',[1,2]])

elif TYPE_LS=='mtx_gen':
    LIBRARY_LS.append([3,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('PROCEDURA INSERIMENTO DATI PER CREAZIONE TABELLE
      DI BASE DELLE FUNZIONI DI PROBABILITA DEI TEMPI DI INGRESSO E ATTESA')
    LIBRARY_LS.append('Nel caso di erronea immissione di dati float, questi verranno
      arrotondati al numero intero inferiore')
    LIBRARY_LS.append(['quantità del lotto','numero intero maggiore di 4','sulla quale elaborare
      le tabelle'])
    if len(DATA_LS)==1:
      LIBRARY_LS.append(['numero di decimali dopo la virgola','maggiore o uguale a 1','che si
      desidera in output file utente'])
    #end if
    INPUT_LS.append([0,1,'I','>',4])
    if len(DATA_LS)==1:
      INPUT_LS.append([0,1,'I','>',0])
    #end if

elif TYPE_LS=='timeless_gen':
    LIBRARY_LS.append([3,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('PROCEDURA INSERIMENTO DATI PER CREAZIONE
      TABULATI DI BASE TIMELESS')
    LIBRARY_LS.append('Nel caso di erronea immissione di dati float, questi verranno
      arrotondati al numero intero inferiore')
    LIBRARY_LS.append(['dimensione massima dei lotti elaborabili in seguito su questi
      tabulati','numero intero, >4, <%d capacita massima matrici base'%(DATA_LS[0]+1),''])
    if len(DATA_LS)==1:
      LIBRARY_LS.append(['numero di decimali dopo la virgola ai fini della stampa su
            file','maggiore o uguale a 1',''])
    INPUT_LS.append([0,1,'I','><',[4,DATA_LS[0]+1]])
    if len(DATA_LS)==1:
      INPUT_LS.append([0,1,'I','>',1])

elif TYPE_LS=='tab_gen':
    LIBRARY_LS.append([3,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('PROCEDURA INSERIMENTO DATI DEL LOTTO PER
      GENERAZIONE TABULATI CASO SPECIFICO IN ASSENZA DI BUFFER')
    LIBRARY_LS.append('Nel caso di erronea immissione di dati float, questi verranno
      arrotondati al numero intero inferiore')
```

```
    LIBRARY_LS.append(['quantità del lotto da processare','numero intero, >4, <%d capacita
      massima matrici base'%(DATA_LS[0]+1),''])
    LIBRARY_LS.append(['il tempo medio di uscita dei componenti a monte della stazione di
      matching','formato xxxx.yyyy',''])
    LIBRARY_LS.append(['il codice relativo all unità di misura dei
      tempi','s=secondi/m=minuti/h=ore',''])
    if len(DATA_LS)==1:
      LIBRARY_LS.append(['numero di decimali dopo la virgola ai fini della stampa su
          file','maggiore o uguale a 1',''])
    #end if
    INPUT_LS.append([0,1,'I','><',[4,DATA_LS[0]+1]])
    INPUT_LS.append([0,1,'F','>',0])
    INPUT_LS.append([0,1,'S','a',['s','m','h']])
    if len(DATA_LS)==1:
      INPUT_LS.append([0,1,'I','>',1])
    #end if

elif TYPE_LS=='buff_gen':
    LIBRARY_LS.append([2,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('PROCEDURA INSERIMENTO DATI BUFFER')
    LIBRARY_LS.append(['la dimensione del buffer','numero intero, >0, <%d'%
      (DATA_LS[0]+1),''])
    if len(DATA_LS)==1:
      LIBRARY_LS.append(['numero di decimali dopo la virgola ai fini della stampa su
          file','maggiore o uguale a 1',''])
    #end if
    INPUT_LS.append([0,1,'I','><',[0,DATA_LS[0]+1]])
    if len(DATA_LS)==1:
      INPUT_LS.append([0,1,'I','>',1])      #codice scelta decimali in stampa
    #end if

elif TYPE_LS=='mtx_inp':
    LIBRARY_LS.append([6,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('PROCEDURA SELEZIONE FILE MATRICE DA CARICARE IN
      MEMORIA')
    LIBRARY_LS.append('I files matrice devono essere nella stessa cartella di questo
      programma')
    LIBRARY_LS.append('I loro nomi sono nella forma MTX_(Numero)u_(Numero)d_*.%s'%
      (F_EXT))
    LIBRARY_LS.append('Il primo numero rappresenta il numero massimo di coppie per le quali
      sono calcolabili i parametri (maggiore di 0')
    LIBRARY_LS.append('Il secondo numero rappresenta la precisione decimale con la quale
      sono stati salvati i dati al loro interno')
    LIBRARY_LS.append(['il numero identificativo delle unità','deve essere un numero intero
      positivo',''])
    LIBRARY_LS.append(['il numero identificativo della precisione decimale','deve essere un
      numero intero positivo',''])
    INPUT_LS.append([0,1,'I','>',4])  #codice scelta unità lotto
```

```
    INPUT_LS.append([0,1,'I','>',0])  #codice scelta precisione

elif TYPE_LS=='timeless_inp':
    LIBRARY_LS.append([6,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('PROCEDURA SELEZIONE FILE TABULATI TIMELESS DA
       CARICARE IN MEMORIA')
    LIBRARY_LS.append('I files dei tabulati timeless devono essere nella stessa cartella di questo
       programma')
    LIBRARY_LS.append('I loro nomi sono nella forma TL_TAB_(Numero)u_(Numero)d_*.
       %s'%(F_EXT))
    LIBRARY_LS.append('Il primo numero rappresenta il numero massimo di coppie per le quali
       sono calcolabili i parametri (maggiore di 0')
    LIBRARY_LS.append('Il secondo numero rappresenta la precisione decimale con la quale
       sono stati salvati i dati al loro interno')
    if len(DATA_LS)>1:
       LIBRARY_LS.append(['il numero identificativo delle unità','numero intero, >4, <%d
       capacita massima matrici base'%(DATA_LS[0]+1),''])
    else:
       LIBRARY_LS.append(['il numero identificativo delle unità','deve essere un numero intero
       positivo',''])
    #end if
    LIBRARY_LS.append(['il numero identificativo della precisione decimale','deve essere un
       numero intero positivo',''])
    if len(DATA_LS)>1:
       INPUT_LS.append([0,1,'I','><',[4,DATA_LS[0]+1]])  #codice scelta unità lotto
    else:
       INPUT_LS.append([0,1,'I','>',4])  #codice scelta unità lotto
    #end if
    INPUT_LS.append([0,1,'I','>',0])  #codice scelta precisione

elif TYPE_LS=='tab_inp':
    LIBRARY_LS.append([8,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('PROCEDURA SELEZIONE FILE ELABORAZIONE DA
       CARICARE IN MEMORIA')
    LIBRARY_LS.append('I files di precedenti elaborazioni devono essere nella stessa cartella di
       questo programma')
    LIBRARY_LS.append('I loro nomi sono nella forma TABS_(Numero)u_(Numero float)
       (sec/min/h)_(Numero)d_*.%s'%(F_EXT))
    LIBRARY_LS.append('Il primo numero rappresenta il numero di coppie per le quali sono stati
       calcolati (maggiore di 4)')
    LIBRARY_LS.append('Il secondo numero rappresenta il tempo medio di uscita dalle stazioni a
       monte con il quale sono stati calcolati (maggiore di 0)')
    LIBRARY_LS.append('Il terzo campo rappresenta l unità di misura dei tempi (sec/min/h)')
    LIBRARY_LS.append('Il quarto numero rappresenta la precisione decimale con la quale sono
       stati salvati i dati al loro interno (maggiore di 0)')
    if len(DATA_LS)>1:
       LIBRARY_LS.append(['il numero identificativo delle unità','numero intero, >4, <%d
              capacita massima tabulati timeless'%(DATA_LS[0]+1),''])
```

```
    else:
        LIBRARY_LS.append(['il numero identificativo delle unità','deve essere un numero intero
            positivo',''])
    #end if
    LIBRARY_LS.append(['il numero identificativo degli intertempi di uscita a
        monte','esattamente come scritto nel nome del file',''])
    LIBRARY_LS.append(['il codice relativo all unità di misura dei
        tempi','s=secondi/m=minuti/h=ore',''])

 LIBRARY_LS.append(['il numero identificativo della precisione decimale','deve essere un
        numero intero positivo',''])
    if len(DATA_LS)>1:
        INPUT_LS.append([0,1,'I','><',[4,DATA_LS[0]+1]])
    else:
        INPUT_LS.append([0,1,'I','>',4])
    #end if
    INPUT_LS.append([0,1,'S','NULL',0])
    INPUT_LS.append([0,1,'S','a',['s','m','h']])
    INPUT_LS.append([0,1,'I','>',0])

elif TYPE_LS=='user_simulation':
    LIBRARY_LS.append([7,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('MENU UTENTE')
    LIBRARY_LS.append('Inserire (1) per eseguire una simulazione singola in assenza di buffer')
    LIBRARY_LS.append('Inserire (2) per eseguire una simulazione singola in presenza di buffer')
    LIBRARY_LS.append('Inserire (3) per eseguire un blocco di simulazioni in assenza di buffer')
    LIBRARY_LS.append('Inserire (4) per eseguire un blocco di simulazioni in presenza di
        buffer')
    LIBRARY_LS.append('Inserire (0) per tornare al menù precedente')
    LIBRARY_LS.append(['','',''])
    INPUT_LS.append([0,1,'I','a',[0,1,2,3,4]])

elif TYPE_LS=='sim_data':
    LIBRARY_LS.append([4,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('PROCEDURA INSERIMENTO DATI DELLA SINGOLA
        SIMULAZIONE\n')  #print è il comando di scrittura a schermo
    LIBRARY_LS.append('Verranno richiesti dei numeri interi positivi.\n')
    LIBRARY_LS.append('Nel caso di erronea immissione di dati float, questi verranno
        arrotondati all intero inferiore')
    LIBRARY_LS.append(['il numero di linee a monte della stazione di accoppiamento','maggiore
        o uguale a 2',''])
    LIBRARY_LS.append(['il tempo medio di uscita dei componenti a monte della stazione di
        matching','formato xxxx.yyyy',''])
    LIBRARY_LS.append(['unità di misura dei tempi', 's=secondi/m=minuti/h=ore',''])
    LIBRARY_LS.append(['quantità del lotto da processare','numero intero maggiore di 4',''])
    LIBRARY_LS.append(['numero di decimali dopo la virgola ai fini della stampa su
        file','maggiore o uguale a 1',''])
    INPUT_LS.append([0,1,'I','>',1])
```

```
    INPUT_LS.append([0,1,'F','>',0])
    INPUT_LS.append([0,1,'S','a',['s','m','h']])
    INPUT_LS.append([0,1,'I','>',4])
    INPUT_LS.append([0,1,'I','>',1])

elif TYPE_LS=='sim_data_m':
    LIBRARY_LS.append([2,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('PROCEDURA INSERIMENTO DATI: DIMENSIONE BLOCCO DI
      SIMULAZIONE\n')
    LIBRARY_LS.append(['numero di simulazioni che si desidera eseguire','maggiore o uguale a
      2',''])
    INPUT_LS.append([0,1,'I','>',1])

elif TYPE_LS=='sim_data_b':
    LIBRARY_LS.append([2,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('PROCEDURA INSERIMENTO DATI: VALORE BUFFER\n')
    LIBRARY_LS.append(['valore limite del buffer','numero intero, >0, <%d'%
      (DATA_LS[0]+1),''])
    INPUT_LS.append([0,1,'I','><',[1,(DATA_LS[0]+1)]])

elif TYPE_LS=='save_sim':
    LIBRARY_LS.append([2,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('Si desidera salvare %s (1=Sì/2=No)?'%(DATA_LS))
    LIBRARY_LS.append(['','',''])
    INPUT_LS.append([0,1,'I','a',[1,2]])

elif TYPE_LS=='main_comp':
    LIBRARY_LS.append([6,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('MENU PRINCIPALE COMPARATORE')
    LIBRARY_LS.append('Inserire (1) per elaborare blocco comparazione su tabulati senza
      buffer')
    LIBRARY_LS.append('Inserire (2) per elaborare blocco comparazione su tabulati con buffer
      (buffer variabile)')
    LIBRARY_LS.append('Inserire (3) per elaborare blocco comparazione su tabulati con buffer
      (buffer fisso, dimensione lotto variabile)')
    LIBRARY_LS.append('Inserire (0) per tornare al menù precedente')
    LIBRARY_LS.append(['','',''])
    INPUT_LS.append([0,1,'I','a',[0,1,2,3]])

elif TYPE_LS=='new_load_mtx':
    LIBRARY_LS.append([3,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('MENU IMPOSTAZIONI SCHEDULATORE')
    LIBRARY_LS.append('Si desidera caricare un diverso set di matrici timeless (1=Sì/2=No)?')
    LIBRARY_LS.append(['','',''])
    INPUT_LS.append([0,1,'I','a',[1,2]])
```

```
elif TYPE_LS=='new_load_tabs':
    LIBRARY_LS.append([3,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('MENU IMPOSTAZIONI SCHEDULATORE')
    LIBRARY_LS.append('Si desidera caricare un diverso set di tabulati di base (1=Sì/2=No)?')
    LIBRARY_LS.append(["","",""])
    INPUT_LS.append([0,1,'I','a',[1,2]])

elif TYPE_LS=='comp_data_tabs':
    LIBRARY_LS.append([2,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('INSERIRE DATI PER CREAZIONE CICLO COMPARAZIONE
      TABULATI SENZA BUFFER')
    LIBRARY_LS.append(['il numero totale di casi che si vogliono comparare','intero positivo
      maggiore o uguale a 1',''])
    LIBRARY_LS.append(['il passo incrementale tra un ciclo e il successivo','intero positivo
      maggiore o uguale a 1','calcolato sulla dimensione del lotto'])
    INPUT_LS.append([0,1,'I','>',0])
    INPUT_LS.append([0,1,'I','>',0])

elif TYPE_LS=='comp_data_buff1':
    LIBRARY_LS.append([2,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('INSERIRE DATI PER CREAZIONE CICLO COMPARAZIONE
      TABULATI CON BUFFER')
    LIBRARY_LS.append(['il numero totale di casi che si vogliono comparare','intero positivo >1
      <%d'%(DATA_LS[0]+1),''])
    INPUT_LS.append([0,1,'I','><',[1,(DATA_LS[0]+1)]])

elif TYPE_LS=='comp_data_buff2':
    LIBRARY_LS.append([2,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('INSERIRE DATI PER CREAZIONE CICLO COMPARAZIONE
      TABULATI CON BUFFER')
    LIBRARY_LS.append(['il passo incrementale tra un ciclo e il successivo','Numero intero >0 <
      %d'%(DATA_LS[0]+1),'calcolato sulla dimensione del buffer'])
    INPUT_LS.append([0,1,'I','><',[0,(DATA_LS[0]+1)]])

elif TYPE_LS=='main_sched':
    LIBRARY_LS.append([5,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('MENU IMPOSTAZIONI SCHEDULATORE')
    LIBRARY_LS.append('Inserire (1) per caricare in memoria ed eseguire file di schedulazione')
    LIBRARY_LS.append('Inserire (2) per creare file di schedulazione')
    LIBRARY_LS.append('Inserire (0) per tornare al menù principale')
    LIBRARY_LS.append(["","",""])
    INPUT_LS.append([0,1,'I','a',[0,1,2]])

elif TYPE_LS=='er_main':
```

```
    LIBRARY_LS.append([2,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('Si desidera eseguire una nuova operazione?')
    LIBRARY_LS.append(['','Sì=1/No=2',''])
    INPUT_LS.append([0,1,'I','a',[1,2]])

elif TYPE_LS=='er_sub':
    LIBRARY_LS.append([2,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('Si desidera eseguire una nuova operazione in questa sezione?')
    LIBRARY_LS.append(['','Sì=1/No=2',''])
    INPUT_LS.append([0,1,'I','a',[1,2]])

elif TYPE_LS=='tab_repeat':
    LIBRARY_LS.append([2,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('Si desidera eseguire una nuova elaborazione di caso specifico?')
    LIBRARY_LS.append(['','Sì=1/No=2',''])
    INPUT_LS.append([0,1,'I','a',[1,2]])

elif TYPE_LS=='scr_main':
    LIBRARY_LS.append([5,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('PROCEDURA CREAZIONE FILE DI SCHEDULAZIONE')
    LIBRARY_LS.append('Inserire (1) per creare nuovo file di schedulazione')
    LIBRARY_LS.append('Inserire (2) per modificare file di schedulazione già esistente (NON
      IMPLEMENTATO)')
    LIBRARY_LS.append('Inserire (0) per uscire')
    LIBRARY_LS.append(['','',''])
    INPUT_LS.append([0,1,'I','a',[0,1,2]])

elif TYPE_LS=='sched_name_load':
    LIBRARY_LS.append([4,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('PROCEDURA SELEZIONE FILE DI SCHEDULAZIONE LAVORO
      DA ESEGUIRE')
    LIBRARY_LS.append('Il file selezionabile deve essere nella stessa cartella del programma
      pyton attualmente in esecuzione')
    LIBRARY_LS.append('I files selezionabili hanno i nomi in forma: SCHED_(descrizione).scd')
    LIBRARY_LS.append(['descrizione del file','',''])
    INPUT_LS.append([0,1,'S','NULL',0])

elif TYPE_LS=='sched_name_save':
    LIBRARY_LS.append([4,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('PROCEDURA INSERIMENTO NOME FILE DI
      SCHEDULAZIONE')
    LIBRARY_LS.append('I files di schedulazione hanno i nomi in forma:
      SCHED_(descrizione).scd')
    LIBRARY_LS.append('Il file verrà salvato nella stessa cartella contenente il programma
```

```
        python attualmente in esecuzione')
    LIBRARY_LS.append(['la descrizione da inserire nel nome','stringa di caratteri',''])
    INPUT_LS.append([0,1,'S','NULL',0])

elif TYPE_LS=='scr_opr_family':
    LIBRARY_LS.append([6,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('MENU COMANDI')
    LIBRARY_LS.append('Inserire (1) per inserire un comando relativo a matrici e tabulati
        teorici')
    LIBRARY_LS.append('Inserire (2) per inserire un comando relativo alle simulazioni')
    LIBRARY_LS.append('Inserire (3) per inserire un comando relativo alle comparazioni')
    LIBRARY_LS.append('Inserire (0) per inserire comando fine schedulazione, salvare ed
        uscire')
    LIBRARY_LS.append(['',','])
    INPUT_LS.append([0,1,'I','a',[0,1,2,3]])

elif TYPE_LS=='scr_opr_theo':
    LIBRARY_LS.append([14,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('MENU COMANDI')
    LIBRARY_LS.append('Inserire (1) per inserire comando caricamento files matrici di base')
    LIBRARY_LS.append('Inserire (2) per inserire comando caricamento files tabulati timeless')
    LIBRARY_LS.append('Inserire (3) per inserire comando caricamento files di dati di un caso
        specifico')
    LIBRARY_LS.append('Inserire (4) per inserire comando elaborazione di matrici di base, loro
        mantenimento in memoria e salvataggio su files')
    LIBRARY_LS.append('Inserire (5) per inserire comando elaborazione di tabulati timeless, loro
        mantenimento in memoria e salvataggio su files')
    LIBRARY_LS.append('Inserire (6) per inserire comando elaborazione di files di dati di un
        caso specifico in assenza di buffer, loro mantenimento in memoria e salvataggio su files')
    LIBRARY_LS.append('Inserire (7) per inserire comando elaborazione di files di dati di un
        caso specifico in presenza di buffer e loro salvataggio su files')
    LIBRARY_LS.append('Inserire (8) per inserire CICLO ELABORAZIONI NUOVI FILES
        MATRICE (la matrice di partenza deve già essere caricata in memoria)')
    LIBRARY_LS.append('Inserire (9) per inserire CICLO ELABORAZIONI NUOVI FILES
        TIMELESS (la matrice di partenza deve già essere caricata in memoria)')
    LIBRARY_LS.append('Inserire (10) per inserire CICLO ELABORAZIONI CASI SPECIFICI
        (il caso specifico di partenza deve già essere caricato in memoria)')
    LIBRARY_LS.append('Inserire (11) per inserire CICLO ELABORAZIONI BUFFER (il caso
        di partenza deve già essere caricato in memoria)')
    LIBRARY_LS.append('Inserire (0) per ritornare al menù precedente')
    LIBRARY_LS.append(['',','])
    INPUT_LS.append([0,1,'I','a',[0,1,2,3,4,5,6,7,8,9,10,11]])

elif TYPE_LS=='scr_opr_sim':
    LIBRARY_LS.append([11,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('MENU COMANDI')
    LIBRARY_LS.append('Inserire (1) per inserire comando generazione simulazione singola
```

```
        senza buffer')
    LIBRARY_LS.append('Inserire (2) per inserire comando generazione simulazione singola con
        buffer')
    LIBRARY_LS.append('Inserire (3) per inserire generazione blocco di simulazioni singole
        senza buffer')
    LIBRARY_LS.append('Inserire (4) per inserire generazione blocco di simulazioni singole con
        buffer')
    LIBRARY_LS.append('Inserire (5) per inserire CICLO ELABORAZIONI simulazione singola
        senza buffer')
    LIBRARY_LS.append('Inserire (6) per inserire CICLO ELABORAZIONI simulazione singola
        con buffer')
    LIBRARY_LS.append('Inserire (7) per inserire CICLO ELABORAZIONI blocco di
        simulazioni singole senza buffer')
    LIBRARY_LS.append('Inserire (8) per inserire CICLO ELABORAZIONI blocco di
        simulazioni singole con buffer')
    LIBRARY_LS.append('Inserire (0) per ritornare al menù precedente')
    LIBRARY_LS.append(["","",""])
    INPUT_LS.append([0,1,'I','a',[0,1,2,3,4,5,6,7,8]])

elif TYPE_LS=='scr_opr_comp':
    LIBRARY_LS.append([7,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('MENU COMANDI')
    LIBRARY_LS.append('Inserire (1) per inserire comando generazione comparazione tabulati
        senza buffer con N incrementale')
    LIBRARY_LS.append('Inserire (2) per inserire comando generazione comparazione tabulati
        con buffer con BUFFER incrementale')
    LIBRARY_LS.append('Inserire (3) per inserire CICLO ELABORAZIONI comparazione
        tabulati senza buffer con N incrementale')
    LIBRARY_LS.append('Inserire (4) per inserire CICLO ELABORAZIONI comparazione
        tabulati con buffer con BUFFER incrementale')
    LIBRARY_LS.append('Inserire (0) per ritornare al menù precedente')
    LIBRARY_LS.append(["","",""])
    INPUT_LS.append([0,1,'I','a',[0,1,2,3,4]])

elif TYPE_LS=='src_close':
    LIBRARY_LS.append([2,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('Si desidera veramente salvare e chiudere il file di schedulazione?')
    LIBRARY_LS.append(['','Sì=1/No=2',''])
    INPUT_LS.append([0,1,'I','a',[1,2]])

elif TYPE_LS=='scr_cyc_field_snb':
    LIBRARY_LS.append([5,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('SCEGLIERE QUALE CAMPO RENDERE INCREMENTALE')
    LIBRARY_LS.append('Inserire (1) per sviluppare una serie di simulazioni singole senza buffer
        INCREMENTANDO IL NUMERO DI PACCHETTI')
    LIBRARY_LS.append('Inserire (2) per sviluppare una serie di simulazioni singole senza buffer
        INCREMENTANDO IL TEMPO DI INTERUSCITA DELLE STAZIONI A MONTE')
```

```
    LIBRARY_LS.append('Inserire (0) per sviluppare una serie di simulazioni singole senza buffer
      INCREMENTANDO DELLE STAZIONI A MONTE')
    LIBRARY_LS.append(["","",""])
    INPUT_LS.append([0,1,'I','a',[0,1,2]])

elif TYPE_LS=='scr_cyc_field_sbuff':
    LIBRARY_LS.append([6,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('SCEGLIERE QUALE CAMPO RENDERE INCREMENTALE')
    LIBRARY_LS.append('Inserire (1) per sviluppare una serie di simulazioni singole con buffer
      INCREMENTANDO IL NUMERO DI PACCHETTI')
    LIBRARY_LS.append('Inserire (2) per sviluppare una serie di simulazioni singole con buffer
      INCREMENTANDO IL TEMPO DI INTERUSCITA DELLE STAZIONI A MONTE')
    LIBRARY_LS.append('Inserire (3) per sviluppare una serie di simulazioni singole con buffer
      INCREMENTANDO DELLE STAZIONI A MONTE')
    LIBRARY_LS.append('Inserire (0) per sviluppare una serie di simulazioni singole con buffer
      INCREMENTANDO LA DIMENSIONE DEL BUFFER')
    LIBRARY_LS.append(["","",""])
    INPUT_LS.append([0,1,'I','a',[0,1,2,3]])

elif TYPE_LS=='scr_cyc_field_mnb':
    LIBRARY_LS.append([6,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('SCEGLIERE QUALE CAMPO RENDERE INCREMENTALE')
    LIBRARY_LS.append('Inserire (1) per sviluppare una serie di simulazioni multiple con buffer
      INCREMENTANDO IL NUMERO DI PACCHETTI')
    LIBRARY_LS.append('Inserire (2) per sviluppare una serie di simulazioni multiple con buffer
      INCREMENTANDO IL TEMPO DI INTERUSCITA DELLE STAZIONI A MONTE')
    LIBRARY_LS.append('Inserire (3) per sviluppare una serie di simulazioni multiple con buffer
      INCREMENTANDO DELLE STAZIONI A MONTE')
    LIBRARY_LS.append('Inserire (0) per sviluppare una serie di simulazioni multiple con buffer
      INCREMENTANDO IL NUMERO DI SIMULAZIONI INCLUSE NEL BLOCCO DI
      CALCOLO')
    LIBRARY_LS.append(["","",""])
    INPUT_LS.append([0,1,'I','a',[0,1,2,3]])

elif TYPE_LS=='scr_cyc_field_mbuff':
    LIBRARY_LS.append([7,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('SCEGLIERE QUALE CAMPO RENDERE INCREMENTALE')
    LIBRARY_LS.append('Inserire (1) per sviluppare una serie di simulazioni multiple con buffer
      INCREMENTANDO IL NUMERO DI PACCHETTI')
    LIBRARY_LS.append('Inserire (2) per sviluppare una serie di simulazioni multiple con buffer
      INCREMENTANDO IL TEMPO DI INTERUSCITA DELLE STAZIONI A MONTE')
    LIBRARY_LS.append('Inserire (3) per sviluppare una serie di simulazioni multiple con buffer
      INCREMENTANDO DELLE STAZIONI A MONTE')
    LIBRARY_LS.append('Inserire (4) per sviluppare una serie di simulazioni multiple con buffer
      INCREMENTANDO IL NUMERO DI SIMULAZIONI INCLUSE NEL BLOCCO DI
      CALCOLO')
    LIBRARY_LS.append('Inserire (0) per sviluppare una serie di simulazioni multiple con buffer
```

```
        INCREMENTANDO LA DIMENSIONE DEL BUFFER')
    LIBRARY_LS.append(["","",""])
    INPUT_LS.append([0,1,'I','a',[0,1,2,3,4]])

elif TYPE_LS=='scr_cyc_sim':
    LIBRARY_LS.append([2,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('INSERIMENTO DATI PER CREAZIONE CICLO: NUMERO DI
        CICLI E PASSO')
    LIBRARY_LS.append(['il numero totale elaborazioni','intero positivo maggiore o uguale a
        1',''])
    LIBRARY_LS.append(['il passo incrementale tra un ciclo e il successivo','intero positivo
        maggiore o uguale a 1',''])
    INPUT_LS.append([0,1,'I','>',0])
    INPUT_LS.append([0,1,'I','>',0])

elif TYPE_LS=='mtx_cyc_load_gen':
    LIBRARY_LS.append([5,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('CREAZIONE COMANDI PER ESEGUIRE UN CICLO DI
        GENERAZIONE DI MATRICI DI BASE')
    LIBRARY_LS.append('Inserire (1) per iniziare il ciclo con un set di matrici di base generate
        ex-novo')
    LIBRARY_LS.append('Inserire (2) per iniziare il ciclo caricando da files un set di matrici di
        base preesistenti')
    LIBRARY_LS.append('Inserire (0) per ritornare al menù precedente')
    LIBRARY_LS.append(["","",""])
    INPUT_LS.append([0,1,'I','a',[0,1,2]])

elif TYPE_LS=='mtx_cyc':
    LIBRARY_LS.append([2,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('INSERIRE DATI PER CREAZIONE CICLO ELABORAZIONE
        MATRICI DI BASE')
    LIBRARY_LS.append(['numero di decimali dopo la virgola ai fini della stampa su file','intero
        positivo maggiore o uguale a 1',''])
    LIBRARY_LS.append(['il numero totale di set di matrici base che si vogliono elaborare','intero
        positivo maggiore o uguale a 1',''])
    LIBRARY_LS.append(['il passo incrementale tra un ciclo e il successivo','intero positivo
        maggiore o uguale a 1',''])
    INPUT_LS.append([0,1,'I','>',0])
    INPUT_LS.append([0,1,'I','>',0])
    INPUT_LS.append([0,1,'I','>',0])

elif TYPE_LS=='timeless_cyc_load_gen':
    LIBRARY_LS.append([5,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('CREAZIONE COMANDI PER ESEGUIRE UN CICLO DI
        GENERAZIONE TABULATI TIMELESS')
    LIBRARY_LS.append('Inserire (1) per iniziare
```

```
        il ciclo con un set di tabulati timeless generati ex-novo')
    LIBRARY_LS.append('Inserire (2) per iniziare il ciclo caricando da files un set di tabulati
        timeless preesistenti')
    LIBRARY_LS.append('Inserire (0) per ritornare al menù precedente')
    LIBRARY_LS.append(["","",""])
    INPUT_LS.append([0,1,'I','a',[0,1,2]])

elif TYPE_LS=='timeless1_cyc':
    LIBRARY_LS.append([3,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('GENERAZIONE CICLO DI TABULAZIONI TIMELESS')
    LIBRARY_LS.append('INSERIMENTO DATI')
    LIBRARY_LS.append(['numero di decimali dopo la virgola ai fini della stampa su
        file','maggiore o uguale a 1',""])
    LIBRARY_LS.append(['il numero totale di casi che si vogliono elaborare','intero positivo >0 e
        <%d'%(DATA_LS[0]+1),""])
    INPUT_LS.append([0,1,'I','>',0])
    INPUT_LS.append([0,1,'I','><',[0,DATA_LS[0]+1]])

elif TYPE_LS=='timeless2_cyc':
    LIBRARY_LS.append([3,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('GENERAZIONE CICLO DI TABULAZIONI TIMELESS')
    LIBRARY_LS.append('INSERIMENTO DATI')
    LIBRARY_LS.append(['il passo incrementale tra un ciclo e il successivo','intero positivo <
        %d'%(DATA_LS[0]+1),""])
    INPUT_LS.append([0,1,'I','><',[0,DATA_LS[0]+1]])


elif TYPE_LS=='tab_cyc_load_gen':
    LIBRARY_LS.append([5,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('CREAZIONE COMANDI PER ESEGUIRE UN CICLO DI
        GENERAZIONE DI TABULATI DI CASI SPECIFICI IN ASSENZA DI BUFFER')
    LIBRARY_LS.append('Inserire (1) per iniziare il ciclo da un set di tabulati generati ex-novo')
    LIBRARY_LS.append('Inserire (2) per iniziare il ciclo caricando da files un set di tabulati di
        partenza preesistenti')
    LIBRARY_LS.append('Inserire (0) per ritornare al menù precedente')
    LIBRARY_LS.append(["","",""])
    INPUT_LS.append([0,1,'I','a',[0,1,2]])

elif TYPE_LS=='tab1_cyc':
    LIBRARY_LS.append([5,0])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('GENERAZIONE CICLO DI TABULAZIONI IN ASSENZA DI
        BUFFER')
    LIBRARY_LS.append('SCELTA DEL CAMPO INCREMENTALE')
    LIBRARY_LS.append('Inserire (1) per incrementare il numero di coppie a ogni ciclo')
    LIBRARY_LS.append('Inserire (2) per incrementare i tempi di uscita dalle stazioni a monte')
    LIBRARY_LS.append(["","",""])
```

```
    INPUT_LS.append([0,1,'I','a',[1,2]])

elif TYPE_LS=='tab2_cyc':
    LIBRARY_LS.append([3,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('GENERAZIONE CICLO DI TABULAZIONI IN ASSENZA DI
      BUFFER')
    LIBRARY_LS.append('INSERIMENTO DATI')
    LIBRARY_LS.append(['numero di decimali dopo la virgola ai fini della stampa su
      file','maggiore o uguale a 1',''])
    LIBRARY_LS.append(['il numero totale di casi che si vogliono elaborare','intero positivo',''])
    INPUT_LS.append([0,1,'I','>',0])
    INPUT_LS.append([0,1,'I','>',0])

elif TYPE_LS=='tab22_cyc':
    LIBRARY_LS.append([3,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('GENERAZIONE CICLO DI TABULAZIONI IN ASSENZA DI
      BUFFER')
    LIBRARY_LS.append('INSERIMENTO DATI')
    LIBRARY_LS.append(['numero di decimali dopo la virgola ai fini della stampa su
      file','maggiore o uguale a 1',''])
    LIBRARY_LS.append(['il numero totale di casi che si vogliono elaborare','intero positivo >0 e
      <%d'%(DATA_LS[0]+1),''])
    INPUT_LS.append([0,1,'I','>',0])
    INPUT_LS.append([0,1,'I','><',[0,DATA_LS[0]+1]])

elif TYPE_LS=='tab3_cyc':
    LIBRARY_LS.append([3,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('GENERAZIONE CICLO DI TABULAZIONI IN ASSENZA DI
      BUFFER')
    LIBRARY_LS.append('INSERIMENTO DATI')
    LIBRARY_LS.append(['il passo incrementale tra un ciclo e il successivo','intero positivo <
      %d'%(DATA_LS[0]+1),''])
    INPUT_LS.append([0,1,'I','><',[0,DATA_LS[0]+1]])

elif TYPE_LS=='buff_cyc':
    LIBRARY_LS.append([3,1])
    LIBRARY_LS.append('\n')
    LIBRARY_LS.append('INSERIRE DATI PER CREAZIONE CICLO ELABORAZIONE
      TABULATI CASI SPECIFICI CON BUFFER')
    LIBRARY_LS.append('Il ciclo verrà impostato incrementando la dimensione del buffer')
    LIBRARY_LS.append(['numero di decimali dopo la virgola ai fini della stampa su
      file','maggiore o uguale a 1',''])
    LIBRARY_LS.append(['il numero totale di casi che si vogliono elaborare','intero positivo
      maggiore o uguale a 1',''])
    LIBRARY_LS.append(['il passo incrementale tra un ciclo e il successivo','intero positivo
      maggiore o uguale a 1',''])
    LIBRARY_LS.append(['il buffer iniziale','intero positivo',''])
```

```
        INPUT_LS.append([0,1,'I','>',0])
        INPUT_LS.append([0,1,'I','>',0])
        INPUT_LS.append([0,1,'I','>',0])
        INPUT_LS.append([0,1,'I','>',0])  #codice scelta precisione

    elif TYPE_LS=='comp_tab_cyc':
        LIBRARY_LS.append([3,1])
        LIBRARY_LS.append('\n')
        LIBRARY_LS.append('INSERIRE DATI PER CREAZIONE CICLO COMPARAZIONI IN
            ASSENZA DI BUFFER')
        LIBRARY_LS.append('Il ciclo verrà impostato incrementando i tempi di interuscita dalle
            stazioni a monte')
        LIBRARY_LS.append(['numero di decimali dopo la virgola ai fini della stampa su
            file','maggiore o uguale a 1',''])
        LIBRARY_LS.append(['il numero totale di cicli di comparazione che si vuole
            implementare','intero positivo maggiore o uguale a 1',''])
        LIBRARY_LS.append(['il passo incrementale del tempo medio di uscita dalle stazioni a monte
            tra un ciclo e il successivo','numero reale >0',''])
        INPUT_LS.append([0,1,'I','>',0])
        INPUT_LS.append([0,1,'I','>',0])
        INPUT_LS.append([0,1,'F','>',0])

    elif TYPE_LS=='comp_buff_cyc':
        LIBRARY_LS.append([3,1])
        LIBRARY_LS.append('\n')
        LIBRARY_LS.append('INSERIRE DATI PER CREAZIONE CICLO COMPARAZIONI IN
            PRESENZA DI BUFFER')
        LIBRARY_LS.append('Il ciclo verrà impostato incrementando i tempi di interuscita dalle
            stazioni a monte')
        LIBRARY_LS.append(['numero di decimali dopo la virgola ai fini della stampa su
            file','maggiore o uguale a 1',''])
        LIBRARY_LS.append(['il numero totale di cicli di comparazione che si vuole
            implementare','intero positivo maggiore o uguale a 1',''])
        LIBRARY_LS.append(['il passo incrementale del tempo medio di uscita dalle stazioni a monte
            tra un ciclo e il successivo','numero reale >0',''])
        INPUT_LS.append([0,1,'I','>',0])
        INPUT_LS.append([0,1,'I','>',0])
        INPUT_LS.append([0,1,'F','>',0])
    else:
        input('ERRORE NEL RICHIAMO MENU INPUT, CODICE SBAGLIATO!!!!
            ARRESTARE!!!')
    #end if
    OUT_LS=[]
    OUT_LS.append(LIBRARY_LS)
    OUT_LS.append(INPUT_LS)
    return(OUT_LS)
#END def LIBRARY_SELECTOR():

def OUTPUT_FILTER(INP_OS):
#Funzione custom che ripulisce i dati ricevuti in input nella variabile INP_OS da tutte le variabili di
```

```
#controllo di stato interne.
#Viene richiamata dalla funzione custom INPUT_MENU.
   OUT_OS=[]
   for i in range(len(INP_OS)):
      OUT_OS.append(INP_OS[i][0])
   #end for i
   return(OUT_OS)
#end OUTPUT_FILTER

def TEST_FIELD(INPUT_TF):
#Funzione custom che verifica la validità, all'interno di un range prestabilito, dei dati ricevuti in
#input e che, in caso di esito positivo, li standardizzerà al fine del loro seguente utilizzo.
#Viene richiamata dalla funzione custom INPUT_MENU.
#Restituisce in output il codice indicante la validità o meno dei dati e, in caso di esito positivo, i dati
#rielaborati.
   try:
      if INPUT_TF[2]=='I':
         INPUT_TF[0]=int(INPUT_TF[0])
      elif INPUT_TF[2]=='F':
         INPUT_TF[0]=mp.mpf(INPUT_TF[0])
      elif INPUT_TF[2]=='S':
         INPUT_TF[0]=str(INPUT_TF[0])
      else:
         input('ERRORE NEL CODICE TIPO DI CAMPO TESTATO IN TEST_FILED')
      #end if
      if INPUT_TF[3]=='>' and INPUT_TF[0]>INPUT_TF[4]:
         INPUT_TF[1]=0
      elif INPUT_TF[3]=='<' and INPUT_TF[0]<INPUT_TF[4]:
         INPUT_TF[1]=0
      elif INPUT_TF[3]=='=' and INPUT_TF[0]==INPUT_TF[4]:
         INPUT_TF[1]=0
      elif INPUT_TF[3]=='not=' and not(INPUT_TF[0]==INPUT_TF[4]):
         INPUT_TF[1]=0
      elif INPUT_TF[3]=='><':
         if INPUT_TF[0]>INPUT_TF[4][0] and INPUT_TF[0]<INPUT_TF[4][1]:
            INPUT_TF[1]=0
         else:
            INPUT_TF[1]=1
         #end if
      elif INPUT_TF[3]=='not><':
         if not(INPUT_TF[0]>INPUT_TF[4][0] and INPUT_TF[0]<INPUT_TF[4][1]):
            INPUT_TF[1]=0
         else:
            INPUT_TF[1]=1
         #end if
      elif INPUT_TF[3]=='a':
         INPUT_TF[1]=1
         for i_test in range(len(INPUT_TF[4])):
            if INPUT_TF[0]==INPUT_TF[4][i_test]:
               INPUT_TF[1]=0
```

```
                break
            #end if
        #end for
    elif INPUT_TF[3]=='not_a':
        INPUT_TF[1]=0
        for i_test in range(len(INPUT_TF[4])):
            if INPUT_TF[0]==INPUT_TF[4][i_test]:
                INPUT_TF[1]=1
                break
            #end if
        #end for
    elif INPUT_TF[3]=='NULL':
        INPUT_TF[1]=0
    else:
        input('ERRORE NEL CODICE TIPO CONFRONTO TESTATO IN TEST_FILED')
    #end if
  except:
    INPUT_TF[1]=1
  #end try
  return(INPUT_TF[1])
#END def TEST_FIELD(INPUT_TF):

def INPUT_MENU(COD_IM,DATA_IM):
#Funzione custom che sovraintende tutti i menù di scelta e le schermate di immissione dati del
#programma.
#Verranno richiamate le funzioni custom  LIBRARY_SELECTOR per la scelta delle linee di testo e
#dei campi di input, TEST_FIELD per verificare che i dati immessi siano all'interno del range di
#validità e OUTPUT_FILTER per ripulire i dati dalle variabili di controllo interne al processo.
#Restituisce alla funzione chiamante l'input ricevuto.
  SERV_INP_IM=LIBRARY_SELECTOR(COD_IM,DATA_IM)
  LIBRARY_IM=SERV_INP_IM[0]
  INPUT_IM=SERV_INP_IM[1]
  ROW=len(INPUT_IM)

  for i_title in range(LIBRARY_IM[0][0]):
    print('%s'%(LIBRARY_IM[i_title+1]))
  #end for i_title
  print('\n')
  i_jump_IM=LIBRARY_IM[0][0]+1
  i_main_IM=0
  STATUS_INPUT=0
  while i_main_IM<ROW and STATUS_INPUT==0:
    for i_sub_IM in range(MENU_RANGE):
      print('\n')
      INPUT_IM[i_main_IM][0]=input('Inserire %s (%s) %s: '%
            (LIBRARY_IM[i_main_IM+i_jump_IM][0],
            LIBRARY_IM[i_main_IM+i_jump_IM][1],
            LIBRARY_IM[i_main_IM+i_jump_IM][2]))
      INPUT_IM[i_main_IM][1]=TEST_FIELD(INPUT_IM[i_main_IM])
      if INPUT_IM[i_main_IM][1]==1 and LIBRARY_IM[0][1]==1:
```

```
                STATUS_INPUT=1
                print('\n')
                print('Errore immissione %s'%(LIBRARY_IM[i_main_IM+i_jump_IM][0]))
                print('Si ricorda che deve essere %s'%(LIBRARY_IM[i_main_IM+i_jump_IM][1]))
                input('%d tentativi rimasti'%(MENU_RANGE-(i_sub_IM+1)))
            elif INPUT_IM[i_main_IM][1]==1 and LIBRARY_IM[0][1]==0:
                STATUS_INPUT=1
                print('\n')
                print('Errore immissione scelta')
                input('%d tentativi rimasti'%(MENU_RANGE-(i_sub_IM+1)))
            elif INPUT_IM[i_main_IM][1]==0:
                STATUS_INPUT=0
                break
            else:
                STATUS_INPUT=1
                input('PROBLEMA INTERNO A FUNZIONE INPUT_MENU, TEST FIELD DA
                    STRANI RISULTATI')
            #end if
        #end for i_main_menu
        i_main_IM+=1
    #end while
    OUT_IM=[]
    OUT_IM.append(STATUS_INPUT)
    if STATUS_INPUT==0:
        SERV_OUT_IM=OUTPUT_FILTER(INPUT_IM)
        for i_out in range(len(SERV_OUT_IM)):
            OUT_IM.append(SERV_OUT_IM[i_out])
        #end for
    #end if
    return(OUT_IM)
#END def INPUT_MENU(COD_IM)

def TEST_OPEN_FILE(F_NAME_TOF,TYPE_TOF):
#Funzione custom dedicata alla verifica della presenza o meno su hard drive della famiglia di files
#che si desidera aprire.
#Viene richiamata dalla funzione custom GENERAL_MATRIX_LOADING
    if TYPE_TOF==1:
        line_TOF=0
        start_range_TOF=2
        F_EXT_TOF=F_EXT
        print('Test presenza files matrici di base')
    elif TYPE_TOF==2:
        line_TOF=4
        start_range_TOF=2
        F_EXT_TOF=F_EXT
        print('Test presenza files tabulati timeless')
    elif TYPE_TOF==3:
        line_TOF=1
        start_range_TOF=3
        F_EXT_TOF=F_EXT
```

```
        print('Test presenza files tabulati caso specifico')
    elif TYPE_TOF==4:
        line_TOF=3
        start_range_TOF=1
        F_EXT_TOF=F_EXT_S
        print('Test presenza files schedulazione')
    elif TYPE_TOF==5:
        range_TOF=1
        print('Apertura files mappatura produttoria')
    else:
        input('ERRORE CHIAMATA TEST_OPEN_FILE, CODICE CHIAMATA ERRATO!!!')
    #end if
    if not(TYPE_TOF==5):
        range_TOF=len(F_BASE_NAME[line_TOF])-start_range_TOF
    #end if
    for i_test in range(range_TOF):
        try:
            if TYPE_TOF==4:
                print('test open for %s.%s'%((F_NAME_TOF,F_EXT_TOF)))
                f=open('%s.%s'%(F_NAME_TOF,F_EXT_TOF),"r")
            elif TYPE_TOF==5:
                print('opening %s'%((F_NAME_TOF)))
                f=open('%s'%(F_NAME_TOF),"r")
            else:
                print('test open for %s%s.%s'%(F_NAME_TOF,F_BASE_NAME[line_TOF]
                      [start_range_TOF+i_test],F_EXT_TOF))
                f=open('%s%s.%s'%(F_NAME_TOF,F_BASE_NAME[line_TOF]
                      [start_range_TOF+i_test],F_EXT_TOF),"r")
            #end if
            STATUS_OPEN_OUT=0
            f.close()
        except:
            STATUS_OPEN_OUT=1
            break
        #end try
    #end for i_test
    return(STATUS_OPEN_OUT)
#END def TEST_OPEN_FILE




def OPEN_FILE(F_NAME_OF,TYPE_OF):
#Funzione custom designata al caricamento in memoria dei dati contenuti in determinate famiglie di
#files.
#Viene richiamata dalla funzione custom GENERAL_MATRIX_LOADING
    if TYPE_OF==1:
        line_OF=0
        start_range_OF=2
        F_EXT_OF=F_EXT
```

```
    print('Apertura files matrici di base')
elif TYPE_OF==2:
    line_OF=4
    start_range_OF=2
    F_EXT_OF=F_EXT
    print('Apertura files tabulati timeless')
elif TYPE_OF==3:
    line_OF=1
    start_range_OF=3
    F_EXT_OF=F_EXT
    print('Apertura files tabulati caso specifico')
elif TYPE_OF==4:
    line_OF=3
    start_range_OF=1
    F_EXT_OF=F_EXT_S
    print('Apertura files schedulazione')
elif TYPE_OF==5:
    range_OF=1
    print('Apertura files mappatura produttoria')
else:
    input('ERRORE CHIAMATA TEST_OPEN_FILE, CODICE CHIAMATA ERRATO!!!')
#end if
if not(TYPE_OF==5):
    range_OF=len(F_BASE_NAME[line_OF])-start_range_OF
#end if
OUT_OF=[]
print('Caricamento in memoria in corso')
for i_open_OF in range(range_OF):
    if TYPE_OF==4:
        print('opening %s.%s'%((F_NAME_OF,F_EXT_OF)))
        f=open('%s.%s'%(F_NAME_OF,F_EXT_OF),"r")
    elif TYPE_OF==5:
        print('opening %s'%((F_NAME_OF)))
        f=open('%s'%(F_NAME_OF),"r")
    else:
        print('opening %s%s.%s'%(F_NAME_OF,F_BASE_NAME[line_OF]
                [start_range_OF+i_open_OF],F_EXT_OF))
        f=open('%s%s.%s'%(F_NAME_OF,F_BASE_NAME[line_OF][
                start_range_OF+i_open_OF],F_EXT_OF),"r")
    #end if
    SERV_FILE=f.readlines()

    if not(TYPE_OF==4):
        FILE_KEY=[]
        FILE_KEY.append(SERV_FILE[0][0])
        i_pointer=0
        i_word=0
        while i_word<3:
            i_pointer+=1
            DELIMITER_CTRL=0
```

```
            integer_count=0
            while DELIMITER_CTRL==0:
               if not(SERV_FILE[0][i_pointer]==FILE_KEY[0]):
                  integer_count+=1
                  i_pointer+=1
               else:
                  DELIMITER_CTRL=1
               #end if
            #end while
            SERV=0
            for i_calc_OF in range(integer_count):
               SERV+=int(SERV_FILE[0][(i_pointer-1)-i_calc_OF])*int(mp.power(10,i_calc_OF))
            #end for i_calc
            FILE_KEY.append(SERV) #FILE_KEY[1] saranno le linee,FILE_KEY[1] saranno le
                  colonne, FILE_KEY[1] sarà la precisione
            i_word+=1
         #end while

         MTX_OUT=mp.zeros(FILE_KEY[1],FILE_KEY[2])
         for i_row in range(FILE_KEY[1]):
            j_start=0
            for j_col in range(FILE_KEY[2]):
               j_pointer=j_start
               while not(SERV_FILE[i_row+1][j_pointer]==FILE_KEY[0]):
                  j_pointer+=1
               #end while
               MTX_OUT[i_row,j_col]=mp.mpf(SERV_FILE[i_row+1][j_start:j_pointer])
               j_start=j_pointer+1
            #end for j_col
         #end for i_row
      else:
         MTX_OUT=SERV_FILE
      #end if
      f.close()
      OUT_OF.append(MTX_OUT)
   #end for i_open
   return(OUT_OF)
#END def OPEN_FILE()

def GENERAL_MATRIX_LOADING(WHO_CALLS,DATA_GML):
#Funzione custom che sovraintende tutte le operazioni richieste all'apertura e al caricamento in
#memoria di files.
#I parametri interni di funzionamento saranno generati in base alle variabili WHO_CALLS e
#DATA_GML.
#Se richiamata da una delle funzioni in modalità utente, richiederà l'input necessario tramite la
#funzione custom INPUT_MENU.
#Utilizzerà le funzioni custom TEST_OPEN_FILE per controllare la presenza dei files richiesti e
#OPEN_FILE per la loro lettura e caricamento in memoria.
#Restituisce in output i dati caricati.
   OUT_GML=[]
```

```
if WHO_CALLS=='mtx':
   INPUT_MTX_GML=INPUT_MENU('mtx_inp',[0])
elif WHO_CALLS=='timeless':
   INPUT_MTX_GML=INPUT_MENU('timeless_inp',[0])
elif WHO_CALLS=='tabs':
   INPUT_MTX_GML=INPUT_MENU('tab_inp',[0])
elif WHO_CALLS=='prod':
   INPUT_MTX_GML=[]
   INPUT_MTX_GML.append(0)
   INPUT_MTX_GML.append(DATA_GML[0])
elif WHO_CALLS=='sched' or WHO_CALLS=='comp':
   INPUT_MTX_GML=[]
   INPUT_MTX_GML.append(0)
   for i_input in range(len(DATA_GML)):
      INPUT_MTX_GML.append(DATA_GML[i_input])
   #end for i_input
else:
   input('PROBLEMA AL PRIMO SELETTORE DI GENERAL_MATRIX_LOADING')
#end if
if INPUT_MTX_GML[0]==1:
   OUT_GML.append(-1)
   print('\n')
   print('Limite errori di immissione dati raggiunto')
elif INPUT_MTX_GML[0]==0:
   if WHO_CALLS=='mtx' or (WHO_CALLS=='sched' and len(DATA_GML)==2):
      INPUT_MTX_GML[0]=1
      F_COD_GML='%s%d%s%d'%(F_BASE_NAME[0]
            [0],int(INPUT_MTX_GML[1]),F_BASE_NAME[0][1],
            int(INPUT_MTX_GML[2]))
      TYPE_GML=1
   elif WHO_CALLS=='timeless' or ((WHO_CALLS=='sched' or WHO_CALLS=='comp') and
      len(DATA_GML)==3):
      INPUT_MTX_GML[0]=2
      F_COD_GML='%s%d%s%d'%(F_BASE_NAME[4][0],
            int(INPUT_MTX_GML[1]),F_BASE_NAME[4][1], int(INPUT_MTX_GML[2]))
      TYPE_GML=2
   elif WHO_CALLS=='tabs' or ((WHO_CALLS=='sched' or WHO_CALLS=='comp') and
            len(DATA_GML)==4):
      INPUT_MTX_GML[0]=3
      INPUT_MTX_GML[2]=mp.mpf(INPUT_MTX_GML[2])
      if INPUT_MTX_GML[2]-int(INPUT_MTX_GML[2])==0:
         INPUT_MTX_GML[2]='%d.0'%(INPUT_MTX_GML[2])
      #end if
      if INPUT_MTX_GML[3]=='s' or INPUT_MTX_GML[3]=='sec':
         INPUT_MTX_GML[3]='sec'
      elif INPUT_MTX_GML[3]=='m' or INPUT_MTX_GML[3]=='min':
         INPUT_MTX_GML[3]='min'
      elif COD_CNG[3]=='h' or INPUT_MTX_GML[3]=='h':
         INPUT_MTX_GML[3]='h'
      else:
```

```
            input('ERRORE IN COD_NAME_GENERATOR, UNITA DI MISURA TEMPI
                ERRATE')
        #end if
        F_COD_GML='%s%d%s%s%s%s%d'%(F_BASE_NAME[1][0],
                INPUT_MTX_GML[1],F_BASE_NAME[1][1],
                INPUT_MTX_GML[2],INPUT_MTX_GML[3],F_BASE_NAME[1][2],
                INPUT_MTX_GML[4])
        TYPE_GML=3
    elif (WHO_CALLS=='sched' or WHO_CALLS=='comp') and len(DATA_GML)==1:
        INPUT_MTX_GML[0]=4
        F_COD_GML='%s%s%s'%(F_BASE_NAME[3][0],
                INPUT_MTX_GML[1],F_BASE_NAME[3][1])
        TYPE_GML=4
    elif WHO_CALLS=='prod':
        INPUT_MTX_GML[0]=5
        F_COD_GML=DATA_GML[0]
        TYPE_GML=5
    else:
        input('PROBLEMA AL SECONDO SELETTORE DI GENERAL_MATRIX_LOADING')
    #end if
    if not(WHO_CALLS=='prod'):
        print('Codice famiglia files: %s'%(F_COD_GML))
    #end if
    INPUT_MTX_GML.append(F_DELIMITER)

    ERROR_CTRL_OPENFILE_GML=TEST_OPEN_FILE(F_COD_GML, TYPE_GML)
    if ERROR_CTRL_OPENFILE_GML==1:
        print('\n')
        print('Errore in apertura files, uno o più files corrispondenti al numero identificativo inserito
                non presenti')
        OUT_GML.append(-1)
    elif ERROR_CTRL_OPENFILE_GML==0:
        print('\n')
        print('I files richiesti sono presenti')
        SERV_GML=OPEN_FILE(F_COD_GML,TYPE_GML)
        OUT_GML.append(1)
        if WHO_CALLS=='tabs' or ((WHO_CALLS=='sched' or WHO_CALLS=='comp') and
                len(DATA_GML)==4):
            GEN_DATA_GML=[]
            GEN_DATA_GML.append(INPUT_MTX_GML[1])
            GEN_DATA_GML.append(INPUT_MTX_GML[2])
            GEN_DATA_GML.append(INPUT_MTX_GML[3])
            OUT_GML.append(GEN_DATA_GML)
        #end if
        for i_out in range(len(SERV_GML)):
            OUT_GML.append(SERV_GML[i_out])
        #end for
        print('Caricamento dati terminato')
    #end if
#end if
```

```
    if not(WHO_CALLS=='sched' or WHO_CALLS=='comp'):
        input('Reindirizzamento a menù principale')
    #end if
    return(OUT_GML)
#END def GENERAL_MATRIX_LOADING()

def LIBRARY_PRINT(TYPE_LP):
#Funzione custom dedicata all'assegnazione delle linee di testo contenute nei files di output della
#sezione comparazione.
#Viene richiamata dalla funzione custom PRINT_FINAL_COMP.
    LIBRARY_LP=[]
    if TYPE_LP=='comp_tabs_ta':
        LIBRARY_LP.append([1,4])
        LIBRARY_LP.append('TEMPI DI ARRIVO')
        LIBRARY_LP.append('CONFRONTO TRA MEDIA DEI TEMPI MEDI DI INGRESSO
            TEORICI E TEMPO MEDIO SIMULATO')
        LIBRARY_LP.append('CONFRONTO TRA TEMPI MEDI DI INGRESSO TEORICI DA
            FUNZIONE GLOBALE E SIMULATI')
        LIBRARY_LP.append('CONFRONTO TRA MEDIA DELLE VARIANZE DEI TEMPI DI
            INGRESSO TEORICI E VARIANZA SIMULATA')
        LIBRARY_LP.append('CONFRONTO TRA VARIANZA DEI TEMPI DI INGRESSO
            TEORICI GLOBALI E VARIANZA SIMULATA')
        LIBRARY_LP.append(['Dimensione lotto','Tempi medi di ingresso da calcoli teorici','Tempi
            medi di ingresso da simulatore','Differenza percentuale rispetto ai dati simulatore'])
        LIBRARY_LP.append(['Dimensione lotto','Tempi medi di ingresso da calcoli teorici','Tempi
            medi di ingresso da simulatore','Differenza percentuale rispetto ai dati simulatore'])
        LIBRARY_LP.append(['Dimensione lotto','Varianza tempi di ingresso da calcoli
            teorici','Varianza tempi di ingresso da simulatore','Differenza percentuale rispetto ai dati
            simulatore'])
        LIBRARY_LP.append(['Dimensione lotto','Varianza tempi di ingresso da calcoli
            teorici','Varianza tempi di ingresso da simulatore','Differenza percentuale rispetto ai dati
            simulatore'])

    elif TYPE_LP=='comp_tabs_tw':
        LIBRARY_LP.append([1,5])
        LIBRARY_LP.append('TEMPI DI ATTESA')
        LIBRARY_LP.append('CONFRONTO TRA MEDIA DEI TEMPI MEDI DI ATTESA
            TEORICI E TEMPO MEDIO SIMULATO')
        LIBRARY_LP.append('CONFRONTO TRA TEMPI MEDI DI ATTESA TEORICI (ricavati
            con legge di Little usando WIP medio da tabella wip e TH teorico di uscita) E TEMPO
            MEDIO SIMULATO')
        LIBRARY_LP.append('CONFRONTO TRA TEMPI MEDI DI ATTESA TEORICI DA
            FUNZIONE GLOBALE E TEMPO MEDIO SIMULATO')
        LIBRARY_LP.append('CONFRONTO TRA MEDIA DELLE VARIANZE DEI TEMPI DI
            ATTESA TEORICI E VARIANZA SIMULATA')
        LIBRARY_LP.append('CONFRONTO TRA VARIANZA DEI TEMPI DI ATTESA TEORICI
            GLOBALI E VARIANZA SIMULATA')
        LIBRARY_LP.append(['Dimensione lotto','Tempi medi di attesa da calcoli teorici','Tempi medi
            di attesa da simulatore','Differenza percentuale rispetto ai dati simulatore'])
        LIBRARY_LP.append(['Dimensione lotto','Tempi medi di attesa da calcoli teorici','Tempi medi
```

```
        di attesa da simulatore','Differenza percentuale rispetto ai dati simulatore'])
    LIBRARY_LP.append(['Dimensione lotto','Tempi medi di attesa da calcoli teorici','Tempi medi
        di attesa da simulatore','Differenza percentuale rispetto ai dati simulatore'])
    LIBRARY_LP.append(['Dimensione lotto','Varianza tempi di attesa da calcoli teorici','Varianza
        tempi di attesa da simulatore','Differenza percentuale rispetto ai dati simulatore'])
    LIBRARY_LP.append(['Dimensione lotto','Varianza tempi di attesa da calcoli teorici','Varianza
        tempi di attesa da simulatore','Differenza percentuale rispetto ai dati simulatore'])

elif TYPE_LP=='comp_tabs_tu':
    LIBRARY_LP.append([1,5])
    LIBRARY_LP.append('TEMPI DI USCITA')
    LIBRARY_LP.append('CONFRONTO TRA MEDIA DEI TEMPI MEDI DI USCITA
        TEORICI E SIMULATI')
    LIBRARY_LP.append('CONFRONTO TRA TEMPI MEDI DI USCITA TEORICI (ricavati
        con legge di Little usando WIP medio da tabella wip e CT pari a tempo attesa medio) E
        SIMULATI')
    LIBRARY_LP.append('CONFRONTO TRA TEMPI MEDI DI USCITA TEORICI DA
        FUNZIONE GLOBALE E SIMULATI')
    LIBRARY_LP.append('CONFRONTO TRA MEDIA DELLE VARIANZE DEI TEMPI DI
        USCITA TEORICI E VARIANZA SIMULATA')
    LIBRARY_LP.append('CONFRONTO TRA VARIANZA DEI TEMPI DI USCITA TEORICI
        GLOBALI E VARIANZA SIMULATA')
    LIBRARY_LP.append(['Dimensione lotto','Tempi medi di uscita da calcoli teorici','Tempi medi
        di uscita da simulatore','Differenza percentuale rispetto ai dati simulatore'])
    LIBRARY_LP.append(['Dimensione lotto','Tempi medi di uscita da calcoli teorici','Tempi medi
        di uscita da simulatore','Differenza percentuale rispetto ai dati simulatore'])
    LIBRARY_LP.append(['Dimensione lotto','Tempi medi di uscita da calcoli teorici','Tempi medi
        di uscita da simulatore','Differenza percentuale rispetto ai dati simulatore'])
    LIBRARY_LP.append(['Dimensione lotto','Varianza tempi di uscita da calcoli teorici','Varianza
        tempi di uscita da simulatore','Differenza percentuale rispetto ai dati simulatore'])
    LIBRARY_LP.append(['Dimensione lotto','Varianza tempi di uscita da calcoli teorici','Varianza
        tempi di uscita da simulatore','Differenza percentuale rispetto ai dati simulatore'])

elif TYPE_LP=='comp_tabs_wip':
    LIBRARY_LP.append([1,4])
    LIBRARY_LP.append('LIVELLI DI WIP')
    LIBRARY_LP.append('CONFRONTO TRA LIVELLI DI WIP MEDI TEORICI E
        SIMULATI')
    LIBRARY_LP.append('CONFRONTO TRA LIVELLI DI WIP MEDI TEORICI (ricavati con
        legge di Little usando TH teorico di ingresso e CT pari a tempo attesa medio) E
        SIMULATI')
    LIBRARY_LP.append('CONFRONTO TRA LIVELLI DI WIP MEDI TEORICI (ricavati con
        legge di Little usando TH teorico di uscita e CT pari a tempo attesa medio) E SIMULATI')
    LIBRARY_LP.append('CONFRONTO TRA LIVELLI DI WIP MEDI TEORICI (ricavati con
        legge di Little usando TH in uscita e CT ricavati da funzioni di densità globali) E
        SIMULATI')
    LIBRARY_LP.append(['Dimensione lotto','Livello di WIP medio da calcoli teorici','Livello di
        WIP medio da simulatore','Differenza percentuale rispetto ai dati simulatore'])
    LIBRARY_LP.append(['Dimensione lotto','Livello di WIP medio da calcoli teorici','Livello di
        WIP medio da simulatore','Differenza percentuale rispetto ai dati simulatore'])
```

```
            LIBRARY_LP.append(['Dimensione lotto','Livello di WIP medio da calcoli teorici','Livello di
              WIP medio da simulatore','Differenza percentuale rispetto ai dati simulatore'])
            LIBRARY_LP.append(['Dimensione lotto','Livello di WIP medio da calcoli teorici','Livello di
              WIP medio da simulatore','Differenza percentuale rispetto ai dati simulatore'])

        elif TYPE_LP=='comp_buff_tw':
            LIBRARY_LP.append([1,1])
            LIBRARY_LP.append('TEMPI DI ATTESA')
            LIBRARY_LP.append('CONFRONTO TRA TEMPI MEDI DI ATTESA TEORICI E
              SIMULATI')
            LIBRARY_LP.append(['Dimensione buffer','Tempi medi di attesa da calcoli teorici','Tempi
              medi di attesa da simulatore','Differenza percentuale rispetto ai dati simulatore'])

        elif TYPE_LP=='comp_buff_tu':
            LIBRARY_LP.append([1,1])
            LIBRARY_LP.append('TEMPI DI USCITA')
            LIBRARY_LP.append('CONFRONTO TRA TEMPI MEDI DI USCITA TEORICI E
              SIMULATI')
            LIBRARY_LP.append(['Dimensione buffer','Tempi medi di uscita da calcoli teorici','Tempi
              medi di uscita da simulatore','Differenza percentuale rispetto ai dati simulatore'])

        elif TYPE_LP=='comp_buff_wip':
            LIBRARY_LP.append([1,2])
            LIBRARY_LP.append('LIVELLI DI WIP')
            LIBRARY_LP.append('CONFRONTO TRA LIVELLI DI WIP MEDI TEORICI E
              SIMULATI')
            LIBRARY_LP.append('CONFRONTO TRA
              NUMERO MEDIO DI INGRESSI TEORICI E SIMULATI PER SATURARE IL BUFFER
              ')
            LIBRARY_LP.append(['Dimensione buffer','Livello di WIP medio da calcoli teorici','Livello di
              WIP medio da simulatore','Differenza percentuale rispetto ai dati simulatore'])
            LIBRARY_LP.append(['Dimensione buffer','Numero di ingressi a saturazione da calcoli
              teorici','Numero di ingressi a saturazione da simulatore','Differenza percentuale rispetto ai
              dati simulatore'])
        return(LIBRARY_LP)
    #end def LIBRARY PRINT

    def EXCEL_CONVERT(LINE_EC):
    #Funzione di conversione dei dati che verranno salvati su files in un formato compatibile con
    #EXCEL.
    #Viene richiamata da tutte le funzioni custom adibite alla creazione e salvataggio di files destinati
    #alla visualizzazione da parte di utenti.
        OUT=''
        for i_field in range(len(LINE_EC)):
            j_letter=0
            while j_letter<len(LINE_EC[i_field]):
                if not(LINE_EC[i_field][j_letter]==F_DELIMITER or LINE_EC[i_field][j_letter]=='.' or
                        LINE_EC[i_field][j_letter]=='e'):
                    OUT+=LINE_EC[i_field][j_letter]
                    j_letter+=1
```

```
            elif LINE_EC[i_field][j_letter]==F_DELIMITER:
               OUT+='.'
               j_letter+=1
            elif LINE_EC[i_field][j_letter]=='.':
               OUT+=F_DELIMITER
               j_letter+=1
            elif LINE_EC[i_field][j_letter]=='e':
               if not(j_letter+1==len(LINE_EC[i_field][j_letter])) and (LINE_EC[i_field]
                     [j_letter+1]=='+' or LINE_EC[i_field][j_letter+1]=='-'):
                  count=0
                  j_count=j_letter+2
                  while not(LINE_EC[i_field][j_count]==F_DELIMITER or LINE_EC[i_field]
                              [j_count]==' '):
                     count+=1
                     j_count+=1
                  #end while
                  if count<3:
                     if count==1:
                        OUT+='E%s00'%(LINE_EC[i_field][j_letter+1])
                     elif count==2:
                        OUT+='E%s0'%(LINE_EC[i_field][j_letter+1])
                     #end if
                  else:
                     OUT+='E%s'%(LINE_EC[i_field][j_letter+1])
                  #end if
                  j_letter+=2
               else:
                  OUT+=LINE_EC[i_field][j_letter]
                  j_letter+=1
               #end if
            #end if
         #end while j_letter
      #end for i_field
      return(OUT)
#end def EXCEL_CONVERT

def PRINT_FINAL_COMP(WHO_CALLS, NAME_PFC, DATA_PRINT_PFC):
#Funzione custom di stampa su files dei dati relativi alla comparazione tra i parametri frutto di
#modellizzazione e di simulazione di un caso specifico in assenza o in presenza di buffer, in
#formato utente.
#Viene richiamata dalla funzione custom  COMP_BFV e COMP_TABS.
#Richiama le funzioni custom LIBRARY_PRINT, per selezionare le librerie di stampa necessarie, e
#EXCEL_CONVERT per convertire i dati in un formato compatibile con EXCEL.
   mp.pretty=True
   if WHO_CALLS=='comp_tabs':
      name_line=0
   elif WHO_CALLS=='comp_buff':
      name_line=1
   #end if
   for i_file in range(len(DATA_PRINT_PFC)):
```

```
        f=open('%s%s%s.%s'%(F_NAME_COMP[name_line][0], NAME_PFC,
          F_NAME_COMP[name_line][i_file+1],F_EXT),'w+')
        SERV_TEXT=LIBRARY_PRINT(DATA_PRINT_PFC[i_file][0])
        SERV_MTX=DATA_PRINT_PFC[i_file][len(DATA_PRINT_PFC[i_file])-1]
        SERV_LINE=[]
        for j_title in range(SERV_TEXT[0][0]):
          SERV_LINE.append(SERV_TEXT[j_title+1])
          SERV_LINE.append('\n')
        #end for j_title
        for j_block in range(SERV_TEXT[0][1]):
          SERV_LINE.append(SERV_TEXT[j_block+1+SERV_TEXT[0][0]])
          SERV_LINE.append('\n')

          mtx_line=DATA_PRINT_PFC[i_file][1+j_block]
          jump=1+SERV_TEXT[0][0]+SERV_TEXT[0][1]+j_block
          for k_line in range(len(SERV_TEXT[jump])):
            SERV_LINE.append('%s%s'%(SERV_TEXT[jump][k_line],F_DELIMITER))
            line=mtx_line[k_line]
            for w_col in range(SERV_MTX.cols):
              SERV_LINE.append('%s%s'%(mp.nstr(SERV_MTX[line,w_col],5),F_DELIMITER))
            #end for w_col
            SERV_LINE.append('\n')
          #end for k_line
          SERV_LINE.append('\n\n')
        #end for j_block
        f.write('%s'%(EXCEL_CONVERT(SERV_LINE)))
        f.close()
      #end for i_file
      mp.pretty=False
      return()
#end PRINT_FINAL_COMP()

def PRINT_TO_SINGLE_FINAL_FILE(WHO_CALLS,NAME,EXT,MTX,MTX2,DELIMITER):
#Funzione custom dedicata alla creazione e alla stampa dei files contenenti i risultati in formato
#utente delle simulazioni singole in assenza o presenza di buffer.
#Viene richiamata dalla funzione custom SINGLE_SIM.
#I dati contenuti sono rielaborati dalla funzione custom EXCEL_CONVERT per essere compatibili
#con EXCEL.
      mp.pretty=True
      f=open("%s.%s" %(NAME,EXT),"w+")
      SERV_LINE=[]
      SERV_LINE.append("PARAMETRI SIMULAZIONE%s\n\n"%(F_DELIMITER))
      SERV_LINE.append("Numero di linee a monte: %s"%(F_DELIMITER))
      SERV_LINE.append("%d%s\n"%(MTX2[1],F_DELIMITER))
      SERV_LINE.append("Tempo medio di interuscita linee a monte:%s"%(F_DELIMITER))
      SERV_LINE.append("%s %s%s\n"%(mp.nstr(MTX2[2],MTX2[4]),MTX2[3],F_DELIMITER))
      SERV_LINE.append("Quantità pacchetti da accoppiare: %s"%(F_DELIMITER))
      SERV_LINE.append("%d%s\n"%(MTX2[0],F_DELIMITER))
      if WHO_CALLS=='single_buff':
        SERV_LINE.append("Dimensione buffer: %s"%(F_DELIMITER))
```

```
        SERV_LINE.append("%d%s\n"%(MTX2[5],F_DELIMITER))
    #end if
    SERV_LINE.append("\n\n\n")
    SERV_LINE.append("PARAMETRI CALCOLATI%s\n\n"%(F_DELIMITER))
    SERV_LINE.append("INTERTEMPO MEDIO DI ARRIVO:%s"%(DELIMITER))
    SERV_LINE.append("%s%s" %(mp.nstr(MTX[0],MTX2[4]),DELIMITER))
    SERV_LINE.append("\n")
    SERV_LINE.append("VARIANZA INTERTEMPI MEDI IN INGRESSO:%s"%(DELIMITER))

    SERV_LINE.append("%s%s" %(mp.nstr(MTX[4],MTX2[4]),DELIMITER))
    SERV_LINE.append("\n\n")
    SERV_LINE.append("TH MEDIO IN INGRESSO:%s"%(DELIMITER))
    SERV_LINE.append("%s%s" %(mp.nstr(1/MTX[0],MTX2[4]),DELIMITER))
    SERV_LINE.append("\n\n")
    SERV_LINE.append("TEMPO MEDIO DI ATTESA:%s"%(DELIMITER))
    SERV_LINE.append("%s%s" %(mp.nstr(MTX[1],MTX2[4]),DELIMITER))
    SERV_LINE.append("\n")
    SERV_LINE.append("VARIANZA TEMPI MEDI DI ATTESA:%s"%(DELIMITER))
    SERV_LINE.append("%s%s" %(mp.nstr(MTX[5],MTX2[4]),DELIMITER))
    SERV_LINE.append("\n")
    SERV_LINE.append("INTERTEMPO MEDIO DI USCITA:%s"%(DELIMITER))
    SERV_LINE.append("%s%s"%(mp.nstr(MTX[2],MTX2[4]),DELIMITER))
    SERV_LINE.append("\n")
    SERV_LINE.append("VARIANZA INTERTEMPI MEDI DI USCITA:%s"%(DELIMITER))
    SERV_LINE.append("%s%s"%(mp.nstr(MTX[6],MTX2[4]),DELIMITER))
    SERV_LINE.append("\n")
    SERV_LINE.append("TH MEDIO DI USCITA:%s"%(DELIMITER))
    SERV_LINE.append("%s%s"%(mp.nstr(1/MTX[2],MTX2[4]),DELIMITER))
    SERV_LINE.append("\n\n")
    SERV_LINE.append("LIVELLO MEDIO DI WIP:%s"%(DELIMITER))
    SERV_LINE.append("%s%s" %(mp.nstr(MTX[3],MTX2[4]),DELIMITER))
    SERV_LINE.append("\n")
    SERV_LINE.append("VARIANZA LIVELLI DI WIP:%s"%(DELIMITER))
    SERV_LINE.append("%s%s" %(mp.nstr(MTX[7],MTX2[4]),DELIMITER))
    SERV_LINE.append("\n")
    if WHO_CALLS=='single_buff':
        SERV_LINE.append("NUMERO MEDIO DI COPPIE ALLA SATURAZIONE BUFFER:
          %s"%(DELIMITER))
        SERV_LINE.append("%s%s" %(mp.nstr(MTX[8],MTX2[4]),DELIMITER))
    #end if
    SERV_LINE.append("\n\n")
    f.write('%s'%(EXCEL_CONVERT(SERV_LINE)))
    f.close()    #comando di chiusura del file precedentemente aperto
    mp.pretty=False
    return()
#end def PRINT_TO_SINGLE_FINAL_FILE

def PRINT_TO_SINGLE_ELABORATION_FILE
(NAME_INT,EXT_INT,F_PREC_INT,F_DELIMITER_INT,TABLE_TEXT_INT,DESCRIPTION_
INT,MTX):
```

```
#Funzione custom adibita alla creazione e salvataggio dei files di output delle funzioni custom
#CASAUL_EXTRACTION, CALC_EVERYTHING_SNB, CALC_EVERYTHING_SBUFF e
#TIMELINE_TO_WIP.
#I dati contenuti sono rielaborati dalla funzione custom EXCEL_CONVERT per essere compatibili
#con EXCEL.
   COL=MTX.cols
   ROW=MTX.rows
   mp.pretty=True
   f=open("%s.%s" %(NAME_INT,EXT_INT),"w+")
   j_start=-10
   j_prnt=0
   SERV_LINE=[]
   while j_start+j_prnt<COL:
      j_start+=10
      j_prnt=0
      SERV_LINE.append("%s%s"%(TABLE_TEXT_INT,F_DELIMITER_INT))
      print("%s%s"%(TABLE_TEXT_INT,F_DELIMITER_INT))
      while j_prnt<10 and j_start+j_prnt<COL:
         SERV_LINE.append("%d%s"%((j_start+j_prnt+1),F_DELIMITER_INT))
         print("%d%s"%((j_start+j_prnt+1),F_DELIMITER_INT))
         j_prnt+=1
      #end while j_prnt
      SERV_LINE.append("\n")
      i_prnt=0
      while i_prnt<ROW:
         SERV_LINE.append("%s%s"%(DESCRIPTION_INT[i_prnt],F_DELIMITER_INT))
         print("%s%s"%(DESCRIPTION_INT[i_prnt],F_DELIMITER_INT))
         j_prnt=0
         while j_prnt<10 and j_start+j_prnt<COL:
            SERV_LINE.append("%s%s"%(mp.nstr(MTX[i_prnt,
                  (j_start+j_prnt)],F_PREC_INT),F_DELIMITER_INT))
            print("%s%s"%(mp.nstr(MTX[i_prnt,
                  (j_start+j_prnt)],F_PREC_INT),F_DELIMITER_INT))
            j_prnt+=1
         #end while j_prnt
         SERV_LINE.append("\n")
         print("\n")
         i_prnt+=1
      #end while i_prnt
      SERV_LINE.append("\n")
      print("\n")
   #end while
   f.write('%s'%(EXCEL_CONVERT(SERV_LINE)))
   f.close()
   mp.pretty=False
   return()
#end def PRINT_TO_SINGLE_ELABORATION_FILE

def PRINT_TO_MULTI_FINAL_FILE
(WHO_CALLS,NAME_INT,EXT_INT,F_DELIMITER_INT,TABLE_TEXT_INT,
```

```
DESCRIPTION_INT,MTX,ARRAY,DATA):
#Funzione custom designata alla creazione dei files di output contenenti i dati elaborati dai
#simulatori a blocchi di simulazioni in presenza o assenza di buffer
#Viene richiamata dalla funzione custom MULTI_SIM.
#I dati contenuti sono rielaborati dalla funzione custom EXCEL_CONVERT per essere compatibili
#con EXCEL.
   COL=MTX.cols
   ROW=MTX.rows
   mp.pretty=True
   f=open("%s.%s" %(NAME_INT,EXT_INT),"w+")
   SERV_LINE=[]
   SERV_LINE.append("PARAMETRI SIMULAZIONE%s\n"%(F_DELIMITER))
   SERV_LINE.append("Numero di simulazioni: %s"%(F_DELIMITER))
   SERV_LINE.append("%d%s\n"%(DATA[5],F_DELIMITER))
   SERV_LINE.append("Numero di linee a monte: %s"%(F_DELIMITER))
   SERV_LINE.append("%d%s\n"%(DATA[0],F_DELIMITER))
   SERV_LINE.append("Tempo medio di interuscita linee a monte:%s"%(F_DELIMITER))
   SERV_LINE.append("%s%s\n"%(mp.nstr(DATA[1],DATA[4]),F_DELIMITER))
   SERV_LINE.append("Quantità pacchetti da accoppiare: %s"%(F_DELIMITER))
   SERV_LINE.append("%d%s\n"%(DATA[3],F_DELIMITER))
   if WHO_CALLS=='multi_buff':
      SERV_LINE.append("Dimensione buffer: %s"%(F_DELIMITER))
      SERV_LINE.append("%d%s\n"%(DATA[6],F_DELIMITER))
   #end if
   SERV_LINE.append("\n\n")
   SERV_LINE.append("PARAMETRI CALCOLATI%s\n"%(F_DELIMITER))
   SERV_LINE.append("Tempo medio interarrivo su %d simulazioni: %s"%
         (DATA[5],F_DELIMITER))
   SERV_LINE.append("%s%s\n"%(mp.nstr(ARRAY[0],DATA[4]),F_DELIMITER))
   SERV_LINE.append("Media delle varianze
         dei tempi medi di interarrivo su %d simulazioni: %s"%(DATA[5],F_DELIMITER))
   SERV_LINE.append("%s%s\n"%(mp.nstr(ARRAY[1],DATA[4]),F_DELIMITER))
   SERV_LINE.append("TH medio arrivo su %d simulazioni: %s"%(DATA[5],F_DELIMITER))
   SERV_LINE.append("%s%s\n"%(mp.nstr(1/ARRAY[0],DATA[4]),F_DELIMITER))
   SERV_LINE.append("Tempo medio di attesa su %d simulazioni: %s"%
         (DATA[5],F_DELIMITER))
   SERV_LINE.append("%s%s\n"%(mp.nstr(ARRAY[2],DATA[4]),F_DELIMITER))
   SERV_LINE.append("Media delle varianze dei tempi medi di attesa su %d simulazioni: %s"%
         (DATA[5],F_DELIMITER))
   SERV_LINE.append("%s%s\n"%(mp.nstr(ARRAY[3],DATA[4]),F_DELIMITER))
   SERV_LINE.append("Tempo medio di interuscita su %d simulazioni: %s"%
         (DATA[5],F_DELIMITER))
   SERV_LINE.append("%s%s\n"%(mp.nstr(ARRAY[4],DATA[4]),F_DELIMITER))
   SERV_LINE.append("Media delle varianze dei tempi medi di interuscita su %d simulazioni:
         %s"%(DATA[5],F_DELIMITER))
   SERV_LINE.append("%s%s\n"%(mp.nstr(ARRAY[5],DATA[4]),F_DELIMITER))
   SERV_LINE.append("TH medio di uscita su %d simulazioni: %s"%(DATA[5],F_DELIMITER))
   SERV_LINE.append("%s%s\n"%(mp.nstr(1/ARRAY[4],DATA[4]),F_DELIMITER))
   SERV_LINE.append("Livello medio di WIP su %d simulazioni:%s "%
         (DATA[5],F_DELIMITER))
```

```
    SERV_LINE.append("%s%s\n"%(mp.nstr(ARRAY[6],DATA[4]),F_DELIMITER))
    SERV_LINE.append("Media delle varianze dei livelli medi di WIP su %d simulazioni:%s "%
          (DATA[5],F_DELIMITER))
    SERV_LINE.append("%s%s\n"%(mp.nstr(ARRAY[7],DATA[4]),F_DELIMITER))
    if WHO_CALLS=='multi_buff':
       SERV_LINE.append("Numero di pacchetti medi in ingresso alla saturazione su %d
          simulazioni:%s "%(DATA[5],F_DELIMITER))
       SERV_LINE.append("%s%s\n"%(mp.nstr(ARRAY[8],DATA[4]),F_DELIMITER))
    #end if
    SERV_LINE.append("\n\n")

    j_start=-10
    j_prnt=0
    while j_start+j_prnt<COL:
       j_start+=10
       j_prnt=0
       SERV_LINE.append("%s%s"%(TABLE_TEXT_INT,F_DELIMITER_INT))
       print("%s%s"%(TABLE_TEXT_INT,F_DELIMITER_INT))
       while j_prnt<10 and j_start+j_prnt<COL:
          SERV_LINE.append("%d%s"%((j_start+j_prnt+1),F_DELIMITER_INT))
          print("%d%s"%((j_start+j_prnt+1),F_DELIMITER_INT))
          j_prnt+=1
       #end while
       SERV_LINE.append("\n")
       print("\n")
       i_prnt=0
       while i_prnt<ROW:
          SERV_LINE.append("%s%s"%(DESCRIPTION_INT[i_prnt],F_DELIMITER_INT))
          print("%s%s"%(DESCRIPTION_INT[i_prnt],F_DELIMITER_INT))
          j_prnt=0
          while j_prnt<10 and j_start+j_prnt<COL:
             SERV_LINE.append("%s%s"%(mp.nstr(MTX[i_prnt,
                   (j_start+j_prnt)],DATA[4]),F_DELIMITER_INT))
             print("%s%s"%(mp.nstr(MTX[i_prnt,(j_start+j_prnt)],DATA[4]),F_DELIMITER_INT))
             j_prnt+=1
          #end while
          SERV_LINE.append("\n")
          print("\n")
          i_prnt+=1
       #end while
       SERV_LINE.append("\n")
       print("\n")
    #end while
    f.write('%s'%(EXCEL_CONVERT(SERV_LINE)))
    f.close()
    mp.pretty=False
    return()
#end PRINT_TO_MULTI_FINAL_FILE

def PRINT_COEF_INT(F_NAME_PF1, MTX, PREC_PF1, start_row, start_col):
```

```
#Funzione custom dedicata alla creazione e salvataggio dei files contenenti i coeficienti delle varie
#funzioni di probabilità cumulata calcolati dal modellizzatore.
#Viene richiamata dalla funzione custom PRINT_TO_USER_FILE.
#I dati vengono salvati in un formato compatibile con EXCEL tramite la funzione custom
#EXCEL_CONVERTER.
    EXT_PF1=F_EXT
    DEL_PF1=F_DELIMITER
    mp.pretty=True
    f=open("%s.%s" %(F_NAME_PF1,EXT_PF1),"w+")
    SERV_line=[]
    SERV_line.append('funzione \ turno%s'%(DEL_PF1))
    if start_col==0:
        a=1
    else:
        a=0
    #end if
    j_col=start_col
    while j_col<=(MTX.cols-1):
        SERV_line.append('%d%s'%(j_col+a,F_DELIMITER))
        j_col+=1
    #end while
    SERV_line.append('\n')
    f.write('%s'%(EXCEL_CONVERT(SERV_line)))

    i_row=start_row
    while i_row<=int(MTX.rows-1):
        SERV_line=[]
        if i_row==start_row:
            SERV_line.append("%s%s"%('Costante',DEL_PF1))
        elif i_row==start_row+1:
            SERV_line.append("e^(-LAMBDA*Z)%s"%(DEL_PF1))
        else:
            SERV_line.append("z^%d*LAMBDA^%d*e^(-LAMBDA*Z)%s"%(i_row-2,i_row-2,
                    DEL_PF1))
        #end if
        j_col=start_col
        while j_col<=(MTX.cols-1):
            SERV_line.append("%s%s" %(mp.nstr(MTX[i_row,j_col],PREC_PF1),DEL_PF1))
            j_col+=1
        #end while
        SERV_line.append("\n")
        f.write('%s'%(EXCEL_CONVERT(SERV_line)))
        i_row+=1
    # while
    f.close()    #comando di chiusura del file precedentemente aperto
    mp.pretty=False
    return()
#end def PRINT_COEF_INT

def PRINT_COEF_DENSITY(NAME_PF2, MTX, PREC_PF2,start_row,start_col):
```

```
#Funzione custom dedicata alla creazione e salvataggio dei files contenenti i coeficienti delle varie
#funzioni di distribuzione di densità della probabilità calcolati dal modellizzatore.
#Viene richiamata dalla funzione custom PRINT_TO_USER_FILE.
#I dati vengono salvati in un formato compatibile con EXCEL tramite la funzione custom
#EXCEL_CONVERTER.
   EXT_PF2=F_EXT
   DEL_PF2=F_DELIMITER

   mp.pretty=True
   f=open("%s.%s" %(NAME_PF2,EXT_PF2),"w+")
   SERV_line=[]
   SERV_line.append("funzione \ turno%s"%(DEL_PF2))
   if start_col==0:
      a=1
   else:
      a=0
   #end if
   j_col=start_col
   while j_col<=(MTX.cols-1):
      SERV_line.append("%d,"%(j_col+a))
      j_col+=1
   #end while
   SERV_line.append("\n")
   f.write('%s'%(EXCEL_CONVERT(SERV_line)))
   i_row=start_row
   while i_row<=(MTX.rows-1):
      SERV_line=[]
      SERV_line.append("z^%d*LAMBDA^%d*e^(-LAMBDA*Z)%s%s"%(i_row-1,i_row,'   ',
         DEL_PF2))
      j_col=start_col
      while j_col<=(MTX.cols-1):
         SERV_line.append("%s%s" %(mp.nstr(MTX[i_row,j_col],PREC_PF2),DEL_PF2))
         j_col+=1
      #end while
      SERV_line.append("\n")
      f.write('%s'%(EXCEL_CONVERT(SERV_line)))
      i_row+=1
   #end while
   f.close()
   mp.pretty=False
   return()
#end def PRINT_COEF_DENSITY

def TIMELESS_LIBRARY(TYPE_TL, DEL_TL):
#Funzione custom che fornisce le librerie di stampa su file alla funzione custom
#PRINT_TIMELESS.
   OUT1_TL=[]
   OUT2_TL=[]
   if TYPE_TL==0:
      OUT1_TL.append(['Prima Linea =%s'%(DEL_TL), 'Intertempi medi di arrivo specifici della
```

```
        singola coppia%s\n'%(DEL_TL)])
    OUT1_TL.append(['Seconda Linea =%s'%(DEL_TL), 'Varianze degli intertempi di arrivo
        specifici della singola coppia%s\n'%(DEL_TL)])
    OUT1_TL.append(['Terza Linea =%s'%(DEL_TL), 'Tempi medi di attesa specifici della
        singola coppia%s\n'%(DEL_TL)])
    OUT1_TL.append(['Quarta Linea =%s'%(DEL_TL), 'Varianze dei tempi di attesa specifici
        della singola coppia%s\n'%(DEL_TL)])
    OUT1_TL.append(['Quinta Linea =%s'%(DEL_TL), 'Intertempi medi di uscita specifici della
        singola coppia%s\n'%(DEL_TL)])
    OUT1_TL.append(['Sesta Linea =%s'%(DEL_TL), 'Varianze degli intertempi di uscita
        specifici della singola coppia%s\n'%(DEL_TL)])
    OUT2_TL.append('Intertempo medio di arrivo specifico della coppia%s'%(DEL_TL))
    OUT2_TL.append('Varianza dei tempi di arrivo specifici della coppia%s'%(DEL_TL))
    OUT2_TL.append('Tempo medio di attesa specifico della singola coppia%s'%(DEL_TL))
    OUT2_TL.append('Varianza dei tempi di attesa specifici della singola coppia%s'%(DEL_TL))
    OUT2_TL.append('Intertempo medio di uscita specifico della singola coppia%s'%(DEL_TL))
    OUT2_TL.append('Varianza degli intertempi di uscita specifici della singola coppia%s'%
        (DEL_TL))

elif TYPE_TL==1:
    OUT1_TL.append(['Prima Linea =%s'%(DEL_TL), 'Media degli intertempi medi di arrivo su
        un lotto di dimensione pari all indice delle colonne%s\n'%(DEL_TL)])
    OUT1_TL.append(['Seconda Linea =%s'%(DEL_TL), 'Media delle varianze degli intertempi
        di arrivo su un lotto di dimensione pari all indice delle colonne%s\n'%(DEL_TL)])
    OUT1_TL.append(['Terza Linea =%s'%(DEL_TL), 'Media dei tempi medi di attesa su un lotto
        di dimensione pari all indice delle colonne%s\n'%(DEL_TL)])
    OUT1_TL.append(['Quarta Linea =%s'%(DEL_TL), 'Media delle varianze dei tempi di attesa
        su un lotto di dimensione pari all indice delle colonne%s\n'%(DEL_TL)])
    OUT1_TL.append(['Quinta Linea =%s'%(DEL_TL), 'Media degli intertempi medi di uscita su
        un lotto di dimensione pari all indice delle colonne%s\n'%(DEL_TL)])
    OUT1_TL.append(['Sesta Linea =%s'%(DEL_TL), 'Media delle varianze degli intertempi di
        uscita su un lotto di dimensione pari all indice delle colonne%s\n'%(DEL_TL)])
    OUT2_TL.append('Media degli intertempi medi di arrivo su N coppie%s'%(DEL_TL))
    OUT2_TL.append('Media delle varianze degli intertempi di arrivo su N coppie%s'%
        (DEL_TL))
    OUT2_TL.append('Media dei tempi medi di attesa su N coppie%s'%(DEL_TL))
    OUT2_TL.append('Media delle varianze dei tempi di attesa su N coppie%s'%(DEL_TL))
    OUT2_TL.append('Media degli intertempi medi di uscita su N coppie%s'%(DEL_TL))
    OUT2_TL.append('Media delle varianze degli intertempi di uscita su N coppie%s'%
        (DEL_TL))

elif TYPE_TL==2:
    OUT1_TL.append(['Prima Linea =%s'%(DEL_TL), 'Intertempo medio di arrivo globale su un
        lotto di dimensione pari all indice delle colonne%s\n'%(DEL_TL)])
    OUT1_TL.append(['Seconda Linea =%s'%(DEL_TL), 'Varianza degli intertempi di arrivo
        globali su un lotto di dimensione pari all indice delle colonne%s\n'%(DEL_TL)])
    OUT1_TL.append(['Terza Linea =%s'%(DEL_TL), 'Tempo medio di attesa globale su un lotto
        di dimensione pari all indice delle colonne%s\n'%(DEL_TL)])
    OUT1_TL.append(['Quarta Linea =%s'%(DEL_TL), 'Media delle varianze dei tempi di attesa
        globali su un lotto di dimensione
```

```
        pari all indice delle colonne%s\n'%(DEL_TL)])
    OUT1_TL.append(['Quinta Linea =%s'%(DEL_TL), 'Intertempo medio di uscita globale su un
        lotto di dimensione pari all indice delle colonne%s\n'%(DEL_TL)])
    OUT1_TL.append(['Sesta Linea =%s'%(DEL_TL), 'Varianza degli intertempi di uscita globali
        su un lotto di dimensione pari all indice delle colonne%s\n'%(DEL_TL)])
    OUT2_TL.append('Intertempo medio di arrivo globale su N coppie%s'%(DEL_TL))
    OUT2_TL.append('Varianza degli intertempi di arrivo globali su N coppie%s'%(DEL_TL))
    OUT2_TL.append('Tempo medio di attesa globale su N coppie%s'%(DEL_TL))
    OUT2_TL.append('Media delle varianze dei tempi di attesa globali su N coppie%s'%
        (DEL_TL))
    OUT2_TL.append('Intertempo medio di uscita globale su N coppie%s'%(DEL_TL))
    OUT2_TL.append('Varianza degli intertempi di uscita globali su N coppie%s'%(DEL_TL))

elif TYPE_TL==3:
    OUT1_TL.append(['Prima Linea =%s'%(DEL_TL), 'Livelli di WIP medi calcolati tramite
        tabella probabilità wip su N coppie%s\n'%(DEL_TL)])
    OUT1_TL.append(['Seconda Linea =%s'%(DEL_TL), 'Livelli di WIP medi su N coppie
        calcolati tramite legge di Little (TH media di intertempi medi di ingresso e CT pari a media
        di tempi medi di attesa)%s\n'%(DEL_TL)])
    OUT1_TL.append(['Terza Linea =%s'%(DEL_TL), 'Livelli di WIP medi su N coppie calcolati
        tramite legge di Little (TH media di intertempi medi di uscita e CT pari a media di tempi
        medi di attesa)%s\n'%(DEL_TL)])
    OUT1_TL.append(['Quarta Linea =%s'%(DEL_TL), 'Livelli di WIP medi su N coppie
        calcolati tramite legge di Little (TH medio di uscita globale e CT pari a attesa media
        globale)%s\n'%(DEL_TL)])
    OUT2_TL.append('WIP 1%s'%(DEL_TL))
    OUT2_TL.append('WIP 2%s'%(DEL_TL))
    OUT2_TL.append('WIP 3%s'%(DEL_TL))
    OUT2_TL.append('WIP 4%s'%(DEL_TL))

elif TYPE_TL==4:
    OUT1_TL.append(['Prima Linea =%s'%(DEL_TL), 'TH di uscita calcolato tramite legge di
        Little (WIP da tabella e CT pari a media di tempi medi di attesa)%s\n'%(DEL_TL)])
    OUT1_TL.append(['Seconda Linea =%s'%(DEL_TL), 'Media dei TH di ingresso medi su N
        coppie%s\n'%(DEL_TL)])
    OUT1_TL.append(['Terza Linea =%s'%(DEL_TL), 'Media dei TH di uscita medi su N coppie
        %s\n'%(DEL_TL)])
    OUT1_TL.append(['Quarta Linea =%s'%(DEL_TL), 'TH di uscita medio globale su N coppie
        %s\n'%(DEL_TL)])
    OUT2_TL.append('TH 1%s'%(DEL_TL))
    OUT2_TL.append('TH 2%s'%(DEL_TL))
    OUT2_TL.append('TH 3%s'%(DEL_TL))
    OUT2_TL.append('TH 4%s'%(DEL_TL))

elif TYPE_TL==5:
    OUT1_TL.append(['Prima Linea =%s'%(DEL_TL), 'CT pari a media di tempi medi di attesa
        su N coppie%s\n'%(DEL_TL)])
    OUT1_TL.append(['Seconda Linea =%s'%(DEL_TL), 'CT su N coppie calcolato tramite legge
        di Little (TH media di intertempi medi di uscita e WIP da tabella)%s\n'%(DEL_TL)])
    OUT1_TL.append(['Terza Linea =%s'%(DEL_TL), 'CT medio globale%s\n'%(DEL_TL)])
```

```
      OUT2_TL.append('CT 1%s'%(DEL_TL))
      OUT2_TL.append('CT 2%s'%(DEL_TL))
      OUT2_TL.append('CT 3%s'%(DEL_TL))
   #end if
   OUT_TL=[OUT1_TL, OUT2_TL]
   return(OUT_TL)
#end def TIMELESS_LIBRARY

def PRINT_TIMELESS(DATA_PTT):
#Funzione custom deputata alla stampa su files dei dati prodotti dalla sezione di modellizzazione
#"dati timeless" in un formato compatibile con EXCEL tramite la funzione custom
#EXCEL_CONVERT.
#Viene richiamata dalla funzione custom PRINT_TO_USER_FILE.
#Utilizza la funzione custom TIMELESS_LIBRARY per selezionare le librerie di stampa specifiche
#dei singoli files.
   EXT_PTT=F_EXT
   DEL_PTT=F_DELIMITER
   PREC_PTT=DATA_PTT[2]
   F_TEXT_PAR=['wip','TH','CT']
   mp.pretty=True
   count_par=0
   start_line_par=0
   for i_file in range(6):
      if i_file<3:
         f=open('%s%d%s.%s'%(F_NAME_USER[3][0], DATA_PTT[1], F_NAME_USER[3]
                  [i_file+1],EXT_PTT),"w+")
         SERV_MTX=DATA_PTT[0][i_file]
         SERV_LIBRARY=TIMELESS_LIBRARY(i_file,DEL_PTT)
      else:
         f=open('%s%du_%s%s.%s'%(F_NAME_USER[3][0], DATA_PTT[1],
                  F_TEXT_PAR[count_par], F_NAME_USER[3][5], EXT_PTT),"w+")
         SERV_LIBRARY=TIMELESS_LIBRARY(i_file,DEL_PTT)
         SERV2_MTX=DATA_PTT[0][3]
         if count_par<2:
            SERV_MTX=mp.zeros(4,SERV2_MTX.cols)
         else:
            SERV_MTX=mp.zeros(3,SERV2_MTX.cols)
         #end if
         for j in range(SERV_MTX.cols):
            for i in range(SERV_MTX.rows):
               SERV_MTX[i,j]=SERV2_MTX[i+start_line_par,j]
            #end for i
         #end for j
         count_par+=1
         start_line_par+=4
      #end if
      ROW=SERV_MTX.rows
      COL=SERV_MTX.cols
      SERV_line=[]
      SERV_line.append('LEGENDA\n')
```

```
for j in range(len(SERV_LIBRARY[0])):
   SERV_TEXT=SERV_LIBRARY[0][j]
   SERV_line.append(SERV_TEXT[0])
   SERV_line.append(SERV_TEXT[1])
#end for j
SERV_line.append('\n\n')

i=0
c1=0
c2=0
SERV_line=[]
SERV_line.append("Pezzi%s"%(DEL_PTT))
for j in range(10):
   SERV_line.append("%d%s"%(j+1,DEL_PTT))
#end for j
SERV_line.append("\n")
f.write('%s'%(EXCEL_CONVERT(SERV_line)))
SERV_line=[]
SERV_line.append(SERV_LIBRARY[1][0])
for i in range(COL):
   SERV_line.append("%s%s"%(mp.nstr(SERV_MTX[0,i],PREC_PTT),DEL_PTT))
   c1+=1
   if c1==10:
      SERV_line.append("\n")
      SERV_line.append(SERV_LIBRARY[1][1])
      c1=c2*10
      while c1<=i:
         SERV_line.append("%s%s"%(mp.nstr(SERV_MTX[1,c1],PREC_PTT),DEL_PTT))
         c1+=1
      #end while
      SERV_line.append("\n\n")
      SERV_line.append(SERV_LIBRARY[1][2])
      c1=c2*10
      while c1<=i:
         if i_file==0:
            SERV_line.append("%s%s"%
                  (mp.nstr(SERV_MTX[2,c1+1],PREC_PTT),DEL_PTT))
         else:
            SERV_line.append("%s%s"%(mp.nstr(SERV_MTX[2,c1],PREC_PTT),DEL_PTT))
         #end if
         c1+=1
      #end while
      SERV_line.append("\n")
      if i_file<5:
         SERV_line.append(SERV_LIBRARY[1][3])
         c1=c2*10
         while c1<=i:
            if i_file==0:
               SERV_line.append("%s%s"%
                     (mp.nstr(SERV_MTX[3,c1+1],PREC_PTT),DEL_PTT))
```

```
            else:
               SERV_line.append("%s%s"%
                     (mp.nstr(SERV_MTX[3,c1],PREC_PTT),DEL_PTT))
            #end if
            c1+=1
         #end while
         SERV_line.append("\n\n")
         if i_file<3:
            SERV_line.append(SERV_LIBRARY[1][4])    #intestazione linea
            c1=c2*10
            while c1<=i:
               SERV_line.append("%s%s"%
                     (mp.nstr(SERV_MTX[4,c1],PREC_PTT),DEL_PTT))
               c1+=1
            #end while
            SERV_line.append("\n")
            SERV_line.append(SERV_LIBRARY[1][5])   #intestazione linea
            c1=c2*10
            while c1<=i:
               SERV_line.append("%s%s"%
                     (mp.nstr(SERV_MTX[5,c1],PREC_PTT),DEL_PTT))
               c1+=1
            #end while
            SERV_line.append("\n\n\n")
            c2+=1
            c1=c2*10
            if c1<COL-1:
               SERV_line.append("Pezzi%s"%(DEL_PTT))
               while c1<(c2+1)*10 and c1<=COL-1:
                  SERV_line.append("%d%s"%(c1+1,DEL_PTT))
                  c1+=1
               #end while
               SERV_line.append("\n")
               SERV_line.append(SERV_LIBRARY[1][0])
               c1=0
            #end if
         #end if
      #end if
   #end if
#end for
c1=c2*10
if c1<COL-1:
   SERV_line.append("\n")
   SERV_line.append(SERV_LIBRARY[1][1])
   while c1<=i:
      SERV_line.append("%s%s"%(mp.nstr(SERV_MTX[1,c1],PREC_PTT),DEL_PTT))
      c1+=1
   #end while
   SERV_line.append("\n\n")
   c1=c2*10
```

```
            SERV_line.append(SERV_LIBRARY[1][2])
            while c1<=i:
               if i_file==0:
                  SERV_line.append("%s%s"%(mp.nstr(SERV_MTX[2,c1+1],PREC_PTT),DEL_PTT))
               else:
                  SERV_line.append("%s%s"%(mp.nstr(SERV_MTX[2,c1],PREC_PTT),DEL_PTT))
               #end if
               c1+=1
            #end while
            SERV_line.append("\n")
            if i_file<5:
               SERV_line.append(SERV_LIBRARY[1][3])
               c1=c2*10
               while c1<=i:
                  if i_file==0:
                     SERV_line.append("%s%s"%
                           (mp.nstr(SERV_MTX[3,c1+1],PREC_PTT),DEL_PTT))
                  else:
                     SERV_line.append("%s%s"%(mp.nstr(SERV_MTX[3,c1],PREC_PTT),DEL_PTT))
                  #end if
                  c1+=1
               #end while
               SERV_line.append("\n\n")
               if i_file<3:
                  c1=c2*10
                  SERV_line.append(SERV_LIBRARY[1][4])
                  while c1<=i:
                     SERV_line.append("%s%s"%(mp.nstr(SERV_MTX[4,c1],PREC_PTT),DEL_PTT))
                     c1+=1
                  #end while
                  SERV_line.append("\n")
                  SERV_line.append(SERV_LIBRARY[1][5])
                  c1=c2*10
                  while c1<=i:
                     SERV_line.append("%s%s"%(mp.nstr(SERV_MTX[5,c1],PREC_PTT),DEL_PTT))
                     c1+=1
                  #end while
               #end if
            #end if
         #end if
         f.write('%s'%(EXCEL_CONVERT(SERV_line)))
         f.close()
      #end for i
      mp.pretty=False
      return()
#end def PRINT_TIMES_TIMELESS

def PRINT_TABS_COMMON(TV_MED_PTC,GLOBAL_PTC,PARAMETERS_PTC, N_PTC,
T_UP_PTC,T_METER_PTC,PREC_PTC):
#Funzione custom che fornisce in output delle stringhe di testo che verranno utilizzate dalle due
```

```
#possibili funzioni custom chiamanti (PRINT_TIMES e PRINT_BUFFER) in fase di salvataggio
#files.
    SERV_PTC=[]
    SERV_PTC.append('DATI INIZIALI DEL SISTEMA\n')
    SERV_PTC.append('Numero di coppie totali entrate nella stazione di accoppiamento: %d
        coppie\n'%(N_PTC))
    SERV_PTC.append('Intertempo medio di uscita delle singole stazioni a monte: %s %s\n'%
        (T_UP_PTC,T_METER_PTC))
    SERV_PTC.append('\n')
    SERV_PTC.append('PARAMETRI CALCOLATI IN ASSENZA DI BUFFER\n')
    SERV_PTC.append('TEMPI DI INTERARRIVO\n')
    SERV_PTC.append("La media dei singoli tempi medi di interarrivo alla stazione di
        accoppiamento: %s %s\n"%(mp.nstr(TV_MED_PTC[0,0],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("La media delle varianze dei singoli tempi medi di interarrivo è: %s
        %s^2\n"%(mp.nstr(TV_MED_PTC[0,1],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("Il tempo medio di interarrivo da distribuzione di probabilità globale) alla
        stazione di accoppiamento: %s %s\n"%
        (mp.nstr(GLOBAL_PTC[0,0],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("La varianza dei tempi medi di interarrivo è: %s %s^2\n"%
        (mp.nstr(GLOBAL_PTC[0,1],PREC_PTC),T_METER_PTC))
    SERV_PTC.append('\n')
    SERV_PTC.append('TEMPI DI ATTESA\n')
    SERV_PTC.append("La media dei singoli tempi medi attesa è: %s %s\n"%
        (mp.nstr(TV_MED_PTC[0,2],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("La media delle varianze dei singoli tempi medi di attesa è: %s %s^2\n"%
        (mp.nstr(TV_MED_PTC[0,3],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("Il tempo medio di attesa (da distribuzione di probabilità globale) alla
        stazione di accoppiamento: %s %s\n"%
        (mp.nstr(GLOBAL_PTC[0,2],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("La varianza dei tempi medi di attesa è: %s %s^2\n"%
        (mp.nstr(GLOBAL_PTC[0,3],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("\n")
    SERV_PTC.append('TEMPI DI INTERUSCITA\n')
    SERV_PTC.append("Il tempo medio di interuscita dalla stazione è: %s %s\n"%
        (mp.nstr(TV_MED_PTC[0,4],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("La media delle varianze dei tempi di interuscita dalla stazione è: %s
        %s^2\n"%(mp.nstr(TV_MED_PTC[0,5],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("Il tempo medio di interuscita (da distribuzione di probabilità globale) alla s
        tazione di accoppiamento: %s %s\n"%
        (mp.nstr(GLOBAL_PTC[0,4],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("La varianza dei tempi medi di interuscita è: %s %s^2\n"%
        (mp.nstr(GLOBAL_PTC[0,5],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("\n\n")
    SERV_PTC.append("I PARAMETRI DEL SISTEMA CALCOLATI CON EGUAGLIANZA DI
        LITTLE\n")
    SERV_PTC.append("DATI INGRESSO: WIP calcolato tramite tabella WIP e CT pari a media
        tempi attesa)\n")
    SERV_PTC.append("i parametri medi del sistema su un lotto da %d pezzi sono:\n"%(N_PTC))
    SERV_PTC.append("    WIP= %s unità\n"%(mp.nstr(PARAMETERS_PTC[0,0],PREC_PTC)))
    SERV_PTC.append("    TH = %s unità/%s\n"%
```

```
        (mp.nstr(PARAMETERS_PTC[0,1],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("    CT = %s %s\n"%
        (mp.nstr(PARAMETERS_PTC[0,2],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("\n")
    SERV_PTC.append("DATI INGRESSO: TH pari al TH in ingresso e CT pari a media tempi
        attesa\n")
    SERV_PTC.append("i parametri medi del sistema su un lotto da %d pezzi sono:\n"%(N_PTC))
    SERV_PTC.append("    WIP= %s unità\n"%(mp.nstr(PARAMETERS_PTC[1,0],PREC_PTC)))
    SERV_PTC.append("    TH = %s unità/%s\n"%
        (mp.nstr(PARAMETERS_PTC[1,1],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("    CT = %s %s\n"%
        (mp.nstr(PARAMETERS_PTC[1,2],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("\n")
    SERV_PTC.append("DATI INGRESSO: TH pari al TH in uscita e CT pari a media tempi
        attesa\n")
    SERV_PTC.append("i parametri medi del sistema su un lotto da %d pezzi sono:\n"%(N_PTC))
    SERV_PTC.append("    WIP= %s unità\n"%(mp.nstr(PARAMETERS_PTC[2,0],PREC_PTC)))
    SERV_PTC.append("    TH = %s unità/%s\n"%
        (mp.nstr(PARAMETERS_PTC[2,1],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("    CT = %s %s\n"%
        (mp.nstr(PARAMETERS_PTC[2,2],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("\n")
    SERV_PTC.append("DATI INGRESSO: TH pari al TH in uscita e WIP calcolato tramite tabella
        WIP\n")
    SERV_PTC.append("i parametri medi del sistema su un lotto da %d pezzi sono:\n"%(N_PTC))
    SERV_PTC.append("    WIP= %s unità\n"%(mp.nstr(PARAMETERS_PTC[3,0],PREC_PTC)))
    SERV_PTC.append("    TH = %s unità/%s\n"%
        (mp.nstr(PARAMETERS_PTC[3,1],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("    CT = %s %s\n"%
        (mp.nstr(PARAMETERS_PTC[3,2],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("\n")
    SERV_PTC.append("DATI INGRESSO: TH di uscita e CT calcolati tramite funzioni di densità
        globali sul lotto\n")
    SERV_PTC.append("i parametri medi del sistema su un lotto da %d pezzi sono:\n"%(N_PTC))
    SERV_PTC.append("    WIP= %s unità\n"%(mp.nstr(PARAMETERS_PTC[4,0],PREC_PTC)))
    SERV_PTC.append("    TH = %s unità/%s\n"%
        (mp.nstr(PARAMETERS_PTC[4,1],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("    CT = %s %s\n"%
        (mp.nstr(PARAMETERS_PTC[4,2],PREC_PTC),T_METER_PTC))
    SERV_PTC.append("\n\n")
    SERV_PTC.append("NB: CT e tempo medio di attesa coincidono perché in questo caso abbiamo
        considerato il tempo fisico di accoppiamento\n")
    SERV_PTC.append('     con entrambi i componenti presenti come nullo\n')
    SERV_PTC.append('     Il Cycle Time è quindi composto dal solo tempo di attesa\n')
    SERV_PTC.append("     Se così non fosse i due differirebbero\n\n\n")
    return(SERV_PTC)
#end PRINT_TABS_COMMON

def PRINT_TIMES(NAME_PF3, TV_PF3, TV_MED_PF3, GLOBAL_PF3, PARAMETERS_PF3,
N_PF3, T_UP_PF3, T_METER_PF3, PREC_PF3):
```

```
#Funzione custom deputata alla creazione e al salvataggio in files compatibili con EXCEL dei dati
#prodotti dalla modellizzazione di un caso specifico in assenza di buffer.
#Viene richiamata dalla funzione custom PRINT_TO_USER_FILE.
#In esecuzione richiamerà le funzioni custom PRINT_TABS_COMMON e EXCEL_CONVERT
    EXT_PF3=F_EXT
    DEL_PF3=F_DELIMITER
    ROW=TV_PF3.rows
    COL=TV_PF3.cols
    mp.pretty=True
    f=open('%s.%s'%(NAME_PF3,EXT_PF3),"w+")
    SERV_line=[]
    SERV_line=PRINT_TABS_COMMON(TV_MED_PF3, GLOBAL_PF3,PARAMETERS_PF3,
          N_PF3 , T_UP_PF3, T_METER_PF3, PREC_PF3)
    f.write('%s'%(EXCEL_CONVERT(SERV_line)))

    i=0
    c1=0
    c2=0
    SERV_line=[]
    SERV_line.append("Pezzi%s"%(DEL_PF3))
    for j in range(10):
       SERV_line.append("%d%s"%(j+1,DEL_PF3))
    #end for j
    SERV_line.append("\n")
    f.write('%s'%(EXCEL_CONVERT(SERV_line)))
    SERV_line=[]
    SERV_line.append("Tempi arrivo%s"%(DEL_PF3))    #intestazione linea
    for i in range(COL-1):
       SERV_line.append("%s%s"%(mp.nstr(TV_PF3[0,i],PREC_PF3),DEL_PF3))
       c1+=1
       if c1==10:
          SERV_line.append("\n")
          SERV_line.append("Varianze tempi arrivo%s"%(DEL_PF3)) #intestazione linea
          c1=c2*10
          while c1<=i:
             SERV_line.append("%s%s"%(mp.nstr(TV_PF3[1,c1],PREC_PF3),DEL_PF3))
             c1+=1
          #end while
          SERV_line.append("\n\n")
          SERV_line.append("Tempi attesa%s"%(DEL_PF3))    #intestazione linea
          c1=c2*10
          while c1<=i:
             SERV_line.append("%s%s"%(mp.nstr(TV_PF3[2,c1+1],PREC_PF3),DEL_PF3))
             c1+=1
          #end while
          SERV_line.append("\n")
          SERV_line.append("Varianze tempi attesa%s"%(DEL_PF3))   #intestazione linea
          c1=c2*10
          while c1<=i:
             SERV_line.append("%s%s"%(mp.nstr(TV_PF3[3,c1+1],PREC_PF3),DEL_PF3))
```

```
            c1+=1
        #end while
        SERV_line.append("\n\n")
        SERV_line.append("Intertempi di uscita%s"%(DEL_PF3))    #intestazione linea
        c1=c2*10
        while c1<=i:
            SERV_line.append("%s%s"%(mp.nstr(TV_PF3[4,c1],PREC_PF3),DEL_PF3))
            c1+=1
        #end while
        SERV_line.append("\n")
        SERV_line.append("Varianze intertempi di uscita%s"%(DEL_PF3))   #intestazione linea
        c1=c2*10
        while c1<=i:
            SERV_line.append("%s%s"%(mp.nstr(TV_PF3[5,c1],PREC_PF3),DEL_PF3))
            c1+=1
        #end while
        SERV_line.append("\n\n\n")
        c2+=1
        c1=c2*10
        if c1<COL-2:
            SERV_line.append("Pezzi%s"%(DEL_PF3))
            while c1<(c2+1)*10 and c1<=COL-2:
                SERV_line.append("%d%s"%(c1+1,DEL_PF3))
                c1+=1
            #end while
            SERV_line.append("\n")
            SERV_line.append("Tempi arrivo%s"%(DEL_PF3))
            c1=0
        #end if
    #end if
#end for
c1=c2*10
if c1<COL-2:
    SERV_line.append("\n")
    SERV_line.append("Varianze tempi di arrivo%s"%(DEL_PF3))
    while c1<=i:
        SERV_line.append("%s%s"%(mp.nstr(TV_PF3[1,c1],PREC_PF3),DEL_PF3))
        c1+=1
    #end while
    SERV_line.append("\n\n")
    c1=c2*10
    SERV_line.append("Tempi di attesa%s"%(DEL_PF3))
    while c1<=i:
        SERV_line.append("%s%s"%(mp.nstr(TV_PF3[2,c1+1],PREC_PF3),DEL_PF3))
        c1+=1
    #end while
    SERV_line.append("\n")
    SERV_line.append("Varianze di tempi attesa%s"%(DEL_PF3))
    c1=c2*10
    while c1<=i:
```

```
            SERV_line.append("%s%s"%(mp.nstr(TV_PF3[3,c1+1],PREC_PF3),DEL_PF3))
            c1+=1
        #end while
        SERV_line.append("\n\n")
        c1=c2*10
        SERV_line.append("Intertempi di uscita%s"%(DEL_PF3))
        while c1<=i:
            SERV_line.append("%s%s"%(mp.nstr(TV_PF3[4,c1],PREC_PF3),DEL_PF3))
            c1+=1
        #end while
        SERV_line.append("\n")
        SERV_line.append("Varianze di tempi attesa%s"%(DEL_PF3))
        c1=c2*10
        while c1<=i:
            SERV_line.append("%s%s"%(mp.nstr(TV_PF3[5,c1],PREC_PF3),DEL_PF3))
            c1+=1
        #end while
    #end if
    f.write('%s'%(EXCEL_CONVERT(SERV_line)))
    f.close()
    mp.pretty=False
    return()
#end def PRINT_TIMES

def PRINT_WIP_TABS(NAME_PF4, MTX, PREC_PF4):
#Funzione custom dedicata alla creazione e salvataggio su file della matrice delle probabilità dei
#livelli di WIP attesi al termine di ogni "turno".
#Viene richiamata dalla funzione custom PRINT_TO_USER_FILE.
#I dati vengono trasformati in formato compatibile con EXCEL con la funzione custom
#EXCEL_CONVERT.
    EXT_PF4=F_EXT
    DEL_PF4=F_DELIMITER
    mp.pretty=True
    f=open("%s.%s" %(NAME_PF4,EXT_PF4),"w+")
    SERV_line=[]
    SERV_line.append("Turno \  WIP%s"%(DEL_PF4))
    for j_col in range(MTX.cols):
        SERV_line.append("%d,"%(j_col))
    #end for j
    SERV_line.append("\n")
    for i_row in range(MTX.rows):
        SERV_line.append("%d%s"%(i_row,DEL_PF4))
        for j_col in range(MTX.cols):
            SERV_line.append("%s%s" %(mp.nstr(MTX[i_row,j_col],PREC_PF4),DEL_PF4))
        #end for j
        SERV_line.append("\n")
    #end for i
    f.write('%s'%(EXCEL_CONVERT(SERV_line)))
    f.close()
    mp.pretty=False
```

```
    return()
#end PRINT_TO_FILE4

def PRINT_BUFFER(FILE_PF5, MTX, BPAR_PF5, TV_PF5, TV_MED_PF5, GLOBAL_PF5,
PARAMETER_PF5, N_PF5, T_UP_PF5, T_METER_PF5, PREC_PF5):
#Funzione custom deputata alla creazione e al salvataggio in files compatibili con EXCEL dei dati
#prodotti dalla modellizzazione di un caso specifico in presenza di buffer.
#Viene richiamata dalla funzione custom PRINT_TO_USER_FILE.
#In esecuzione richiamerà le funzioni custom PRINT_TABS_COMMON e EXCEL_CONVERT
    EXT_PF5=F_EXT
    DEL_PF5=F_DELIMITER
    mp.pretty=True
    f=open("%s.%s" %(FILE_PF5,EXT_PF5),"w+")
    N_PF5=MTX.rows
    BUFF=MTX.cols-3

    SERV_line=PRINT_TABS_COMMON(TV_MED_PF5,GLOBAL_PF5,PARAMETER_PF5,
        N_PF5 ,T_UP_PF5, T_METER_PF5, PREC_PF5)
    f.write('%s'%(EXCEL_CONVERT(SERV_line)))
    SERV_line=[]
    SERV_line.append("\n\nPARAMETRI MEDI DEL SISTEMA CON BUFFER = %d:\n"%
        (BUFF))
    SERV_line.append("    WIP= %s unità\n"%(mp.nstr(BPAR_PF5[0,0],PREC_PF5)))
    SERV_line.append("    TH = %s unità/%s\n"%
        (mp.nstr(BPAR_PF5[0,1],PREC_PF5),T_METER_PF5))
    SERV_line.append("    CT = %s %s\n"%
        (mp.nstr(BPAR_PF5[0,2],PREC_PF5),T_METER_PF5))
    SERV_line.append("    Numero di pezzi entrati prima di una interruzione: %d\n"%
        (BPAR_PF5[0,3]))
    SERV_line.append("\n\n")
    f.write('%s'%(EXCEL_CONVERT(SERV_line)))
    SERV_line=[]
    SERV_line.append("Turno \  WIP%s"%(DEL_PF5))
    for j_col in range(MTX.cols-1):
        SERV_line.append("%d,"%(j_col))
    #end for j
    SERV_line.append("%s%s" %('STOP',DEL_PF5))
    SERV_line.append("\n")
    f.write('%s'%(EXCEL_CONVERT(SERV_line)))

    SERV_line=[]
    for i_row in range(MTX.rows):
        SERV_line.append("%d%s"%(i_row,DEL_PF5))
        for j_col in range(MTX.cols):
            SERV_line.append("%s%s" %(mp.nstr(MTX[i_row,j_col],PREC_PF5),DEL_PF5))
        #end FOR J
        SERV_line.append("\n")
    #end for i
    f.write('%s'%(EXCEL_CONVERT(SERV_line)))
    f.close()
```

```
    mp.pretty=False
    return()
#end PRINT_TO_FILE5

def PRINT_TO_USER_FILE(WHO_CALLS,DATA_PUF):
#Funzione custom deputata alla coordinazione delle operazioni necessarie alla creazione e
#salvataggio dei files dati prodotti dalle funzioni custom NEW_SET_ELABORATION, TIMELESS
#e GENERAL_MATRIX_GENERATION in un formato compatibile con EXCEL.
#In base alla funzione richiamante, verranno richiamate le funzioni custom PRINT_COEF_INT,
#PRINT_COEF_DENSITY, PRINT_TIMELESS, PRINT_WIP_TABS, PRINT_TIMES,
#PRINT_BUFFER.
    N_PUF=DATA_PUF[0]
    F_PREC_PUF=DATA_PUF[1]
    if WHO_CALLS=='mtx_gen':
        i_mtx=2
        while i_mtx<len(DATA_PUF):
            if DATA_PUF[i_mtx][0]=='int':
                SERV_PUF=DATA_PUF[i_mtx][2]
                PRINT_COEF_INT('%s_%d%s'%(F_NAME_USER[0][0], N_PUF, F_NAME_USER[0]
                    [i_mtx-1]), DATA_PUF[i_mtx][1], F_PREC_PUF, SERV_PUF[0], SERV_PUF[1])
            elif DATA_PUF[i_mtx][0]=='den':
                SERV_PUF=DATA_PUF[i_mtx][2]
                PRINT_COEF_DENSITY('%s_%d%s'%(F_NAME_USER[0][0], N_PUF,
                    F_NAME_USER[0][i_mtx-1]), DATA_PUF[i_mtx][1], F_PREC_PUF,
                    SERV_PUF[0], SERV_PUF[1])
            else:
                input('ERRORE STAMPA DENTRO PRINT_TO_USER_FILE dentro colonna if')
            #end if
            i_mtx+=1
            #end while
        #end for i_mtx
    elif WHO_CALLS=='timeless':
        PRINT_TIMELESS([[DATA_PUF[0][0], DATA_PUF[0][1], DATA_PUF[0][2],
            DATA_PUF[0][3]], DATA_PUF[1], DATA_PUF[2]])
        PRINT_WIP_TABS('%s%d%s'%(F_NAME_USER[3][0],DATA_PUF[1],F_NAME_USER[3]
            [4]), DATA_PUF[0][4], DATA_PUF[2])
    elif WHO_CALLS=='tab1_gen':
        PRINT_TIMES('%s%s%s'%(F_NAME_USER[1][0],DATA_PUF[0],F_NAME_USER[1][1]),
            DATA_PUF[3], DATA_PUF[4], DATA_PUF[5], DATA_PUF[6], DATA_PUF[1],
            DATA_PUF[7], DATA_PUF[8], DATA_PUF[2])
    elif WHO_CALLS=='tab2_gen':
        PRINT_WIP_TABS('%s%s%s'%(F_NAME_USER[1][0],DATA_PUF[0],
            F_NAME_USER[1][2]), DATA_PUF[2], DATA_PUF[1])
    elif WHO_CALLS=='buff_gen':
        PRINT_BUFFER('%s%s%s%d'%(F_NAME_USER[2][0],
            DATA_PUF[0],F_NAME_USER[2][1],DATA_PUF[3]), DATA_PUF[4],
            DATA_PUF[5], DATA_PUF[6], DATA_PUF[7], DATA_PUF[8], DATA_PUF[9],
            DATA_PUF[1], DATA_PUF[10], DATA_PUF[11], DATA_PUF[2])
    #end if
    return()
```

```
#end def PRINT_TO_USER_FILE

def PRINT_TO_FILE_PRG(PRG_OUT, COD_OUT):
#Funzione custom deputata alla creazione e salvataggio dei files dati prodotti dalle funzioni custom
#NEW_SET_ELABORATION, TIMELESS e GENERAL_MATRIX_GENERATION in un
#formato leggibile dal programma stesso.
    mp.pretty=True
    F_PREC_INT=COD_OUT[2]
    DEL_INT=COD_OUT[1]
    if COD_OUT[0]==1:
        line_text=0
        start_range=2
        F_NAME_COD='%s%d%s%d'%(F_BASE_NAME[line_text]
          [0],COD_OUT[3],F_BASE_NAME[line_text][1],COD_OUT[2])
    elif COD_OUT[0]==2:
        line_text=1
        start_range=3
        F_NAME_COD='%s%d%s%s%s%s%d'%(F_BASE_NAME[line_text][0],
          COD_OUT[3],F_BASE_NAME[line_text[1],
          COD_OUT[4],COD_OUT[5],F_BASE_NAME[line_text][2],COD_OUT[2])
    elif COD_OUT[0]==3:
        line_text=2
        start_range=4
        F_NAME_COD='%s%d%s%s%s%s%d%s%d'%(F_BASE_NAME[line_text[0],
          COD_OUT[3],F_BASE_NAME[line_text[1],COD_OUT[4],COD_OUT[5],
          F_BASE_NAME[line_text]  [2],COD_OUT[6],F_BASE_NAME[line_text][3],
          COD_OUT[2])
    elif COD_OUT[0]==4:
        line_text=3
        start_range=2
        F_NAME_COD='%s%d%s%d'%(F_BASE_NAME[line_text][0],
          COD_OUT[3],F_BASE_NAME[line_text][1],COD_OUT[2])
    elif COD_OUT[0]==5:
        line_text=4
        start_range=2
        F_NAME_COD='%s%d%s%d'%(F_BASE_NAME[line_text][0],
          COD_OUT[3],F_BASE_NAME[line_text][1],COD_OUT[2])
    else:
        input('ERRORE CHIAMATA PRINT_TO_FILE_PRG, CODICE TYPE_OUT ERRATO!!!')
    #end if
    range_out=len(F_BASE_NAME[line_text])-start_range
    for i_print in range(range_out):
        f=open('%s%s.%s'%(F_NAME_COD,F_BASE_NAME[line_text]
          [start_range+i_print],F_EXT),"w+")
        print('%s%s.%s saved'%(F_NAME_COD,F_BASE_NAME[line_text]
          [start_range+i_print],F_EXT))
        MTX=PRG_OUT[i_print]
        ROW=MTX.rows
        COL=MTX.cols
        f.write('%s%d%s%d%s%d%s\n'%
```

```
        (DEL_INT,ROW,DEL_INT,COL,DEL_INT,F_PREC_INT,DEL_INT))
      for i_rows in range(ROW):
        for j_cols in range(COL):
          f.write('%s%s'%(mp.nstr(MTX[i_rows,j_cols],F_PREC_INT),DEL_INT))
        #end for j_cols
        f.write('\n')
      #end for i_rows
      f.close()
   #end for i_print
   return()
#end PRINT_TO_FILE_PRG

def PRODUTTORIA(lim_inf,lim_sup):
#Funzione custom adibita al calcolo delle produttorie.
#Restituisce in output il valore elaborato.
   if PROD_KIND==0:
      OUT=PROD_MAIN[0,lim_sup]/PROD_MAIN[0,lim_inf-1]
   else:
      OUT=PROD_MAIN[lim_inf,lim_sup]  #sfrutta la mappatura iniziale dei valori di produttoria
   return (OUT)
#end PRODUTTORIA

def MAXIMUM(A,B):
#Funzione custom che restituisce in output il massimo di due argomenti.
   if A<B:
      A=B
   #end if
   return(A)
#end MAXIMUM

def CALCOLO_DELTA_Z(M,n):
#Funzione custom adibita al calcolo dei coeficienti della matrice delta, su cui si baseranno quasi
#tutti i calcoli successivi presenti nella fase di modellizzazione.
#Viene richiamata dalla funzione custom GENERAL_MATRIX_GENERATION.
#Richiama la funzione custom PRODUTTORIA.
#Restituisce in output i dati elaborati.
   w=2
   while (w<n):
      print('%d'%(n-w))
      w+=1
      M[1,w]=PRODUTTORIA(1,w-1)
      M[2,w]=-M[1,w]
      k=2
      while k<w:
         k+=1
         z=k-2
         M[k,w]=-(mp.mpf(1)-mp.power(mp.mpf(1/2),mp.mpf(w+1-z)))*PRODUTTORIA(z+1,w-1)
      #end while
      M[w+1,w]=mp.mpf(-0.75)
      M[w+2,w]=mp.mpf(-1/(2*w))
```

```
    #end while
    return(M)
#end CALCOLO_DELTA_Z

def CALCOLO_INTZ_COEFZ(DZ,IZ,CZ,N_INT):
#Funzione custom adibita al calcolo dei coeficienti delle funzioni di densita di probabilità e di
#probabilità cumulata dei tempi di attesa.
#Viene richiamata dalla funzione custom GENERAL_MATRIX_GENERATION.
#Restituisce in output i dati elaborati.
    w=2
    while w<N_INT:
        print('%d'%(N_INT-w))
        w+=1
        k=0
        while k<w+2:
            k+=1
            z=0
            while z<w:
                z+=1
                IZ[k,w]+=CZ[z,w]*DZ[k,z]
            #end while
        #end while
        k=1
        i=w+1
        while k<i:
            CZ[k,i]=IZ[k+2,w]*mp.mpf(k)-IZ[k+1,w]
            k+=1
        #end WHILE
        CZ[k,i]=-IZ[k+1,w]
    #end while
    OUT=[]
    OUT.append(IZ)
    OUT.append(CZ)
    return(OUT)
#end CALCOLO_INTZ_COEFZ

def GENERAL_MATRIX_GENERATION(WHO_CALLS,DATA_GMG):
#Funzione custom adibita all'elaborazione di parte delle matrici di base contenenti i coeficienti delle
#varie distribuzioni di probabilità su cui si basano tutti i calcoli di modellizzazione.
#Viene richiamata in modalità utente dalla funzione custom USER_THEORY_MAIN o, altrimenti,
#dallo schedulatore.
#In modalità utente, l'immissione dell'input necessario sarà gestita dalla funzione custom
#INPUT_MENU.
#In esecuzione richiamerà le funzioni custom  CALCOLO_DELTA_Z, CALCOLO_INTZ_COEFZ
#e PRODUTTORIA.
#La creazione e salvataggio dei file verrà eseguita dalle funzioni PRINT_TO_USER_FILE e
#PRINT_TO_FILE_PRG.
#I dati elaborati verranno restituiti in output alla funzione chiamante.
    OUT_GMG=[]
    OUT_GMG.append(0)
```

```
MENU_CTRL_GMG=0
while MENU_CTRL_GMG==0:
   SAVE_MTX_SUB_GMG=0
   if WHO_CALLS=='sched':
      INPUT_GMG=[]
      INPUT_GMG.append(0)
      INPUT_GMG.append(DATA_GMG[0])
      INPUT_GMG.append(DATA_GMG[1])
   else:
      INPUT_GMG=INPUT_MENU('mtx_gen',[0])
   if INPUT_GMG[0]==1:
      MENU_CTRL_GMG=(-1)
      print('Limite errori di immissione dati raggiunto')
   elif INPUT_GMG[0]==0:
      MENU_CTRL_GMG=0
      print('Inizializzazione tabelle')
      N_GMG=int(INPUT_GMG[1])
      F_PREC_GMG=int(INPUT_GMG[2])
      DELTA_Z=mp.zeros(N_GMG+2,N_GMG)
      COEF_INT_Z=mp.zeros(N_GMG+2,N_GMG)
      COEF_Z=mp.zeros(N_GMG+1,N_GMG+1)
      DELTA_Z[1,1]=mp.mpf(1)     #
      DELTA_Z[2,1]=mp.mpf(-1)
      DELTA_Z[3,1]=mp.mpf(-0.5)
      DELTA_Z[1,2]=mp.mpf(1)
      DELTA_Z[2,2]=mp.mpf(-1)
      DELTA_Z[3,2]=mp.mpf(-0.75)
      DELTA_Z[4,2]=mp.mpf(-0.25)
      COEF_INT_Z[1,0]=mp.mpf(1)
      COEF_INT_Z[2,0]=mp.mpf(-1)
      COEF_INT_Z[1,1]=mp.mpf(1)
      COEF_INT_Z[2,1]=mp.mpf(-1)
      COEF_INT_Z[3,1]=mp.mpf(-0.5)
      COEF_INT_Z[1,2]=mp.mpf(1)
      COEF_INT_Z[2,2]=mp.mpf(-1)
      COEF_INT_Z[3,2]=mp.mpf(-0.625)
      COEF_INT_Z[4,2]=mp.mpf(-0.125)
      COEF_Z[1,1]=mp.mpf(1)
      COEF_Z[1,2]=mp.mpf(0.5)
      COEF_Z[2,2]=mp.mpf(0.5)
      COEF_Z[1,3]=mp.mpf(3/8)
      COEF_Z[2,3]=mp.mpf(3/8)
      COEF_Z[3,3]=mp.mpf(0.125)
      print('Inizio computazione tabella DELTA')
      DELTA_Z=CALCOLO_DELTA_Z(DELTA_Z,N_GMG-1)
      print('Computazione tabella DELTA terminata')
      TEMP=CALCOLO_INTZ_COEFZ(DELTA_Z,COEF_INT_Z,COEF_Z,N_GMG-1)
      COEF_INT_Z=TEMP[0]
      COEF_Z=TEMP[1]
      COEF_INT_TA=mp.zeros(N_GMG,N_GMG)
```

```
COEF_TA=mp.zeros(N_GMG,N_GMG)
COEF_INT_TA[0,0]=mp.mpf(-1)
COEF_INT_TA[0,1]=mp.mpf(-1)
COEF_INT_TA[1,1]=mp.mpf(-1)
COEF_INT_TA[0,2]=mp.mpf(-1)
COEF_INT_TA[1,2]=mp.mpf(-1)
COEF_INT_TA[2,2]=mp.mpf(-1/4)
COEF_TA[0,0]=mp.mpf(2)
COEF_TA[0,1]=mp.mpf(1)
COEF_TA[1,1]=mp.mpf(2)
COEF_TA[0,2]=mp.mpf(1)
COEF_TA[1,2]=mp.mpf(3/2)
COEF_TA[2,2]=mp.mpf(1/2)

print('Inizio computazione rimanenti tabelle')
w=2
count=1
total_count=0
while w<N_GMG-1:
   w+=1
   COEF_INT_TA[0,w]=-COEF_Z[1,w]
   k=0
   while k<=w-2:
      COEF_INT_TA[0,w]=COEF_INT_TA[0,w]-COEF_Z[w-k,w]*PRODUTTORIA(2,w-
         1-k)
      k+=1
   #end while
   k=1
   while k<w-1:
      COEF_INT_TA[k,w]=-COEF_Z[k,w]/mp.mpf(k)-COEF_Z[k+1,w]
      j=0
      while j<=w-k-2:
         COEF_INT_TA[k,w]=COEF_INT_TA[k,w]-COEF_Z[w-
               j,w]*PRODUTTORIA(k+1,w-1-j)
         j+=1
      #end while
      COEF_TA[k-1,w]=mp.mpf(-2)*COEF_INT_TA[k-1,w]
               +mp.mpf(k)*COEF_INT_TA[k,w]
      k+=1
   #end while
   k=w-1
   COEF_INT_TA[k,w]=-COEF_Z[w,w]-COEF_Z[k,w]/mp.mpf(k)
   COEF_TA[k-1,w]=mp.mpf(-2)*COEF_INT_TA[k-1,w]
         +mp.mpf(k)*COEF_INT_TA[k,w]
   k=w
   COEF_INT_TA[k,w]=-COEF_Z[k,w]/mp.mpf(w)
   COEF_TA[k-1,w]=mp.mpf(-2)*COEF_INT_TA[k-1,w]
         +mp.mpf(k)*COEF_INT_TA[k,w]
   COEF_TA[k,w]=mp.mpf(-2)*COEF_INT_TA[k,w]
   if count==10:
```

```
        total_count+=1
        print('Please wait')
        print('%d cycles missing'%(N_GMG-total_count*10))
        count=1
    else:
        count+=1
    #end if
#end while

COEF_TU=mp.zeros(2,N_GMG)
COEF_TU[0,0]=1
j=1
while j<N_GMG:
    print(j)
    SERV=COEF_Z[1,j]/2
    if j>1:
        m=2
        while m<j+1:
            SERV+=COEF_Z[m,j]*PRODUTTORIA(1,m-1)/mp.power(2,m)
            m+=1
        #end while
    #end if
    COEF_TU[0,j]=1-SERV
    COEF_TU[1,j]=SERV
    j+=1
#end while
print('Inizio elaborazione parametri funzioni globali')
CINT_WAIT_GLBL=mp.zeros(N_GMG,N_GMG)
CINT_WAIT_GLBL[0,0]=(-1)
CDEN_WAIT_GLBL=mp.zeros(N_GMG,N_GMG)
CDEN_WAIT_GLBL[0,0]=1
SERV_MTX=mp.zeros(N_GMG,N_GMG)
SERV_MTX[0,0]=(-1)

j=1
SERV_MTX[0,j]=-COEF_Z[1,1]
SERV_WAIT=0
i=2
while i<j+1:
    m=2
    while m<i+1:
        SERV_WAIT+=PRODUTTORIA(1,m-1)*COEF_Z[m,i]
        m+=1
    #end while
    SERV_WAIT+=COEF_Z[1,i]
    i+=1
#end while
SERV_MTX[0,j]+=SERV_WAIT*(-1)
k=1
while k<j+1:
```

```
    i=k+1
    while i<(j+1)+1:
        m=k+1
        while m<i+1:
            SERV_MTX[k,j]+=PRODUTTORIA(k+1,m-1)*COEF_Z[m,i]
            m+=1
        #end while
        i+=1
    #end while
    SERV_MTX[k,j]=(-1)*SERV_MTX[k,j]
    k+=1
#end while

j=2
while j<N_GMG:
    print(j)
    SERV_MTX[0,j]=SERV_MTX[0,j-1]-COEF_Z[1,j]
    SERV_WAIT=0
    m=2
    while m<j+1:
        SERV_WAIT+=PRODUTTORIA(1,m-1)*COEF_Z[m,j]
        m+=1
    #end while
    SERV_WAIT=SERV_WAIT*(-1)
    SERV_MTX[0,j]+=SERV_WAIT

    k=1
    while k<j+1:
        SERV_MTX[k,j]=SERV_MTX[k,j-1]
        SERV_WAIT=0
        m=k+1
        while m<j+1:
            SERV_WAIT+=PRODUTTORIA(k+1,m-1)*COEF_Z[m,j]
            m+=1
        #end while
        SERV_WAIT=(-1)*SERV_WAIT
        SERV_MTX[k,j]+=SERV_WAIT
        k+=1
    #end while
    j+=1
#end while

j=1
while j<N_GMG:
    for i in range(SERV_MTX.rows):
        CINT_WAIT_GLBL[i,j]=SERV_MTX[i,j]/(j+1)
    #end for i
    j+=1
#end while
```

```
j=1
while j<N_GMG:
   for k in range((j+1)-1):
      CDEN_WAIT_GLBL[k,j]=-CINT_WAIT_GLBL[k,j]+
        (k+1)*CINT_WAIT_GLBL[k+1,j]
   #end for k
   CDEN_WAIT_GLBL[j,j]=-CINT_WAIT_GLBL[j,j]
   j+=1
#end while

CINT_ARR_GLBL=mp.zeros(N_GMG,N_GMG)
CINT_ARR_GLBL[0,0]=-1
SERV_MTX=mp.zeros(N_GMG,N_GMG)
SERV_MTX[0,0]=(-1)
CDEN_ARR_GLBL=mp.zeros(N_GMG,N_GMG)
CDEN_ARR_GLBL[0,0]=2

j=1
SERV_MTX[0,j]=(-1/2)
SERV_ARR=0
i=0
while i<(j+1):
   SERV_ARR+=COEF_TA[0,i]
   i+=1
#end while
SERV_MTX[0,j]=SERV_MTX[0,j]*SERV_ARR
SERV_ARR=0
i=1
while i<(j+1):
   m=1
   while m<(i+1):
      SERV_ARR+=COEF_TA[m,i]*PRODUTTORIA(1,m)/mp.power(2,m+1)
      m+=1
   #end while
   i+=1
#end while
SERV_MTX[0,j]+=SERV_ARR
k=1
while k<(j+1):
   SERV_MTX[k,j]=-1
   SERV_ARR=0
   i=k
   while i<(j+1):
      m=k
      while m<i+1:
         SERV_ARR+=COEF_TA[m,i]*PRODUTTORIA(k+1,m)/mp.power(2,m-k+1)
         m+=1
      #end while
      i+=1
   #end while
```

```
    SERV_MTX[k,j]=SERV_MTX[k,j]*SERV_ARR
    k+=1
#end while

j=2
while j<N_GMG:
    print(j)
    SERV_MTX[0,j]=SERV_MTX[0,j-1]-COEF_TA[0,j]/2
    SERV_ARR=0
    m=1
    while m<j+1:
        SERV_ARR+=COEF_TA[m,j]*PRODUTTORIA(1,m)/mp.power(2,m+1)
        m+=1
    #end while
    SERV_ARR=(-1)*SERV_ARR
    SERV_MTX[0,j]+=SERV_ARR

    k=1
    while k<j+1:
        SERV_MTX[k,j]=SERV_MTX[k,j-1]
        SERV_ARR=0
        m=k
        while m<j+1:
            SERV_ARR+=COEF_TA[m,j]*PRODUTTORIA(k+1,m)/mp.power(2,m-k+1)
            m+=1
        #end while
        SERV_ARR=(-1)*SERV_ARR
        SERV_MTX[k,j]+=SERV_ARR
        k+=1
    #end while
    j+=1
#end while

j=1
while j<N_GMG:
    for i in range(N_GMG):
        CINT_ARR_GLBL[i,j]=SERV_MTX[i,j]/(j+1)
    #end for i
    j+=1
#end while
j=1
while j<N_GMG:
    for k in range((j+1)-1):
        CDEN_ARR_GLBL[k,j]=-2*CINT_ARR_GLBL[k,j]+
          (k+1)*CINT_ARR_GLBL[k+1,j]
    #end for k
    CDEN_ARR_GLBL[j,j]=-2*CINT_ARR_GLBL[j,j]
    j+=1
#end while
```

```
CDEN_EXIT_GLBL=mp.zeros(2,N_GMG)
CDEN_EXIT_GLBL[0,0]=1
j=1
while j<N_GMG:
   print(j)
   i=1
   SERV_EXIT=0
   while i<(j+1):
      SERV_EXIT+=COEF_TU[1,i]
      i+=1
   #end while
   SERV_EXIT=SERV_EXIT/(j+1)
   CDEN_EXIT_GLBL[0,j]=1-SERV_EXIT
   CDEN_EXIT_GLBL[1,j]=SERV_EXIT
   j+=1
#end while

print ('Elaborazione tabelle finita')
print('Salvataggio in corso')
print('Le tabelle verranno salvate nella stessa cartella in cui è presente il file del
        programma\n')

#end if
COEF_INT_TA_print=mp.zeros(COEF_INT_TA.rows+1,COEF_INT_TA.cols)
for j in range(COEF_INT_TA_print.cols):
   COEF_INT_TA_print[0,j]=1
#end for j
for i in range(COEF_INT_TA_print.rows-1):
   for j in range(COEF_INT_TA_print.cols):
      COEF_INT_TA_print[i+1,j]=COEF_INT_TA[i,j]
   #end for j
#end for i
if not(WHO_CALLS=='sched'):
   print('Salvataggio files utente')
   DATA_PRINT_GMG=[N_GMG, F_PREC_GMG, ['int',DELTA_Z,[1,1]],
        ['int',COEF_INT_Z,[1,1]], ['den',COEF_Z,[1,1]], ['int',COEF_INT_TA_print,[0,0]],
        ['den',COEF_TA,[0,0]]]
   PRINT_TO_USER_FILE('mtx_gen',DATA_PRINT_GMG)
#end if
CODE_KEY_GMG=[1,F_DELIMITER,F_PREC_PRG,N_GMG]
PRG_OUT_PRINT_GMG=[]
PRG_OUT_PRINT_GMG.append(DELTA_Z)
PRG_OUT_PRINT_GMG.append(COEF_INT_Z)
PRG_OUT_PRINT_GMG.append(COEF_Z)
PRG_OUT_PRINT_GMG.append(COEF_INT_TA)
PRG_OUT_PRINT_GMG.append(COEF_TA)
PRG_OUT_PRINT_GMG.append(COEF_TU)
PRG_OUT_PRINT_GMG.append(CINT_ARR_GLBL)
PRG_OUT_PRINT_GMG.append(CDEN_ARR_GLBL)
PRG_OUT_PRINT_GMG.append(CINT_WAIT_GLBL)
```

```
    PRG_OUT_PRINT_GMG.append(CDEN_WAIT_GLBL)
    PRG_OUT_PRINT_GMG.append(CDEN_EXIT_GLBL)
    print('Salvataggio files dati')
    PRINT_TO_FILE_PRG(PRG_OUT_PRINT_GMG,CODE_KEY_GMG)
    print('Salvataggio terminato')

    if not(WHO_CALLS=='sched'):
       SERV_GMG=INPUT_MENU('mtx_sav',0) #domanda se voglio mantenere in memoria le
            matrici appena generate
       ERROR_CTRL_GMG=SERV_GMG[0]
    else:
       ERROR_CTRL_GMG=0
    #end if
    if ERROR_CTRL_GMG==1:    #controllo errore in immissione risposta
       print('Limite errori di immissione dati raggiunto')
       input('Le matrici locali non verranno mantenute in memoria')
    else:                    #ramo immissione corretta
       if not(WHO_CALLS=='sched'):
          MENU_CTRL_GMG=SERV_GMG[1]
       else:
          MENU_CTRL_GMG=1
       #end if
       if MENU_CTRL_GMG==2: #controllo se risposta negativa
            MENU_CTRL_GMG=0
       elif MENU_CTRL_GMG==1:              #ramo risposta positiva
          OUT_GMG=[] #resetta OUT_GMG
          OUT_GMG.append(1)
          OUT_GMG.append(DELTA_Z)
          OUT_GMG.append(COEF_INT_Z)
          OUT_GMG.append(COEF_Z)
          OUT_GMG.append(COEF_INT_TA)
          OUT_GMG.append(COEF_TA)
          OUT_GMG.append(COEF_TU)
          OUT_GMG.append(CINT_ARR_GLBL)
          OUT_GMG.append(CDEN_ARR_GLBL)
          OUT_GMG.append(CINT_WAIT_GLBL)
          OUT_GMG.append(CDEN_WAIT_GLBL)
          OUT_GMG.append(CDEN_EXIT_GLBL)
          SAVE_MTX_SUB_GMG=1
       #end if
    #end if
    #FINE OPERAZIONE 'MANTERE IN MEMORIA O MENO'
#end if
if WHO_CALLS=='sched':
    MENU_CTRL_GMG=(-1)
else:
    if not(MENU_CTRL_GMG==(-1)):
       SERV_GMG=INPUT_MENU('er_sub',0)
       ERROR_CTRL_GMG=SERV_GMG[0]
       if ERROR_CTRL_GMG==1:
```

```
                MENU_CONTROL_GMG=(-1)
                print('Limite errori di immissione dati raggiunto')
            else:
                MENU_CTRL_GMG=SERV_GMG[1]
                if MENU_CTRL_GMG==1:
                    MENU_CTRL_GMG=0
                elif MENU_CTRL_GMG==2:
                    MENU_CTRL_GMG=-1
                #end if
            #end if
        #end if
    #end if
  #end while
  if not(WHO_CALLS=='sched'):
    input('Reindirizzamento a menù principale')
  #end if
  return(OUT_GMG)
#end GENERAL_MATRIX_GENERATION()
def CALCOLO_PROBABILITA_WIP(WIP, C_Z, NI):
#Funzione custom adibita al calcolo delle probabilità di avere un determinato livello di WIP alla
#fine di un determinato turno in assenza di buffer.
#Viene richiamata dalla funzione custom TIMELESS.
#In esecuzione richiama la funzione custom PRODUTTORIA.
#Restituisce in output i dati elaborati.
  i=0
  count=0
  while i<NI-1:
    if count==10:
      print('Please wait')
      print('%d to go'%(NI-i))
      count=1
    else:
      count+=1
    #end if
    i=i+1
    WIP[i,0]=C_Z[1,i+1]*mp.mpf(1/2)
    m=2
    while m<=i+1:
      WIP[i,0]+=C_Z[m,i+1]*PRODUTTORIA(1,m-1)*mp.power(mp.mpf(1/2),mp.mpf(m))
      m+=1
    #end while
    j=0
    while j<i+1:
      j+=1
      k=j-1
      while k<i+1:
        WIP[i,j]+=WIP[i-1,k]*mp.power(mp.mpf(1/2),mp.mpf(k-j+2))
        k+=1
      #end while
    #end while
```

```
        SERV=mp.mpf(0)
        k=1
        while k<=i+1:
            SERV+=WIP[i,k]
            k+=1
        #end while
        ALPHA=mp.mpf((1-WIP[i,0])/SERV)
        j=0
        while j<i+1:
            j+=1
            WIP[i,j]=ALPHA*WIP[i,j]
        #end while
    #end while
    i=NI
    j=0
    WIP[i,0]=mp.mpf(0)
    while j<i+1:
        j+=1
        WIP[i,j]=WIP[i-1,j-1]
    #end while
    return (WIP)
#end CALCOLO_PROBABILITA_WIP

def CALCOLO_PROB_WIP_BUFFER(WIP_BUFF_CPWB):
#Funzione custom deputata all'elaborazione delle percentuali riguardanti il livello di WIP atteso al
#termine di ogni "turno" in un caso specifico in presenza di buffer.
#Viene chiamata solo dalla funzione custom BUFFER_ELABORATION.
#Restituisce alla funzione chiamante la matrice che le contiene.
    BUFF_CPWB=WIP_BUFF_CPWB.cols-3
    N_CPWB=WIP_BUFF_CPWB.rows-1
    i=BUFF_CPWB+1
    count=0
    while i<=N_CPWB:
        if count==10:
            print('Please wait')
            print('%d to go'%(N_CPWB-i))
            count=1
        else:
            count+=1
        #end if
        k=0
        while k<=BUFF_CPWB:
            WIP_BUFF_CPWB[i,0]+=WIP_BUFF_CPWB[i-1,k]*
                    mp.power(mp.mpf(1/2),mp.mpf(k+1))
            k+=1
        #end while
        j=1
        while j<=BUFF_CPWB:
            k=j-1
            while k<=BUFF_CPWB:
```

```
            WIP_BUFF_CPWB[i,j]+=WIP_BUFF_CPWB[i-1,k]*mp.power(mp.mpf(1/2),mp.mpf(k-
                j+2))
            k+=1
          #end while
          j+=1
        #end while
        WIP_BUFF_CPWB[i,BUFF_CPWB+1]=WIP_BUFF_CPWB[i-1, BUFF_CPWB]
          *mp.mpf(1/2)
        WIP_BUFF_CPWB[i,BUFF_CPWB+2]=mp.mpf(1)
        for k in range(WIP_BUFF_CPWB.cols-1):
          WIP_BUFF_CPWB[i,BUFF_CPWB+2]-=WIP_BUFF_CPWB[i,k]
        #end for k
        i+=1
    #end while
    return(WIP_BUFF_CPWB)
#end CALCOLO_PROB_WIP_BUFFER

def CALCOLO_PARAMETRI_BUFF(WIP_BUFF_CPB, LAMBDA_CPB, TV_PREV,
BUFF_CPB):
#Funzione custom deputata al calcolo dei parametri di modellizzazione di un caso specifico in
#presenza di buffer. Viene chiamata solo dalla funzione custom BUFFER_ELABORATION, a cui il
#risultato prodotto verrà restituito in forma di array.
    TV_CPB=TV_PREV.copy()   #crea fisicamente una copia in memoria, nn usa puntatori
    N_TO_STOP_CPB=mp.mpf(0)
    SERV_CPB=mp.mpf(0)
    COL=TV_CPB.cols
    BUFF=WIP_BUFF_CPB.cols-3
    N_CPB=WIP_BUFF_CPB.rows-2
    i=BUFF_CPB
    while i<=N_CPB-1:
        N_TO_STOP_CPB+=mp.mpf(i+1)*WIP_BUFF_CPB[i,BUFF_CPB+1]
        SERV_CPB+=WIP_BUFF_CPB[i,BUFF_CPB+1]
        i+=1
    #end while
    N_TO_STOP_CPB=int(N_TO_STOP_CPB/SERV_CPB)+1

    CT_CPB=0
    TUG_CPB=0
    SERV2_CPB=0
    SERV_TIME_CT=0
    SERV_TIME_TU=0
    for i in range(BUFF_CPB):
        SERV_TIME_CT+=TV_CPB[2,i+1]
        SERV_TIME_TU+=TV_CPB[4,i]
    #end for i
    i=BUFF_CPB
    while i<(WIP_BUFF_CPB.rows-2):
        SERV_TIME_CT+=TV_CPB[2,i+1]
        CT_CPB+=((SERV_TIME_CT)/(i+1))*WIP_BUFF_CPB[i,BUFF_CPB+1]
```

```
        SERV_TIME_TU+=TV_CPB[4,i]
        TUG_CPB+=(SERV_TIME_TU/(i+1))*WIP_BUFF_CPB[i,BUFF_CPB+1]

        SERV2_CPB+=WIP_BUFF_CPB[i,BUFF_CPB+1]
        i+=1
    #end while
    SERV_TIME_CT+=TV_CPB[2,TV_CPB.cols-1]
    CT_CPB+=(SERV_TIME_CT/(TV_CPB.cols-1))*(1-WIP_BUFF_CPB[WIP_BUFF_CPB.rows-
        2,BUFF_CPB+2])

    SERV_TIME_TU+=TV_CPB[4,TV_CPB.cols-2]
    TUG_CPB+=(SERV_TIME_TU/(TV_CPB.cols-1))*(1-
        WIP_BUFF_CPB[WIP_BUFF_CPB.rows-2,BUFF_CPB+2])

    SERV2_CPB+=(1-WIP_BUFF_CPB[WIP_BUFF_CPB.rows-2,BUFF_CPB+2])
    CT_CPB=CT_CPB/SERV2_CPB
    TUG_CPB=TUG_CPB/SERV2_CPB
    TH_CPB=1/TUG_CPB
    WIP_CPB=CT_CPB/TUG_CPB

    OUT_CPB=mp.zeros(1,4)
    OUT_CPB[0,0]=WIP_CPB
    OUT_CPB[0,1]=TH_CPB
    OUT_CPB[0,2]=CT_CPB
    OUT_CPB[0,3]=N_TO_STOP_CPB
    return(OUT_CPB)
#end CALCOLO_PARAMETRI_BUFF

def BUFFER_ELABORATION(WHO_CALLS,MTX_ARRAY):
#Funzione custom adibita alla modellizzazione dei parametri di un caso specifico in presenza di
#buffer.
#Può operare solo quando i dati della modellizzazione riguardante il medesimo caso in assenza di
#buffer sono già presenti in memoria.
#Può essere richiamata sia dalla modalità utente che dallo schedulatore.
#In modalità utente, i dati richiesti saranno richiesti tramite la funzione custom INPUT_MENU.
#Durante l'esecuzione dei calcoli verranno richiamate le funzioni custom
#CALCOLO_PROB_WIP_BUFFER, per il calcolo delle percentuali riguardanti il livello di WIP
#atteso alla fine di ogni "turno", e  CALCOLO_PARAMETRI_BUFF per l'elaborazione finale.
#Stampa l'output su file tramite la funzione custom PRINT_TO_USER_FILE.
    OUT_BFF=[]
    OUT_BFF.append(0)
    if WHO_CALLS=='new_set' or ((WHO_CALLS=='main' or WHO_CALLS=='sched' or
        WHO_CALLS=='comp') and MTX_ARRAY[0]==0):
        N_BFF=MTX_ARRAY[1][0]
        T_UPSTREAM_BFF=MTX_ARRAY[1][1]
        METER_BFF=MTX_ARRAY[1][2]
        TV_BFF=MTX_ARRAY[2]
        TV_MED_BFF=MTX_ARRAY[3]
        WIP_NB_BFF=MTX_ARRAY[4]
        GLOBAL_BFF=MTX_ARRAY[5]
```

```
    NB_PARAMETERS_BFF=MTX_ARRAY[6]
    BUFF_CTRL=0
elif WHO_CALLS=='main' and MTX_ARRAY[0]==1:
    print('Nessuna matrice di base presente in memoria')
    print('Generarne di nuove o caricarne di passate nel menù principale')
    BUFF_CTRL=-1
elif WHO_CALLS=='sched' and MTX_ARRAY[0]==0:
    print('Nessuna matrice di base presente in memoria')
    print('Programma di schedulazione errato')
    BUFF_CTRL=-1
elif WHO_CALLS=='comp' and MTX_ARRAY[0]==0:
    print('Nessuna matrice di base presente in memoria')
    print('Errore interno programma comparazione')
    BUFF_CTRL=-1
else:
    print('BUFFER ELABORATION who_calls selection wrong')
#end if
if BUFF_CTRL==0:
    LAMBDA_BFF=1/T_UPSTREAM_BFF
    if WHO_CALLS=='sched' or WHO_CALLS=='comp':
        BUFFER=int(MTX_ARRAY[7])
        F_PREC_BFF=int(MTX_ARRAY[8])
        BUFF_CTRL=0
    else:
        SERV_BUFF=INPUT_MENU('buff_gen',[N_BFF])    #input dati per buffer
        if SERV_BUFF[0]==1:    #controllo se input immesso correttamente
            input('Operazione interrotta causa raggiunto limite errori in immissione dati')
            BUFF_CTRL=-1
        else:              #ramo risposta positiva
            BUFF_CTRL=0
            BUFFER=SERV_BUFF[1]
            F_PREC_BFF=SERV_BUFF[2]
        #end if
    #end if
if BUFF_CTRL==0:
    print('Inizio calcoli')
    WIP_BUFF=mp.zeros(N_BFF+1,BUFFER+3)
    for i in range(BUFFER+1):
        for j in range (BUFFER+2):
            WIP_BUFF[i,j]=WIP_NB_BFF[i,j]
        #end for j
    #end for i
    WIP_BUFF=CALCOLO_PROB_WIP_BUFFER(WIP_BUFF)
    BUFF_PARAMETERS=CALCOLO_PARAMETRI_BUFF(WIP_BUFF,
      LAMBDA_BFF,TV_BFF,BUFFER)


    F_NAME_BFF='%du%s%s%dbuff'%
      (N_BFF,mp.nstr(T_UPSTREAM_BFF,F_PREC_BFF),METER_BFF,BUFFER)
    DATA_BFF=[F_NAME_BFF, N_BFF, F_PREC_BFF, BUFFER, WIP_BUFF,
```

```
            BUFF_PARAMETERS, TV_BFF, TV_MED_BFF, GLOBAL_BFF,
            NB_PARAMETERS_BFF, T_UPSTREAM_BFF, METER_BFF]
        PRINT_TO_USER_FILE('buff_gen', DATA_BFF)
        print('Calcoli e salvataggio eseguiti')
        OUT_BFF=[]
        OUT_BFF.append(1)
        OUT_BFF.append([N_BFF,T_UPSTREAM_BFF,METER_BFF])
        OUT_BFF.append(TV_BFF)
        OUT_BFF.append(TV_MED_BFF)
        OUT_BFF.append(WIP_NB_BFF)
        OUT_BFF.append(GLOBAL_BFF)
        OUT_BFF.append(NB_PARAMETERS_BFF)
        OUT_BFF.append(BUFFER)
        OUT_BFF.append(BUFF_PARAMETERS)
    else:
        OUT_BFF=[]
        OUT_BFF.append(0)
    #end if
    if not(WHO_CALLS=='sched' or WHO_CALLS=='comp') and WHO_CALLS=='main':
        input('Reindirizzamento a menù principale')
    #end if
    return(OUT_BFF)
#end def BUFFER_ELABORATION(WHO_CALLS,MTX_ARRAY)




def CALCOLO_TA_VA(CT,n):
#Funzione custom adibita al calcolo dei parametri delle distribuzioni di probabilità dei tempi di
#arrivo.
#Viene richiamata da TIMELESS.
#In esecuzione richiama la funzione custom PRODUTTORIA.
#Restituisce in output i dati elaborati.
    k=0
    T_int=mp.mpf(0)
    V_int=mp.mpf(0)
    while k<=n:
        T_int=T_int+CT[k,n]*PRODUTTORIA(mp.mpf(1),mp.mpf(k+1))*
            mp.power(mp.mpf(1/2),mp.mpf(k+2))
        V_int=V_int+CT[k,n]*PRODUTTORIA(mp.mpf(1),mp.mpf(k+2))*
            mp.power(mp.mpf(1/2),mp.mpf(k+3))
        k+=1
    #end while
    V_int=V_int-mp.power(T_int,mp.mpf(2))
    OUT=[T_int, V_int]
    return(OUT)
#end CALCOLO_TA_VA

def CALCOLO_TE_VE(M,n):
```

```
#Funzione custom adibita al calcolo dei parametri delle distribuzioni di probabilità dei tempi di
#attesa.
#Viene richiamata da TIMELESS.
#In esecuzione richiama la funzione custom PRODUTTORIA.
#Restituisce in output i dati elaborati.
   A_int=mp.mpf(0.0)
   VA_int=mp.mpf(0.0)
   k=0
   while k<n:
      k+=1
      A_int=A_int+M[k,n]*PRODUTTORIA(1,k)
      VA_int=VA_int+M[k,n]*PRODUTTORIA(1,k+1)
   #end while
   VA_int=VA_int-mp.power(A_int,mp.mpf(2))
   OUT=[A_int, VA_int]
   return(OUT)
#end CALCOLO_TE_VE

def CALCOLO_TU_VU(CFU,w):
#Funzione custom adibita al calcolo dei parametri delle distribuzioni di probabilità dei tempi di
#uscita.
#Viene richiamata da TIMELESS.
#Restituisce in output i dati elaborati.
   k=0
   T_int=(CFU[1,w]+1)
   V_int=(2+4*CFU[1,w])
   V_int=V_int-mp.power(T_int,2)
   OUT=[T_int, V_int]
   return(OUT)
#end CALCOLO_TU_VU
def GLOBAL_CALC(N_GLBL, CDEN_ARR_GLBL, CDEN_WAIT_GLBL,
CDEN_EXIT_GLBL):
#Funzione adibita al calcolo dei coeficienti delle varie distribuzioni di probabilità globali con
#LAMBDA=1.
#Viene richiamata dalla funzione custom TIMELESS.
#In esecuzione eseguirà la funzione custom PRODUTTORIA.
#Restituisce in output i dati elaborati.
   WAIT_MED_GLBL=0
   N_GLBL+=1
   for k in range(N_GLBL):
      SERV_WAIT=PRODUTTORIA(1,k+1)
      WAIT_MED_GLBL+=SERV_WAIT*CDEN_WAIT_GLBL[k,N_GLBL-1]
   #end for k
   WAIT_VAR_GLBL=0
   for k in range(N_GLBL):
      SERV_WAIT=PRODUTTORIA(1,k+2)
      WAIT_VAR_GLBL+=SERV_WAIT*CDEN_WAIT_GLBL[k,N_GLBL-1]
   #end for k
   WAIT_VAR_GLBL=WAIT_VAR_GLBL-mp.power(WAIT_MED_GLBL,2)
   TARR_MED_GLBL=0
```

```
    for k in range(N_GLBL):
        TARR_MED_GLBL+=CDEN_ARR_GLBL[k,
            N_GLBL-1]*(PRODUTTORIA(1,k+1)/mp.power(2,k+2))
    #end for k
    TARR_VAR_GLBL=0
    for k in range(N_GLBL):
        TARR_VAR_GLBL+=CDEN_ARR_GLBL[k,N_GLBL-1]
            *(PRODUTTORIA(1,k+2)/mp.power(2,k+3))
    #end for k
    TARR_VAR_GLBL-=mp.power(TARR_MED_GLBL,2)

    TEXIT_MED_GLBL=(1+CDEN_EXIT_GLBL[1,N_GLBL-1])
    TEXIT_VAR_GLBL=(2+4*CDEN_EXIT_GLBL[1,N_GLBL-1])-
            mp.power(TEXIT_MED_GLBL,2)

    OUT=mp.zeros(1,6)
    OUT[0,0]=TARR_MED_GLBL
    OUT[0,1]=TARR_VAR_GLBL
    OUT[0,2]=WAIT_MED_GLBL
    OUT[0,3]=WAIT_VAR_GLBL
    OUT[0,4]=TEXIT_MED_GLBL
    OUT[0,5]=TEXIT_VAR_GLBL
    return(OUT)
#END def GLOBAL_CALC()

def TIMELESS(WHO_CALLS,DATA_TL):
#Funzione custom adibita al calcolo di alcune delle matrici contenenti i coeficienti, i valori medi ed
#altri parametri del modellizzatore per Lambda=1.
#Può esser richiamata dalla funzione custom USER_THEORY_MAIN e dallo schedulatore.
#In modalità utente, gli input necessari verranno richiesti tramite la funzione INPUT_MENU.
#In esecuzione richiamerà le funzioni custom CALCOLO_TA_VA, CALCOLO_TE_VE,
#CALCOLO_TU_VU, GLOBAL_CALC, CALCOLO_PROBABILITA_WIP.
#Per la creazione e salvataggio dei dati su file verranno richiamate le funzioni
#PRINT_TO_FILE_PRG e PRINT_TO_USER_FILE.
#Restituirà in output i dati generati.
    OUT_TL=[]
    OUT_TL.append(0)
    CTRL_TL=0
    if DATA_TL[0]==1:
        print('Nessuna matrice di base presente in memoria')
        print('Generare/Caricare in memoria dal menù principale')
        CTRL_TL=-1
    else:
        CTRL_TL=0
        DELTA_Z=DATA_TL[1]
        COEF_INT_Z=DATA_TL[2]
        COEF_Z=DATA_TL[3]
        COEF_INT_TA=DATA_TL[4]
        COEF_TA=DATA_TL[5]
        N_MTX=DATA_TL[6]
```

```
    COEF_TU=DATA_TL[7]
    CINT_ARR_GL=DATA_TL[8]
    CDEN_ARR_GL=DATA_TL[9]
    CINT_WAIT_GL=DATA_TL[10]
    CDEN_WAIT_GL=DATA_TL[11]
    CDEN_EXIT_GL=DATA_TL[12]
    if WHO_CALLS=='sched':
       N_TL=int(DATA_TL[13])
       PREC_TL=int(DATA_TL[14])
    #end if
 #end if
 while CTRL_TL==0:
    if not(WHO_CALLS=='sched'):
       SERV_TL=INPUT_MENU('timeless_gen',[N_MTX])
       if SERV_TL[0]==1:
          CTRL_TL=(-1)
          print('Limite errori di immissione dati raggiunto')
       else:
          MENU_CTRL_TL=0
          N_TL=int(SERV_TL[1])
          PREC_TL=SERV_TL[2]
       #end if
    #end if
    if CTRL_TL==0:
       F_NAME_TL='%du%dd'%(N_TL,PREC_TL)
       print ('Inizio calcoli tempi medi e varianze specifici della coppia')
       TIME_VAR_TL=mp.zeros(6,N_TL+1)

       TA_TL=[]
       VA_TL=[]
       TE_TL=[]
       VE_TL=[]
       TU_TL=[]
       VU_TL=[]
       for i in range(N_TL):
          TA_TL.append(mp.mpf(0))
          VA_TL.append(mp.mpf(0))
          TE_TL.append(mp.mpf(0))
          VE_TL.append(mp.mpf(0))
          TU_TL.append(mp.mpf(0))
          VU_TL.append(mp.mpf(0))
       #end for i
       TE_TL.append(mp.mpf(0))
       VE_TL.append(mp.mpf(0))

       SERV_TL=CALCOLO_TA_VA(COEF_TA,0)
       TA_TL[0]=SERV_TL[0]
       VA_TL[0]=SERV_TL[1]
       TU_TL[0]=1
       VU_TL[0]=1
```

```
w=1
count=1
while w<N_TL:
   SERV_TL=CALCOLO_TA_VA(COEF_TA,w)
   TA_TL[w]=SERV_TL[0]
   VA_TL[w]=SERV_TL[1]
   SERV_TL=CALCOLO_TE_VE(COEF_Z,w)
   TE_TL[w]=SERV_TL[0]
   VE_TL[w]=SERV_TL[1]
   SERV_TL=CALCOLO_TU_VU(COEF_TU,w)
   TU_TL[w]=SERV_TL[0]
   VU_TL[w]=SERV_TL[1]
   w+=1
   if count==10:
      print('Please wait')
      print('%d to go'%(N_TL-w))
      count=1
   else:
      count+=1
   #end if
#end while
w=N_TL
SERV_TL=CALCOLO_TE_VE(COEF_Z,w)
TE_TL[w]=SERV_TL[0]
VE_TL[w]=SERV_TL[1]

for i in range(len(TA_TL)):
   TIME_VAR_TL[0,i]=TA_TL[i]
   TIME_VAR_TL[1,i]=VA_TL[i]
   TIME_VAR_TL[2,i]=TE_TL[i]
   TIME_VAR_TL[3,i]=VE_TL[i]
   TIME_VAR_TL[4,i]=TU_TL[i]
   TIME_VAR_TL[5,i]=VU_TL[i]
#end for i
TIME_VAR_TL[2,len(TE_TL)-1]=TE_TL[len(TE_TL)-1]
TIME_VAR_TL[3,len(VE_TL)-1]=VE_TL[len(TE_TL)-1]
print ('Calcoli tempi medi e varianze specifici della coppia terminati')

print ('Inizio calcoli media dei tempi medi e delle varianze su N coppie')
TIME_VAR_MED_TL=mp.zeros(6,N_TL)
count=1
TIME_VAR_MED_TL[0,0]=TIME_VAR_TL[0,0]
TIME_VAR_MED_TL[1,0]=TIME_VAR_TL[1,0]
TIME_VAR_MED_TL[2,0]=TIME_VAR_TL[2,1]
TIME_VAR_MED_TL[3,0]=TIME_VAR_TL[3,1]
TIME_VAR_MED_TL[4,0]=TIME_VAR_TL[4,0]
TIME_VAR_MED_TL[5,0]=TIME_VAR_TL[5,0]
j=1
while j<N_TL:
```

```
    if count==10:
        print('Please wait')
        print('%d to go'%(N_TL-j))
        count=1
    else:
        count+=1
    #end if
    TIME_VAR_MED_TL[0,j]=TIME_VAR_TL[0,j]+TIME_VAR_MED_TL[0,j-1]
    TIME_VAR_MED_TL[1,j]=TIME_VAR_TL[1,j]+TIME_VAR_MED_TL[1,j-1]
    TIME_VAR_MED_TL[2,j]=TIME_VAR_TL[2,j+1]+TIME_VAR_MED_TL[2,j-1]
    TIME_VAR_MED_TL[3,j]=TIME_VAR_TL[3,j+1]+TIME_VAR_MED_TL[3,j-1]
    TIME_VAR_MED_TL[4,j]=TIME_VAR_TL[4,j]+TIME_VAR_MED_TL[4,j-1]
    TIME_VAR_MED_TL[5,j]=TIME_VAR_TL[5,j]+TIME_VAR_MED_TL[5,j-1]
    j+=1
#end while
for j in range(N_TL):
    TIME_VAR_MED_TL[0,j]=TIME_VAR_MED_TL[0,j]/(j+1)
    TIME_VAR_MED_TL[1,j]=TIME_VAR_MED_TL[1,j]/(j+1)
    TIME_VAR_MED_TL[2,j]=TIME_VAR_MED_TL[2,j]/(j+1)
    TIME_VAR_MED_TL[3,j]=TIME_VAR_MED_TL[3,j]/(j+1)
    TIME_VAR_MED_TL[4,j]=TIME_VAR_MED_TL[4,j]/(j+1)
    TIME_VAR_MED_TL[5,j]=TIME_VAR_MED_TL[5,j]/(j+1)
#end for j
print ('Calcoli media dei tempi medi e delle varianze su N coppie terminati')
print ('Inizio calcoli dei tempi medi e delle varianze globali su N coppie')
GLOBAL_TL=mp.zeros(6,N_TL)
GLOBAL_TL[0,0]=mp.mpf(0.5)
GLOBAL_TL[1,0]=mp.mpf(0.25)
GLOBAL_TL[2,0]=mp.mpf(1)
GLOBAL_TL[3,0]=mp.mpf(1)
GLOBAL_TL[4,0]=mp.mpf(1)
GLOBAL_TL[5,0]=mp.mpf(1)

count=1
for j in range(N_TL-1):
    if count==10:
        print('Please wait')
        print('%d to go'%(N_TL-j))
        count=1
    else:
        count+=1
    #end if
    SERV_TL=GLOBAL_CALC(j+1, CDEN_ARR_GL, CDEN_WAIT_GL,
          CDEN_EXIT_GL)
    for i in range(6):
        GLOBAL_TL[i,j+1]=SERV_TL[i]
    #end for i
#end for j
print ('Calcoli dei tempi medi e delle varianze globali su N coppie terminati')
```

```
print('Inizio elaborazione tabella probabilità WIP senza buffer')
WIP_NO_BUFF_TL=mp.zeros((N_TL),(N_TL+1))
WIP_NO_BUFF_TL[0,0]=mp.mpf(0.5)
WIP_NO_BUFF_TL[0,1]=mp.mpf(0.5)
WIP_NO_BUFF_TL=CALCOLO_PROBABILITA_WIP(WIP_NO_BUFF_TL, COEF_Z,
        N_TL-1)
print('Elaborazione tabella probabilità WIP senza buffer terminata')

print('Inizio elaborazione parametri timeless')
TEMPO_TOT_TL=[]
TEMPO_TOT_TL.append(TIME_VAR_TL[0,0]+TIME_VAR_TL[2,1])
i=1
while i<N_TL:
   TEMPO_TOT_TL.append(0)
   j=0
   while j<i+1:
      TEMPO_TOT_TL[i]+=TIME_VAR_TL[0,j]
      j+=1
   #end for j
   TEMPO_TOT_TL[i]+=TIME_VAR_TL[2,i+1]
   i+=1
#end for i

NOBUFF_PARAMETERS_TL=mp.zeros(11,N_TL)
count=1
for z in range(N_TL):
   if count==10:
      print('Please wait')
      print('%d to go'%(N_TL-z))
      count=1
   else:
      count+=1
   #end if
   for i in range(z):
      SERV_TL=mp.mpf(0)
      for j in range(i+2):
         SERV_TL+=WIP_NO_BUFF_TL[i,j]*mp.mpf(j)
      #end for j
      SERV_TL=SERV_TL*TIME_VAR_TL[0,i+1]
      NOBUFF_PARAMETERS_TL[0,z]+=SERV_TL
   #end for i
   NOBUFF_PARAMETERS_TL[0,z]=NOBUFF_PARAMETERS_TL[0,z]/
        TEMPO_TOT_TL[z]
   SERV_TL=mp.mpf(0)
   j=1
   while j<=z+1:
      SERV_TL+=WIP_NO_BUFF_TL[z,j]*mp.mpf(j/2)
      j+=1
   #end while
   SERV_TL=SERV_TL*TIME_VAR_TL[2,z+1]/TEMPO_TOT_TL[z]
```

```
            NOBUFF_PARAMETERS_TL[0,z]+=SERV_TL

            NOBUFF_PARAMETERS_TL[1,z]=(1/TIME_VAR_MED_TL[0,z])*
                TIME_VAR_MED_TL[2,z]
            NOBUFF_PARAMETERS_TL[2,z]=(1/TIME_VAR_MED_TL[4,z])*
                TIME_VAR_MED_TL[2,z]
            NOBUFF_PARAMETERS_TL[3,z]=(1/GLOBAL_TL[4,z])*GLOBAL_TL[2,z]

            NOBUFF_PARAMETERS_TL[4,z]=NOBUFF_PARAMETERS_TL[0,z]/
                TIME_VAR_MED_TL[2,z]
            NOBUFF_PARAMETERS_TL[5,z]=1/TIME_VAR_MED_TL[0,z]
            NOBUFF_PARAMETERS_TL[6,z]=1/TIME_VAR_MED_TL[4,z]
            NOBUFF_PARAMETERS_TL[7,z]=1/GLOBAL_TL[4,z]
            NOBUFF_PARAMETERS_TL[8,z]=TIME_VAR_MED_TL[2,z]
            NOBUFF_PARAMETERS_TL[9,z]=NOBUFF_PARAMETERS_TL[0,z]/
                NOBUFF_PARAMETERS_TL[6,z]
            NOBUFF_PARAMETERS_TL[10,z]=GLOBAL_TL[2,z]
        #end for i
        print('Elaborazione parametri timeless terminata')

        print('Salvataggio in corso')
        CODE_KEY_TL=[5,F_DELIMITER,F_PREC_PRG,N_TL]
        PRG_OUT_PRINT_TL=[TIME_VAR_TL, TIME_VAR_MED_TL, WIP_NO_BUFF_TL,
            GLOBAL_TL, NOBUFF_PARAMETERS_TL]
        PRINT_TO_FILE_PRG(PRG_OUT_PRINT_TL,CODE_KEY_TL)
        DATA_PRINT_USER_TL=[[TIME_VAR_TL, TIME_VAR_MED_TL, GLOBAL_TL,
            NOBUFF_PARAMETERS_TL, WIP_NO_BUFF_TL], N_TL, PREC_TL]
        PRINT_TO_USER_FILE('timeless', DATA_PRINT_USER_TL)
        #end if
        print('Salvataggio terminato')

        if not(WHO_CALLS=='sched'):
            SERV_TL=INPUT_MENU('er_sub',0)
            if SERV_TL[0]==1:   #
                CTRL_TL=-1
                print('Limite errori di immissione dati raggiunto')
            else:
                if SERV_TL[1]==1:
                    CTRL_TL=0
                else:
                    CTRL_TL=-1
                #end if
            #end if
        else:
            CTRL_TL=-1
        #end if
    #end if
#end while
if not(WHO_CALLS=='sched'):
    SERV_TL=INPUT_MENU('timeless_sav',[0])
```

```
        if SERV_TL[0]==1:
            print('Limite errori di immissione dati raggiunto')
            input('I dati di elaborazione non verranno mantenuti in memoria')
            OUT_TL=[]
            OUT_TL.append(0)
        else:
            if SERV_TL[0]==2:
                OUT_TL=[]
                OUT_TL.append(0)
            else: #ramo risposta positiva
                OUT_TL=[]
                OUT_TL.append(1)
            #end if
    else:
        OUT_TL=[]
        OUT_TL.append(1)
    #end if
    if OUT_TL[0]==1:
        OUT_TL.append(N_TL)
        OUT_TL.append(TIME_VAR_TL)
        OUT_TL.append(TIME_VAR_MED_TL)
        OUT_TL.append(WIP_NO_BUFF_TL)
        OUT_TL.append(GLOBAL_TL)
        OUT_TL.append(NOBUFF_PARAMETERS_TL)
    #end if
    if not(WHO_CALLS=='sched'):
        print('Reindirizzamento a menù precedente')
    #end if
    return(OUT_TL)
#END def TIMELESS(WHO_CALLS,DATA_TL):

def NEW_SET_ELABORATION(WHO_CALLS, DATA_NSE):
#Funzione custom deputata al calcolo dei parametri della modellizzazione di un caso specifico in
#assenza di buffer.
#Viene richiamato in modalità utente dalle funzioni custom USER_THEORY_MAIN e
#COMP_TABS dopo la selezione dell'operazione di comparazione. Può anche essere eseguito in
#modalità schedulatore.
#In modalità utente l'inserimento dei dati avviene tramite la funzione custom INPUT_MENU.
#Il caricamento dei file necessari, quando necessario, verrà eseguito dalla funzione custom
#GENERAL_MATRIX_LOADING.
#Durante il salvataggio dei dati su file verranno richiamate le funzioni custom
PRINT_TO_FILE_PRG, per la creazione dei file dati interni al programma, e
PRINT_TO_USER_FILE per quelli in formato leggibile in EXCEL.
    OUT_NSE=[]
    OUT_NSE.append(0)
    MENU_CTRL_NSE=0
    if DATA_NSE[0]==1:
        print('Nessuna tabulato timeless presente in memoria')
        print('Generare/Caricare in memoria dal menù principale')
        MENU_CTRL_NSE=-1
```

```
else:
   MENU_CTRL_NSE=0
   N_TAB_TL_NSE=DATA_NSE[1]
   TIME_VAR_TL_NSE=DATA_NSE[2]
   TIME_VAR_MED_TL_NSE=DATA_NSE[3]
   WIP_NO_BUFF_TL_NSE=DATA_NSE[4]
   GLOBAL_TL_NSE=DATA_NSE[5]
   NOBUFF_PARAMETERS_TL_NSE=DATA_NSE[6]
   if WHO_CALLS=='sched':
      N_NSE=DATA_NSE[7]
      T_UPSTREAM_NSE=mp.mpf(DATA_NSE[8])
      METER_NSE=DATA_NSE[9]
      F_PREC_NSE=DATA_NSE[10]
   #end if
#end if
while MENU_CTRL_NSE==0:
   if WHO_CALLS=='nobuff':
      SERV_NSE=INPUT_MENU('tab_gen',[N_TAB_TL_NSE])
      if SERV_NSE[0]==1:
         MENU_CTRL_NSE=(-1)
         print('Limite errori di immissione dati raggiunto')
      else:
         MENU_CTRL_NSE=0
         N_NSE=SERV_NSE[1]
         T_UPSTREAM_NSE=mp.mpf(SERV_NSE[2])
         METER_NSE=SERV_NSE[3]
         F_PREC_NSE=SERV_NSE[4]
      #end if
   #end if
   if MENU_CTRL_NSE==0:
      if METER_NSE=='s' or METER_NSE=='S' or METER_NSE=='sec':
         METER_NSE='sec'
      elif METER_NSE=='m' or METER_NSE=='M' or METER_NSE=='min':
         METER_NSE='min'
      else:
         METER_NSE='h'
      #end if
      LAMBDA_NSE=1/T_UPSTREAM_NSE
      LAMBDA_QUAD_NSE=mp.power(LAMBDA_NSE,2)
      F_NAME_NSE='%du%s%s'%(N_NSE,mp.nstr
               (T_UPSTREAM_NSE,F_PREC_NSE),METER_NSE)

      print('Elaborazione tempi medi e varianze in corso')
      TIME_VAR=mp.zeros(6,N_NSE+1)


      for i in range(3):
         for j in range(N_NSE):
            TIME_VAR[i*2,j]=TIME_VAR_TL_NSE[i*2,j]/LAMBDA_NSE
            TIME_VAR[(i*2)+1,j]=TIME_VAR_TL_NSE[(i*2)+1,j]/LAMBDA_QUAD_NSE
```

```
    #end for j
#end for i
TIME_VAR[2,N_NSE]=TIME_VAR_TL_NSE[2,N_NSE]/LAMBDA_NSE
TIME_VAR[3,N_NSE]=TIME_VAR_TL_NSE[3,N_NSE]/LAMBDA_QUAD_NSE

TIME_VAR_MED=mp.zeros(1,6)
GLOBAL_NSE=mp.zeros(1,6)
for j in range(3):
    TIME_VAR_MED[0,j*2]=TIME_VAR_MED_TL_NSE[j*2,N_NSE-1]/LAMBDA_NSE
    TIME_VAR_MED[0,(j*2)+1]=TIME_VAR_MED_TL_NSE[(j*2)+1, N_NSE-
        1]/LAMBDA_QUAD_NSE
    GLOBAL_NSE[0,j*2]=GLOBAL_TL_NSE[j*2,N_NSE-1]/LAMBDA_NSE
    GLOBAL_NSE[0,(j*2)+1]=GLOBAL_TL_NSE[(j*2)+1,N_NSE-
        1]/LAMBDA_QUAD_NSE
#end for j
print('Elaborazione tempi medi e varianze terminata')

print('Calcolo Matrice probabilità WIP senza buffer in corso')
WIP_NO_BUFF=mp.zeros((N_NSE),(N_NSE+1))
for i in range(WIP_NO_BUFF.rows):
    for j in range(WIP_NO_BUFF.cols):
        WIP_NO_BUFF[i,j]=WIP_NO_BUFF_TL_NSE[i,j]
    #end for j
#end for i
print('Calcolo Matrice probabilità WIP senza buffer in corso terminato')
print('Elaborazione tempi medi e varianze globali in corso')

NOBUFF_PARAMETERS=mp.zeros(5,3)

NOBUFF_PARAMETERS[0,0]=NOBUFF_PARAMETERS_TL_NSE[0,N_NSE-1]
NOBUFF_PARAMETERS[0,1]=NOBUFF_PARAMETERS_TL_NSE[4,N_NSE-
        1]*LAMBDA_NSE
NOBUFF_PARAMETERS[0,2]=TIME_VAR_MED[0,2]

NOBUFF_PARAMETERS[1,0]=NOBUFF_PARAMETERS_TL_NSE[1,N_NSE-1]
NOBUFF_PARAMETERS[1,1]=NOBUFF_PARAMETERS_TL_NSE[5,N_NSE-
        1]*LAMBDA_NSE
NOBUFF_PARAMETERS[1,2]=TIME_VAR_MED[0,2]

NOBUFF_PARAMETERS[2,0]=NOBUFF_PARAMETERS_TL_NSE[2,N_NSE-1]
NOBUFF_PARAMETERS[2,1]=NOBUFF_PARAMETERS_TL_NSE[6,N_NSE-
        1]*LAMBDA_NSE
NOBUFF_PARAMETERS[2,2]=TIME_VAR_MED[0,2]

NOBUFF_PARAMETERS[3,0]=NOBUFF_PARAMETERS_TL_NSE[0,N_NSE-1]
NOBUFF_PARAMETERS[3,1]=NOBUFF_PARAMETERS_TL_NSE[6,N_NSE-
        1]*LAMBDA_NSE
NOBUFF_PARAMETERS[3,2]=NOBUFF_PARAMETERS_TL_NSE[9,N_NSE-
        1]/LAMBDA_NSE
```

```
        NOBUFF_PARAMETERS[4,0]=NOBUFF_PARAMETERS_TL_NSE[3,N_NSE-1]
        NOBUFF_PARAMETERS[4,1]=NOBUFF_PARAMETERS_TL_NSE[7,N_NSE-
              1]*LAMBDA_NSE
        NOBUFF_PARAMETERS[4,2]=NOBUFF_PARAMETERS_TL_NSE[10,N_NSE-
              1]/LAMBDA_NSE
        print('Elaborazione tempi medi e varianze globali terminata')
        print('Calcoli terminati')

        print('Salvataggio in corso')
        CODE_KEY_NSE=[2,F_DELIMITER,F_PREC_PRG,N_NSE,
              str(T_UPSTREAM_NSE),METER_NSE]
        PRG_OUT_PRINT_NSE=[TIME_VAR, TIME_VAR_MED, WIP_NO_BUFF,
              GLOBAL_NSE, NOBUFF_PARAMETERS]
        PRINT_TO_FILE_PRG(PRG_OUT_PRINT_NSE,CODE_KEY_NSE)

        DATA_NSE=[F_NAME_NSE, N_NSE, F_PREC_NSE, TIME_VAR, TIME_VAR_MED,
              GLOBAL_NSE, NOBUFF_PARAMETERS, T_UPSTREAM_NSE, METER_NSE]
        PRINT_TO_USER_FILE('tab1_gen',DATA_NSE)
        if not(WHO_CALLS=='sched'):
          DATA_NSE=[F_NAME_NSE,F_PREC_NSE, WIP_NO_BUFF]
          PRINT_TO_USER_FILE('tab2_gen',DATA_NSE)
        #end if
        print('Salvataggio terminato')
      #end if
      if not(WHO_CALLS=='sched'):
        SERV_NSE=INPUT_MENU('tab_repeat',0)
        if SERV_NSE[0]==1:
          MENU_CTRL_NSE=-1
          print('Limite errori di immissione dati raggiunto')
        else:
          MENU_CTRL_NSE=SERV_NSE[1]
          if MENU_CTRL_NSE==1:
            MENU_CTRL_NSE=0
          else:
            MENU_CTRL_NSE=-1
          #end if
        #end if
      else:
        MENU_CTRL_NSE=-1
      #end if
    #end while
    if not(WHO_CALLS=='sched'):
      SERV_NSE=INPUT_MENU('tab_sav',0)
      if SERV_NSE[0]==1:
        print('Limite errori di immissione dati raggiunto')
        input('I dati di elaborazione non verranno mantenuti in memoria')
      else:
        if SERV_NSE[1]==1:   #RAMO RISPOSTA POSITIVA
          SAVE_CTRL_NSE=0
        else:
```

```
            SAVE_CTRL_NSE=1
        #end if
    else:
        SAVE_CTRL_NSE=0
    #end if
    if SAVE_CTRL_NSE==0:
        OUT_NSE=[1, [N_NSE, T_UPSTREAM_NSE, METER_NSE], TIME_VAR,
          TIME_VAR_MED, WIP_NO_BUFF, GLOBAL_NSE, NOBUFF_PARAMETERS]
    else:
        OUT_NSE=[0]
    if not(WHO_CALLS=='sched'):
        input('Reindirizzamento a menù sezione elaborazioni')
    #end if
    return(OUT_NSE)
#end NEW_SET_ELABORATION()

def USER_THEORY_MAIN():
#Funzione custom adibita alla gestione del menù contenente le operazioni eseguibili in modalità
#utente della sezione di modellizzazione e al mantenimento dei loro dati comuni.
#Viene richiamata dalla funzione custom USER_MAIN.
#L'immissione della scelta avviene attraverso la funzione custom INPUT_MENU, in base ad essa
#verranno richiamate GENERAL_MATRIX_LOADING, TIMELESS,
#NEW_SET_ELABORATION e BUFFER_ELABORATION.
    STATUS_MTX_MAIN=1
    STATUS_TABS_MAIN=1
    STATUS_TABS_TIMELESS_MAIN=1
    MENU_CTRL_MAIN=0
    while MENU_CTRL_MAIN==0:
        DATA_MTX_MAIN=[]
        DATA_TABS_TIMELESS_MAIN=[]
        DATA_TABS_MAIN=[]
        DATA_MTX_MAIN.append(STATUS_MTX_MAIN)
        DATA_TABS_TIMELESS_MAIN.append(STATUS_TABS_TIMELESS_MAIN)
        DATA_TABS_MAIN.append(STATUS_TABS_MAIN)
        if STATUS_MTX_MAIN==0:
            DATA_MTX_MAIN.append(DELTA_Z_MAIN)
            DATA_MTX_MAIN.append(CINT_Z_MAIN)
            DATA_MTX_MAIN.append(CF_Z_MAIN)
            DATA_MTX_MAIN.append(CINT_TA_MAIN)
            DATA_MTX_MAIN.append(CF_TA_MAIN)
            DATA_MTX_MAIN.append(N_MTX_MAIN)
            DATA_MTX_MAIN.append(CF_TU_MAIN)
            DATA_MTX_MAIN.append(CINT_TA_GLBL_MAIN)
            DATA_MTX_MAIN.append(CDEN_TA_GLBL_MAIN)
            DATA_MTX_MAIN.append(CINT_TW_GLBL_MAIN)

            DATA_MTX_MAIN.append(CDEN_TW_GLBL_MAIN)
            DATA_MTX_MAIN.append(CDEN_TU_GLBL_MAIN)
        #end if
        if STATUS_TABS_TIMELESS_MAIN==0:
```

```
    DATA_TABS_TIMELESS_MAIN.append(N_TABS_TIMELESS_MAIN)
    DATA_TABS_TIMELESS_MAIN.append(TIME_VAR_TIMELESS_MAIN)
    DATA_TABS_TIMELESS_MAIN.append(TIME_VAR_TIMELESS_MED_MAIN)
    DATA_TABS_TIMELESS_MAIN.append(WIP_NO_BUFF_TIMELESS_MAIN)
    DATA_TABS_TIMELESS_MAIN.append(GLOBAL_TIMELESS_MAIN)
    DATA_TABS_TIMELESS_MAIN.append
            (NOBUFF_TIMELESS_PARAMETERS_MAIN)
#end if
if STATUS_TABS_MAIN==0:
    DATA_TABS_MAIN.append
            ([N_TABS_MAIN,T_UPSTREAM_MAIN,METER_MAIN])
    DATA_TABS_MAIN.append(TIME_VAR_MAIN)
    DATA_TABS_MAIN.append(TIME_VAR_MED_MAIN)
    DATA_TABS_MAIN.append(WIP_NO_BUFF_MAIN)
    DATA_TABS_MAIN.append(GLOBAL_MAIN)
    DATA_TABS_MAIN.append(NOBUFF_PARAMETERS_MAIN)
#end if

SERV_MAIN=INPUT_MENU('user_theory',0)
ERROR_CTRL_MAIN=SERV_MAIN[0]
if ERROR_CTRL_MAIN==1:
    print('Limite errori di immissione dati raggiunto')
    MENU_CTRL_MAIN=-1
elif ERROR_CTRL_MAIN==0:
    MENU_CTRL_MAIN=SERV_MAIN[1]
    if MENU_CTRL_MAIN==1:
        SERV_MAIN=GENERAL_MATRIX_LOADING('mtx',0)
        if SERV_MAIN[0]==1:
            STATUS_MTX_MAIN=0
            DELTA_Z_MAIN=SERV_MAIN[1]
            CINT_Z_MAIN=SERV_MAIN[2]
            CF_Z_MAIN=SERV_MAIN[3]
            CINT_TA_MAIN=SERV_MAIN[4]
            CF_TA_MAIN=SERV_MAIN[5]
            N_MTX_MAIN=DELTA_Z_MAIN.cols
            CF_TU_MAIN=SERV_MAIN[6]
            CINT_TA_GLBL_MAIN=SERV_MAIN[7]
            CDEN_TA_GLBL_MAIN=SERV_MAIN[8]
            CINT_TW_GLBL_MAIN=SERV_MAIN[9]
            CDEN_TW_GLBL_MAIN=SERV_MAIN[10]
            CDEN_TU_GLBL_MAIN=SERV_MAIN[11]
        else:
            MENU_CTRL_MAIN==0
        #end if

    elif MENU_CTRL_MAIN==2:
        SERV_MAIN=GENERAL_MATRIX_LOADING('timeless',0)
        if SERV_MAIN[0]==1:
            STATUS_TABS_TIMELESS_MAIN=0
            TIME_VAR_TIMELESS_MAIN=SERV_MAIN[1]
```

```
        TIME_VAR_TIMELESS_MED_MAIN=SERV_MAIN[2]
        WIP_NO_BUFF_TIMELESS_MAIN=SERV_MAIN[3]
        GLOBAL_TIMELESS_MAIN=SERV_MAIN[4]
        NOBUFF_TIMELESS_PARAMETERS_MAIN=SERV_MAIN[5]
        N_TABS_TIMELESS_MAIN=TIME_VAR_TIMELESS_MED_MAIN.cols
    else:
        MENU_CTRL_MAIN==0
    #end if

elif MENU_CTRL_MAIN==3:
    SERV_MAIN=GENERAL_MATRIX_LOADING('tabs',0)
    if SERV_MAIN[0]==1:
        STATUS_TABS_MAIN=0
        N_TABS_MAIN=int(SERV_MAIN[1][0])
        T_UPSTREAM_MAIN=mp.mpf(SERV_MAIN[1][1])
        METER_MAIN=SERV_MAIN[1][2]
        TIME_VAR_MAIN=SERV_MAIN[2]
        TIME_VAR_MED_MAIN=SERV_MAIN[3]
        WIP_NO_BUFF_MAIN=SERV_MAIN[4]
        GLOBAL_MAIN=SERV_MAIN[5]
        NOBUFF_PARAMETERS_MAIN=SERV_MAIN[6]
    else:
        MENU_CTRL_MAIN==0
    #end if

elif MENU_CTRL_MAIN==4:
    SERV_MAIN=GENERAL_MATRIX_GENERATION('User_Main',0)
    if SERV_MAIN[0]==1:
        STATUS_MTX_MAIN=0
        DELTA_Z_MAIN=SERV_MAIN[1]
        CINT_Z_MAIN=SERV_MAIN[2]
        CF_Z_MAIN=SERV_MAIN[3]
        CINT_TA_MAIN=SERV_MAIN[4]
        CF_TA_MAIN=SERV_MAIN[5]
        N_MTX_MAIN=DELTA_Z_MAIN.cols
        CF_TU_MAIN=SERV_MAIN[6]
        CINT_TA_GLBL_MAIN=SERV_MAIN[7]
        CDEN_TA_GLBL_MAIN=SERV_MAIN[8]
        CINT_TW_GLBL_MAIN=SERV_MAIN[9]
        CDEN_TW_GLBL_MAIN=SERV_MAIN[10]
        CDEN_TU_GLBL_MAIN=SERV_MAIN[11]
    #end if
    print('Generazione matrici terminata,premere INVIO\n')

elif MENU_CTRL_MAIN==5:
    SERV_MAIN=TIMELESS('timeless',DATA_MTX_MAIN)
    if SERV_MAIN[0]==1:
        STATUS_TABS_TIMELESS_MAIN=0
        N_TABS_TIMELESS_MAIN=SERV_MAIN[1]
        TIME_VAR_TIMELESS_MAIN=SERV_MAIN[2]
```

```
                TIME_VAR_TIMELESS_MED_MAIN=SERV_MAIN[3]
                WIP_NO_BUFF_TIMELESS_MAIN=SERV_MAIN[4]
                GLOBAL_TIMELESS_MAIN=SERV_MAIN[5]
                NOBUFF_TIMELESS_PARAMETERS_MAIN=SERV_MAIN[6]
            #end if
            print('Generazione matrici terminata,premere INVIO\n')

        elif MENU_CTRL_MAIN==6:
            SERV_MAIN=NEW_SET_ELABORATION
                ('nobuff',DATA_TABS_TIMELESS_MAIN)
            if SERV_MAIN[0]==1:
                STATUS_TABS_MAIN=0
                N_TABS_MAIN=int(SERV_MAIN[1][0])
                T_UPSTREAM_MAIN=mp.mpf(SERV_MAIN[1][1])
                METER_MAIN=SERV_MAIN[1][2]
                TIME_VAR_MAIN=SERV_MAIN[2]
                TIME_VAR_MED_MAIN=SERV_MAIN[3]
                WIP_NO_BUFF_MAIN=SERV_MAIN[4]
                GLOBAL_MAIN=SERV_MAIN[5]
                NOBUFF_PARAMETERS_MAIN=SERV_MAIN[6]
            #end if
        elif MENU_CTRL_MAIN==7:
            BUFFER_ELABORATION('main',DATA_TABS_MAIN)
        elif MENU_CTRL_MAIN==0:
            MENU_CTRL_MAIN=(-1)
        #end if
    #end if
    if not(MENU_CTRL_MAIN==(-1)):
        MENU_CTRL_MAIN=0
#end while
input('Reindirizzamento a menù precedente')
return()
#End def USER_THEORY_MAIN()

def MIN_ARG(A_INT,B_INT):
#Funzione custom che restituisce in output il minore tra i due valori che gli vengono forniti in input.
    if B_INT<A_INT:
        OUT=B_INT
    else:
        OUT=A_INT
    #end if
    return(OUT)
#end MIN_ARG

def DESCRIPTION_LINE(ROW_INT):
#Funzione custom che restituisce in output delle stringhe di testo che saranno utilizzate dalle
#funzioni custom chiamanti nella stampa dei loro files.
    i_desc=0
    DESC_INT=[]
    while i_desc<ROW_INT:
```

```
        DESC_INT.append('Linea %d'%(i_desc+1))
        i_desc+=1
    #end while
    return(DESC_INT)
#end def DESCRIPTION_LINE



def CASUAL_EXTRACTION(WHO_CALLS,T_INT,INT_LAMBDA,
LIM_INT,FILE_NAME_COD_CE,F_PREC_CE):
#Funzione custom deputata all'estrazione casuale dei tempi.
#Viene richiamata esclusivamente dalla funzione custom SINGLE_SIM sia in modalità utente che
#in modalita schedulazione. In modalità utente verrà data la possibilità di produrre un file di output
#tramite le funzioni custom INPUT_MENU e PRINT_TO_SINGLE_ELABORATION_FILE.
#Restituisce i dati prodotti alla funzione chiamante.
    COL=T_INT.cols
    ROW=T_INT.rows
    if WHO_CALLS=='single_nb' or WHO_CALLS=='single_buff':
        SERV_CE=INPUT_MENU('save_sim','estrazioni casuali')
        if SERV_CE[0]==1:
            print('I risultati non verranno esportati su file')
            OUT_FILE_CTRL=0
        else:
            if SERV_CE[1]==1:
                OUT_FILE_CTRL=1
            else:
                OUT_FILE_CTRL=0
            #end if
        #end if
    elif WHO_CALLS=='sched':
        OUT_FILE_CTRL=1
    else:
        OUT_FILE_CTRL=0
    #end if

    i_int=0
    while i_int<ROW:
        j_int=0
        while j_int<COL:
            EXT_CONT=0
            while EXT_CONT==0:
                SERV_INT=random.randint(1,LIM_INT)/LIM_INT
                T_INT[i_int,j_int]=-(1/INT_LAMBDA)*mp.log(1-SERV_INT)
                if T_INT[i_int,j_int]<10*(1/INT_LAMBDA):
                    EXT_CONT=1
                #end IF
            #end WHILE
            j_int+=1
        #end while
        i_int+=1
```

```
    #end while
    if OUT_FILE_CTRL==1:
        F_NAME_INT='%s_estrazioni casuali'%(FILE_NAME_COD_CE)
        F_EXT_INT=F_EXT
        TABLE_TEXT='Linea \ Componente'
        DESCRIPTION=DESCRIPTION_LINE(ROW)
        PRINT_TO_SINGLE_ELABORATION_FILE(F_NAME_INT,F_EXT_INT,
          F_PREC_CE,F_DELIMITER,TABLE_TEXT,DESCRIPTION,T_INT)
    return(T_INT)
#end CASUAL_EXTRACTION

def CALC_EVERYTHING_SNB(WHO_CALLS,EXT_INT,TLA_INT,TDA_INT,
F_NAME_COD_CE,F_PREC_CE):
#Funzione custom deputata al calcolo dei dati di una simulazione singola in assenza di buffer.
#Viene richiamata esclusivamente dalla funzione custom SINGLE_SIM.
#Quando in modalità utente viene richiesta una simulazione singola, vi è la possibilità tramite le
#funzioni custom INPUT_MENU, DESCRIPTION_LINE e
#PRINT_TO_SINGLE_ELABORATION_FILE di produrre dei files di output contenenti i dati
#prodotti.
#Restituisce in output alla funzione chiamante quanto elaborato.
    COL=int(EXT_INT.cols)
    ROW=int(EXT_INT.rows)
    if WHO_CALLS=='single_nb':
        SERV_CE=INPUT_MENU('save_sim','timeline ingressi dalle linee')
        if SERV_CE[0]==1:
            print('I risultati non verranno esportati su file')
            OUT1_FILE_CTRL=0
        else:
            if SERV_CE[1]==1:
                OUT1_FILE_CTRL=1
            else:
                OUT1_FILE_CTRL=0
            #end if
        #end if
        SERV_CE=INPUT_MENU('save_sim','intertempi di ingresso, tempi di attesa, timeline con
          ingressi e uscite, livelli di WIP')
        if SERV_CE[0]==1:
            print('I risultati non verranno esportati su file')
            OUT2_FILE_CTRL=0
        else:
            if SERV_CE[1]==1:
                OUT2_FILE_CTRL=1
            else:
                OUT2_FILE_CTRL=0
            #end if
        #end if
    elif WHO_CALLS=='sched':

        OUT1_FILE_CTRL=1
        OUT2_FILE_CTRL=1
```

```
else:
   OUT1_FILE_CTRL=0
   OUT2_FILE_CTRL=0
#end if
for i_int in range(ROW):
    TLA_INT[i_int,0]=EXT_INT[i_int,0]
#end for i_int
MIN_INT=TLA_INT[0,0]
MAX_INT=TLA_INT[0,0]
for i_int in range(ROW):
   if TLA_INT[i_int,0]<MIN_INT:
      MIN_INT=TLA_INT[i_int,0]
   if TLA_INT[i_int,0]>MAX_INT:
      MAX_INT=TLA_INT[i_int,0]
   #end if
#end for i_int
TDA_INT[0,0]=MIN_INT
TDA_INT[1,0]=MAX_INT

j_int=1
while j_int<COL:
   MIN_INT=TLA_INT[0,j_int-1]+EXT_INT[0,j_int]
   MAX_INT=TLA_INT[0,j_int-1]+EXT_INT[0,j_int]
   for i_int in range(ROW):
      TLA_INT[i_int,j_int]=TLA_INT[i_int,(j_int-1)]+EXT_INT[i_int,j_int]
   #end for i_int
   for i_int in range(ROW):
      if TLA_INT[i_int,j_int]<MIN_INT:
         MIN_INT=TLA_INT[i_int,j_int]
      #end if
      if TLA_INT[i_int,j_int]>MAX_INT:
         MAX_INT=TLA_INT[i_int,j_int]
      #end if
   #end for i_int
   TDA_INT[0,j_int]=MIN_INT
   TDA_INT[1,j_int]=MAX_INT
   j_int+=1
#end while

TDA_INT[2,0]=TDA_INT[0,0]
TDA_INT[3,0]=TDA_INT[1,0]-TDA_INT[0,0]
TDA_INT[4,0]=TDA_INT[1,0]-TDA_INT[0,0]
j_int=1
while j_int<COL:
   TDA_INT[2,j_int]=TDA_INT[0,j_int]-TDA_INT[0,j_int-1]
   TDA_INT[3,j_int]=TDA_INT[1,j_int]-TDA_INT[0,j_int]
   TDA_INT[4,j_int]=TDA_INT[1,j_int]-TDA_INT[1,j_int-1]
   j_int+=1
#end while
```

```
    if OUT1_FILE_CTRL==1:
        F_NAME_INT='%s_timeline ingressi dalle linee'%(F_NAME_COD_CE)
        F_EXT_INT=F_EXT
        TABLE_TEXT='Linea \ Componente'
        DESCRIPTION=DESCRIPTION_LINE(ROW)
        PRINT_TO_SINGLE_ELABORATION_FILE(F_NAME_INT,F_EXT_INT,
            F_PREC_CE,F_DELIMITER,TABLE_TEXT,DESCRIPTION,TLA_INT)
    #end if
    if OUT2_FILE_CTRL==1:
        F_NAME_INT='%s_timeline stazione accoppiamento, intertempi ingresso, tempi di attesa,
            tempi di uscita'%(F_NAME_COD_CE)
        F_EXT_INT=F_EXT
        TABLE_TEXT='Descrizione \ Pacchetto componenti'
        DESCRIPTION=['Tempo ingresso primo componente pacchetto','Tempo uscita pacchetto
            completo','Intertempi ingresso','Tempi di attesa','Intertempi di uscita']
        PRINT_TO_SINGLE_ELABORATION_FILE(F_NAME_INT,F_EXT_INT,F_PREC_CE,
            F_DELIMITER,TABLE_TEXT,DESCRIPTION,TDA_INT)
    #end if
    OUT=[]
    OUT.append(TLA_INT)
    OUT.append(TDA_INT)
    return(OUT)
#end def CALC_EVERYTHING_SNB

def CALC_EVERYTHING_SBUFF(WHO_CALLS,EXT_INT,BUFF_INT,
F_NAME_CE,F_PREC_CE):
#Funzione custom deputata al calcolo dei dati di una simulazione singola in presenza di buffer.
#Viene richiamata esclusivamente dalla funzione custom SINGLE_SIM.
#In esecuzione verrà richiamata anche la funzione custom  MIN_ARG.
#Quando in modalità utente viene richiesta una simulazione singola, vi è la possibilità tramite le
#funzioni custom INPUT_MENU, DESCRIPTION_LINE e
#PRINT_TO_SINGLE_ELABORATION_FILE di produrre dei files di output contenenti i dati
#prodotti.
#Restituisce in output alla funzione chiamante quanto elaborato.
    if WHO_CALLS=='single_buff':
        SERV_CE=INPUT_MENU('save_sim','timeline ingressi dalle linee')
        if SERV_CE[0]==1:
            print('I risultati non verranno esportati su file')
            OUT1_FILE_CTRL=0
        else:
            if SERV_CE[1]==1:
                OUT1_FILE_CTRL=1
            else:
                OUT1_FILE_CTRL=0
            #end if
        #end if
        SERV_CE=INPUT_MENU('save_sim','intertempi di ingresso, tempi di attesa, timeline con
            ingressi e uscite, livelli di WIP')
        if SERV_CE[0]==1:
            print('I risultati non verranno esportati su file')
```

```
      OUT2_FILE_CTRL=0
   else:
      if SERV_CE[1]==1:
         OUT2_FILE_CTRL=1
      else:
         OUT2_FILE_CTRL=0
      #end if
   #end if
elif WHO_CALLS=='sched':
   OUT1_FILE_CTRL=1
   OUT2_FILE_CTRL=1
else:
   OUT1_FILE_CTRL=0
   OUT2_FILE_CTRL=0
#end if
PASSO=2*BUFF_INT
COL=int(EXT_INT.cols)
ROW=int(EXT_INT.rows)
TLA_INT=mp.zeros(ROW,COL)
TDA_INT=mp.zeros(5,COL)
START_INDEX_INT=0
REWORK_INT=3
z_wip=0
while not(REWORK_INT==0):
   END_J=MIN_ARG((START_INDEX_INT+PASSO-1),(COL-1))

   if REWORK_INT==2:
      for i_int in range(ROW):
         TLA_INT[i_int,START_INDEX_INT]=TLA_INT[i_int,START_INDEX_INT-
               1]+EXT_INT[i_int,START_INDEX_INT]
      #end for i_int
      j_wip=START_INDEX_INT-1
   elif REWORK_INT==1:
      for i_int in range(ROW):
         TLA_INT[i_int,START_INDEX_INT]=TDA_INT[1,START_INDEX_INT-
               1]+EXT_INT[i_int,START_INDEX_INT]
      #end for i_int
      B_LEVEL_INT=1
      z_wip=START_INDEX_INT
      j_wip=START_INDEX_INT
   else:
      for i_int in range(ROW):
         TLA_INT[i_int,START_INDEX_INT]=EXT_INT[i_int,START_INDEX_INT]
      #end for i_int
      B_LEVEL_INT=1
      z_wip=0
      j_wip=0
   #end if
   MIN_INT=TLA_INT[0,START_INDEX_INT]
   MAX_INT=TLA_INT[0,START_INDEX_INT]
```

```
for i_int in range(ROW):
   if TLA_INT[i_int,START_INDEX_INT]<MIN_INT:
      MIN_INT=TLA_INT[i_int,START_INDEX_INT]
      #end if
   if TLA_INT[i_int,START_INDEX_INT]>MAX_INT:
      MAX_INT=TLA_INT[i_int,START_INDEX_INT]
   #end if
#end for i_int
TDA_INT[0,START_INDEX_INT]=MIN_INT
TDA_INT[1,START_INDEX_INT]=MAX_INT

j_int=START_INDEX_INT+1
while j_int<=END_J:
   MIN_INT=TLA_INT[0,j_int-1]+EXT_INT[0,j_int]
   MAX_INT=TLA_INT[0,j_int-1]+EXT_INT[0,j_int]
   for i_int in range(ROW):
      TLA_INT[i_int,j_int]=TLA_INT[i_int,(j_int-1)]+EXT_INT[i_int,j_int]
   #end for i_int
   for i_int in range(ROW):
      if TLA_INT[i_int,j_int]<MIN_INT:
         MIN_INT=TLA_INT[i_int,j_int]
         #end if
      if TLA_INT[i_int,j_int]>MAX_INT:
         MAX_INT=TLA_INT[i_int,j_int]
      #end if
   #end for i_int
   TDA_INT[0,j_int]=MIN_INT
   TDA_INT[1,j_int]=MAX_INT
   j_int+=1
#end while

WHILE_STATUS=0
while WHILE_STATUS==0:
   WHILE_CONTROL=0
   if not(j_wip==COL-1 and z_wip==j_wip):
      while z_wip<(j_wip+1) and WHILE_CONTROL==0:
         if TDA_INT[1,z_wip]<TDA_INT[0,j_wip+1]:
            B_LEVEL_INT+=(-1)
            if B_LEVEL_INT<0:
               input('ERRORE BUFFER SOTTO ZERO')
             #end if
            z_wip+=1
         else:
            WHILE_CONTROL=1
         #end if
      #end while
      if B_LEVEL_INT+1>BUFF_INT and j_wip+2<COL:
         WHILE_STATUS=1
         START_INDEX_INT=j_wip+2  #MAX di j_wip+2 =COL
         REWORK_INT=1
```

```
            elif B_LEVEL_INT+1>BUFF_INT and not(j_wip+2<COL):
               WHILE_STATUS=1
               REWORK_INT=0
            elif B_LEVEL_INT+1<=BUFF_INT and j_wip+1<END_J:
               WHILE_STATUS=0
               REWORK_INT=2
               j_wip+=1
               B_LEVEL_INT+=1
            elif B_LEVEL_INT+1<=BUFF_INT and j_wip+1==END_J and j_wip+2<COL:
               WHILE_STATUS=1
               REWORK_INT=2
               START_INDEX_INT=j_wip+2
               B_LEVEL_INT+=1
            elif B_LEVEL_INT+1<=BUFF_INT and j_wip+1==END_J and not(j_wip+2<COL):
               WHILE_STATUS=1
               REWORK_INT=0
            #end if
         else:
            WHILE_STATUS=1
            REWORK_INT=0
         #end if
   #end while
   TDA_INT[2,0]=TDA_INT[0,0]
   TDA_INT[3,0]=TDA_INT[1,0]-TDA_INT[0,0]
   TDA_INT[4,0]=TDA_INT[1,0]-TDA_INT[0,0]

   j_int=1
   while j_int<COL:
      TDA_INT[2,j_int]=TDA_INT[0,j_int]-TDA_INT[0,j_int-1]
      TDA_INT[3,j_int]=TDA_INT[1,j_int]-TDA_INT[0,j_int]
      TDA_INT[4,j_int]=TDA_INT[1,j_int]-TDA_INT[1,j_int-1]
      j_int+=1
   #end while

   if OUT1_FILE_CTRL==1:
      F_NAME_INT='%s_timeline ingressi dalle linee'%(F_NAME_CE)
      F_EXT_INT=F_EXT
      TABLE_TEXT='Linea \ Componente'
      DESCRIPTION=DESCRIPTION_LINE(ROW)
      PRINT_TO_SINGLE_ELABORATION_FILE(F_NAME_INT,F_EXT_INT,F_PREC_CE,
         F_DELIMITER,TABLE_TEXT,DESCRIPTION,TLA_INT)
   #end if
   if OUT2_FILE_CTRL==1:
      F_NAME_INT='%s_intertempi ingresso, tempi di attesa, intertempi di uscita, timeline
         stazione accoppiamento'%(F_NAME_CE)
      F_EXT_INT=F_EXT
      TABLE_TEXT='Descrizione \ Pacchetto componenti'
      DESCRIPTION=['Tempo ingresso primo componente pacchetto','Tempo uscita pacchetto
         completo','Intertempi ingresso','Tempi di attesa','Intertempi di uscita']
      PRINT_TO_SINGLE_ELABORATION_FILE(F_NAME_INT,F_EXT_INT,F_PREC_CE,
```

```
            F_DELIMITER,TABLE_TEXT,DESCRIPTION,TDA_INT)
    #end if
    OUT=[]
    OUT.append(TLA_INT)
    OUT.append(TDA_INT)
    return(OUT)
#end def CALC_EVERYTHING_SBUFF

def TIMELINE_TO_WIP(WHO_CALLS, F_NAME_TW, F_PREC_TW, TDA_INT):
#Funzione custom dedicata alla creazione della timeline degli eventi alla stazione di accoppiamento
#e della modifica che questi apportano ai livelli di WIP.
#Viene richiamata esclusivamente dalla funzione custom SINGLE_SIM.
#Quando in modalità utente viene richiesta una simulazione singola, vi è la possibilità tramite le
#funzioni custom INPUT_MENU e PRINT_TO_SINGLE_ELABORATION_FILE di produrre dei
#files di output contenenti i dati prodotti.
#Restituisce in output alla funzione chiamante quanto elaborato.
    if WHO_CALLS=='single_buff' or WHO_CALLS=='single_nb':
        SERV_CE=INPUT_MENU('save_sim','timeline evoluzione wip')
        if SERV_CE[0]==1:
            print('I risultati non verranno esportati su file')
            OUT1_FILE_CTRL=0
        else:
            if SERV_CE[1]==1:
                OUT1_FILE_CTRL=1
            else:
                OUT1_FILE_CTRL=0
            #end if
        #end if
    else:
        OUT1_FILE_CTRL=0
    #end if

    COL=TDA_INT.cols
    TW_INT=mp.zeros(3,2*COL)
    SERV_WIP=[]
    SERV_WIP.append(0)
    SERV_WIP.append(0)

    for j_int in range(COL):
        TW_INT[0,j_int]=TDA_INT[0,j_int]
        TW_INT[1,j_int]=1
        TW_INT[0,COL+j_int]=TDA_INT[1,j_int]
        TW_INT[1,COL+j_int]=(-1)
    for j_int in range(2*COL):
        z_int=j_int+1
        while z_int<2*COL:
            if TW_INT[0,j_int]>TW_INT[0,z_int]:
                SERV_WIP[0]=TW_INT[0,z_int]
                SERV_WIP[1]=TW_INT[1,z_int]
                TW_INT[0,z_int]=TW_INT[0,j_int]
```

```
            TW_INT[1,z_int]=TW_INT[1,j_int]
            TW_INT[0,j_int]=SERV_WIP[0]
            TW_INT[1,j_int]=SERV_WIP[1]
        #end if
        z_int+=1
    #end while
  TW_INT[2,0]=1
  for j_int in range(2*COL-1):
    TW_INT[2,j_int+1]=TW_INT[2,j_int]+TW_INT[1,j_int+1]
  #end for j_int
  if OUT1_FILE_CTRL==1:
    F_NAME_INT='%s_timeline_wip'%(F_NAME_TW)
    F_EXT_INT=F_EXT
    TABLE_TEXT='Descrizione \ Componente'
    DESCRIPTION=['Timeline eventi','Modificatore WIP','Livello di WIP totale']
    PRINT_TO_SINGLE_ELABORATION_FILE(F_NAME_INT,F_EXT_INT,
      F_PREC_TW,F_DELIMITER,TABLE_TEXT,DESCRIPTION,TW_INT)
  #end if
  return(TW_INT)
#end def TIMELINE_TO_WIP

def N_STOP_CALC(TW_INT,BUFF_INT,N_INT):
#Funzione custom che crea un array contente ogni numero di pacchetti entranti alla stazione di
#accoppiamento tra due arresti di produzione dovuti a saturazione del buffer.
#Può essere richiamata solo dalla funzione custom SINGLE_SIM, alla quale restituirà in output i
#dati prodotti.
  COL=TW_INT.cols
  N_TO_STOP_INT=mp.zeros(1,int((N_INT)/BUFF_INT)+1)
  n_stop_count=0
  n_stop_index=0
  j_int=0
  while j_int<COL:
    if TW_INT[1,j_int]>0:
      n_stop_count+=1
      if TW_INT[2,j_int]==BUFF_INT+1:
        N_TO_STOP_INT[0,n_stop_index]=n_stop_count
        n_stop_index+=1
        n_stop_count=0
      #end if
    #end if
    j_int+=1
  #end while
  return(N_TO_STOP_INT)
#end def N_TO_STOP_INT

def VAR_CALC(MED_INT,MTX_INT,line):
#Funzione custom dedicata al calcolo della varianza dei dati che le vengono forniti in input.
#Viene richiamata esclusivamente dalla funzione custom SINGLE_SIM, alla quale restituirà in
#output i dati prodotti.
  COL=MTX_INT.cols
```

```
    SERV1_INT=0
    for j in range(COL):
        SERV2_INT=MTX_INT[line,j]-MED_INT
        SERV1_INT+=mp.power(SERV2_INT,2)
    #end for j
    SERV1_INT=SERV1_INT/COL
    if SERV1_INT<0:
        input('ERRORE VARIANZA')
    #end if
    return(SERV1_INT)
#end def VAR_CALC

def SINGLE_SIM(WHO_CALLS,DATA_SS):
#Funzione custom designata al calcolo dei parametri della simulazione singola di casi specifici con
#o senza buffer.
#Può essere chiamata direttamente in modalità utente tramite la funzione custom
#USER_SIMULATION_MAIN.
#Può anche essere richiamata da MULTI_SIM e dallo schedulatore.
#In modalità utente i dati iniziali verranno richiesti tramite la funzione custom INPUT_MENU.
#Durante l'esecuzione dei calcoli verranno richiamate le funzioni custom
#CASUAL_EXTRACTION per la generazione di tempi casuali, CALC_EVERYTHING_SNB per i
#calcoli in assenza di buffer, CALC_EVERYTHING_SBUFF in presenza di buffer,
#TIMELINE_TO_WIP, N_STOP_CALC e VAR_CALC.
#In caso di chiamata diretta specifica di questa funzione dalla modalità utente o schedulatore,
#verranno prodotti dei files di output tramite la funzione custom
#PRINT_TO_SINGLE_FINAL_FILE.
#Restituisce dei dati alla funzione chiamante solo se richiamata tramite MULTI_SIM.
    CTRL_SS=0
    while CTRL_SS==0:
        if WHO_CALLS=='single_nb':
            TEXT_CALL_SS='single_nb'
        elif WHO_CALLS=='single_buff':
            TEXT_CALL_SS='single_buff'
        #end if
        if not(WHO_CALLS=='multi_nb' or WHO_CALLS=='multi_buff' or
WHO_CALLS=='sched'):
            SERV_SS=INPUT_MENU('sim_data',0)
            if SERV_SS[0]==1:
                print('Raggiunto limite di errori in immissione dati')
                CTRL_SS=-1
            else:
                CTRL_SS=0
                N_LINES_SS=SERV_SS[1]
                T_UPSTREAM_SS=SERV_SS[2]
                METER_SS=SERV_SS[3]
                N_SS=SERV_SS[4]
                F_PREC_SS=SERV_SS[5]
            #end if
            if WHO_CALLS=='single_buff' and CTRL_SS==0:
                SERV_SS=INPUT_MENU('sim_data_b',[N_SS])
```

```
        if SERV_SS[0]==1:
            print('Raggiunto limite di errori in immissione dati')
            CTRL_SS=-1
        else:
            CTRL_SS=0
            BUFFER_SS=SERV_SS[1]
        #end if
    #end if
else:
    TEXT_CALL_SS=WHO_CALLS
    CTRL_SS=0
    N_LINES_SS=DATA_SS[0]
    T_UPSTREAM_SS=mp.mpf(DATA_SS[1])
    METER_SS=DATA_SS[2]
    N_SS=DATA_SS[3]
    F_PREC_SS=DATA_SS[4]
    if len(DATA_SS)==6:
        BUFFER_SS=DATA_SS[5]
    #end if
#end if
if CTRL_SS==0:
    if METER_SS=='s' or METER_SS=='S' or METER_SS=='sec':
        METER_SS='sec'
    elif METER_SS=='m' or METER_SS=='M' or METER_SS=='min':
        METER_SS='min'
    else:
        METER_SS='h'
    #end if
    LAMBDA_SS=1/T_UPSTREAM_SS
    if len(DATA_SS)==5 or WHO_CALLS=='single_nb':
        FILE_NAME_COD_SS='NBS_%du_%dLNS_%s%sTM'%(N_SS, N_LINES_SS ,
            mp.nstr(T_UPSTREAM_SS,F_PREC_SS), METER_SS)
    elif len(DATA_SS)==6 or WHO_CALLS=='single_buff':
        FILE_NAME_COD_SS='BS_%du_%dLNS_%s%sTM_%dBUFF'%(N_SS,
            N_LINES_SS, mp.nstr(T_UPSTREAM_SS,F_PREC_SS), METER_SS,
            BUFFER_SS)
    #end if

    T_EXTRACT=mp.zeros(N_LINES_SS, N_SS)
    T_EXTRACT=CASUAL_EXTRACTION(TEXT_CALL_SS, T_EXTRACT,
            LAMBDA_SS, EXTRACTION_RANGE, FILE_NAME_COD_SS, F_PREC_SS)
    if len(DATA_SS)==5 or WHO_CALLS=='single_nb':
        TIMELINE_ARRIVALS=mp.zeros(N_LINES_SS,N_SS)
        TIME_ARR_WAIT_WIP=mp.zeros(5,N_SS)
        SERV_OUT=CALC_EVERYTHING_SNB(TEXT_CALL_SS, T_EXTRACT,
            TIMELINE_ARRIVALS, TIME_ARR_WAIT_WIP, FILE_NAME_COD_SS,
            F_PREC_SS)
    elif len(DATA_SS)==6 or WHO_CALLS=='single_buff':
        SERV_OUT=CALC_EVERYTHING_SBUFF(TEXT_CALL_SS, T_EXTRACT,
            BUFFER_SS, FILE_NAME_COD_SS, F_PREC_SS)
```

```
#end if
TIME_ARRIVALS=SERV_OUT[0]
TIME_ARR_WAIT_WIP=SERV_OUT[1]

TIMELINE_IO_WIP=TIMELINE_TO_WIP(TEXT_CALL_SS, FILE_NAME_COD_SS,
        F_PREC_SS, TIME_ARR_WAIT_WIP)
if len(DATA_SS)==6 or WHO_CALLS=='single_buff':
   N_STOP=N_STOP_CALC(TIMELINE_IO_WIP,BUFFER_SS,N_SS)
#end if
SERV1=0
SERV2=0
SERV3=0
for i in range(N_SS):
   SERV1+=TIME_ARR_WAIT_WIP[2,i]
   SERV2+=TIME_ARR_WAIT_WIP[3,i]
   SERV3+=TIME_ARR_WAIT_WIP[4,i]
#end for i
TA_MED=SERV1/(N_SS)
TW_MED=SERV2/(N_SS)
TEXIT_MED=SERV3/(N_SS)

SERV1=0
for j in range(2*(N_SS)-1):
   SERV1=SERV1+TIMELINE_IO_WIP[2,j]*(TIMELINE_IO_WIP[0,j+1]-
        TIMELINE_IO_WIP[0,j])
#end for j
SERV2=TIMELINE_IO_WIP[0,2*(N_SS)-1]-TIMELINE_IO_WIP[0,0]
WIP_MED=SERV1/SERV2

TA_VAR=VAR_CALC(TA_MED,TIME_ARR_WAIT_WIP,2)
TW_VAR=VAR_CALC(TW_MED,TIME_ARR_WAIT_WIP,3)
TEXIT_VAR=VAR_CALC(TEXIT_MED,TIME_ARR_WAIT_WIP,4)
WIP_VAR=VAR_CALC(WIP_MED,TIMELINE_IO_WIP,2)

if len(DATA_SS)==6 or WHO_CALLS=='single_buff':
   SERV1=0
   SERV2=0
   for j in range(N_STOP.cols):
      if N_STOP[0,j]>0:
         SERV1+=N_STOP[0,j]
         SERV2+=1
      #end if
   #end for j
   if SERV2==0:
      N_STOP_MED=N_SS
   else:
      N_STOP_MED=1+int(SERV1/SERV2)
   #end if
#end if
```

```
        OUT=[]
        OUT2=[]
        OUT.append(TA_MED)
        OUT.append(TW_MED)
        OUT.append(TEXIT_MED)
        OUT.append(WIP_MED)
        OUT.append(TA_VAR)
        OUT.append(TW_VAR)
        OUT.append(TEXIT_VAR)
        OUT.append(WIP_VAR)
        OUT2.append(N_SS)
        OUT2.append(N_LINES_SS)
        OUT2.append(T_UPSTREAM_SS)
        OUT2.append(METER_SS)
        OUT2.append(F_PREC_SS)
        if len(DATA_SS)==6 or WHO_CALLS=='single_buff':
            OUT.append(N_STOP_MED)
            OUT2.append(BUFFER_SS)
        #end if
        if not(WHO_CALLS=='multi_nb' or WHO_CALLS=='multi_buff') and (len(DATA_SS)==5
                or WHO_CALLS=='single_nb'):
            PRINT_TO_SINGLE_FINAL_FILE('single_nb','%s_risultati'%
                (FILE_NAME_COD_SS), F_EXT, OUT, OUT2, F_DELIMITER)
        elif not(WHO_CALLS=='multi_nb' or WHO_CALLS=='multi_buff') and
                (len(DATA_SS)==6 or WHO_CALLS=='single_buff'):
            PRINT_TO_SINGLE_FINAL_FILE('multi_buff','%s_risultati finali'%
                (FILE_NAME_COD_SS),F_EXT, OUT, OUT2, F_DELIMITER)
        #end if

        if WHO_CALLS=='multi_nb' or WHO_CALLS=='multi_buff' or WHO_CALLS=='sched':
            CTRL_SS=-1
        if not(CTRL_SS==-1):
            print('Operazione conclusa')
            SERV_SS=INPUT_MENU('er_sub',0)
            if SERV_SS[0]==1:
                print('Raggiunto limite di errori in immissione')
                CTRL_SS=-1
            else:
                if SERV_SS[1]==1:
                    CTRL_SS=0
                else:
                    CTRL_SS=-1
                #end if
            #end if
        #end if
    #end if
#end while
if not(WHO_CALLS=='multi_nb' or WHO_CALLS=='multi_buff' or WHO_CALLS=='sched'):
    input('Reindirizzamento a menù precedente')
    OUT_SS=0
```

```
    else:
        OUT_SS=OUT
    #end if
    return(OUT_SS)
#end def SINGLE_SIM()

def MULTI_SIM(WHO_CALLS,DATA_MS):
#Funzione custom che sovraintende i blocchi di simulazioni con o senza buffer.
#Può essere richiamata da COMP_BFV, COMP_TABS e USER_SIMULATION_MAIN in
#modalità utente, e dallo schedulatore.
#In modalità utente una serie di IF in cascata richiederà i dati iniziali tramite la funzione custom
#INPUT_MENU.
#All'interno di un ciclo FOR verrà eseguito il numero di simulazioni singole richieste richiamando
#la funzione custom SINGLE_SIM, dandole in input i dati necessari al suo funzionamento e il
#comando di esecuzione in modalità "buffer" o "no-buffer".
#Al termine dei calcoli verranno prodotti dei files di output tramite la funzione
#PRINT_TO_MULTI_FINAL_FILE.
#Nel caso in cui sia stata chiamata da COMP_BFV o da COMP_TABS restituirà alla funzione
#chiamante i dati prodotti.
    OUT_MS=[0,0]
    CTRL_MS=0
    while CTRL_MS==0:
        if not(WHO_CALLS=='sched' or WHO_CALLS=='comp'):
            SERV_MS=INPUT_MENU('sim_data_m',0)
            if SERV_MS[0]==1:
                print('Raggiunto limite di errori in immissione')
                CTRL_MS=-1
            else:
                N_SIM_MS=SERV_MS[1]
                SERV_MS=INPUT_MENU('sim_data',0)
                if SERV_MS[0]==1:
                    print('Raggiunto limite di errori in immissione')
                    CTRL_MS=-1
                else:
                    CTRL_MS=0
                    N_LINES_MS=SERV_MS[1]
                    T_UPSTREAM_MS=SERV_MS[2]
                    METER_MS=SERV_MS[3]
                    N_MS=SERV_MS[4]
                    F_PREC_MS=SERV_MS[5]
                    if WHO_CALLS=='multi_buff':
                        SERV_MS=INPUT_MENU('sim_data_b',[N_MS])
                        if SERV_MS[0]==1:
                            print('Raggiunto limite di errori in immissione')
                            CTRL_MS=-1
                        else:
                            BUFFER_MS=SERV_MS[1]
                        #end if
                    #end if
                #end if
```

```
    #end if
  else:
    CTRL_MS=0
    N_LINES_MS=DATA_MS[0]
    T_UPSTREAM_MS=mp.mpf(DATA_MS[1])
    METER_MS=DATA_MS[2]
    N_MS=DATA_MS[3]
    F_PREC_MS=DATA_MS[4]
    N_SIM_MS=DATA_MS[5]
    if len(DATA_MS)==7:
      BUFFER_MS=DATA_MS[6]
    #end if
  #end if
  if CTRL_MS==0:
    if METER_MS=='s' or METER_MS=='S' or METER_MS=='sec':
      METER_MS='sec'
    elif METER_MS=='m' or METER_MS=='M' or METER_MS=='min':
      METER_MS='min'
    else:
      METER_MS='h'
    #end if
    LAMBDA_MS=1/T_UPSTREAM_MS
    if WHO_CALLS=='multi_nb' or len(DATA_MS)==6:
      FILE_NAME_COD_MS='NBM_%du_%dLNS_%s%sTM_%dSIM'%(N_MS,
          N_LINES_MS, mp.nstr(T_UPSTREAM_MS,F_PREC_MS), METER_MS,
          N_SIM_MS)
      FINAL_ARRAY=mp.zeros(8,N_SIM_MS)
    elif WHO_CALLS=='multi_buff' or len(DATA_MS)==7:
      FILE_NAME_COD_MS='BM_%du_%dLNS_%s%sTM_%dBUFF_%dSIM'%(N_MS,
          N_LINES_MS, mp.nstr(T_UPSTREAM_MS,F_PREC_MS), METER_MS,
          BUFFER_MS, N_SIM_MS)
      FINAL_ARRAY=mp.zeros(9,N_SIM_MS)
    #end if

    count_to_print=0
    count_to_print_2=0
    for COUNT_MAIN in range(N_SIM_MS):
      count_to_print+=1
      if count_to_print==10:
        count_to_print_2+=1
        print('Please wait, %d simulation executed'%(count_to_print_2*10))
        count_to_print=0
      #end if
      DATA_SERV=[]
      DATA_SERV.append(N_LINES_MS)
      DATA_SERV.append(T_UPSTREAM_MS)
      DATA_SERV.append(METER_MS)
      DATA_SERV.append(N_MS)
      DATA_SERV.append(F_PREC_MS)
      if WHO_CALLS=='multi_nb' or len(DATA_MS)==6:
```

```
      SERV_MS=SINGLE_SIM('multi_nb',DATA_SERV)
   elif WHO_CALLS=='multi_buff' or len(DATA_MS)==7:
      DATA_SERV.append(BUFFER_MS)
      SERV_MS=SINGLE_SIM('multi_buff', DATA_SERV)
   #end if

   FINAL_ARRAY[0,COUNT_MAIN]=SERV_MS[0]
   FINAL_ARRAY[2,COUNT_MAIN]=SERV_MS[1]
   FINAL_ARRAY[4,COUNT_MAIN]=SERV_MS[2]
   FINAL_ARRAY[6,COUNT_MAIN]=SERV_MS[3]
   FINAL_ARRAY[1,COUNT_MAIN]=SERV_MS[4]
   FINAL_ARRAY[3,COUNT_MAIN]=SERV_MS[5]
   FINAL_ARRAY[5,COUNT_MAIN]=SERV_MS[6]
   FINAL_ARRAY[7,COUNT_MAIN]=SERV_MS[7]
   if WHO_CALLS=='multi_buff' or len(DATA_MS)==7:
      FINAL_ARRAY[8,COUNT_MAIN]=SERV_MS[8]
   #end if
#end for COUNT_MAIN
FINAL=[]

for i in range(FINAL_ARRAY.rows):
   SERV1=0
   for j in range(N_SIM_MS):
      SERV1+=FINAL_ARRAY[i,j]
   #end for j
   SERV1=SERV1/N_SIM_MS
   FINAL.append(SERV1)
#end for i
OUT_MS=[1,FINAL]

F_EXT_MS=F_EXT
TABLE_TEXT='Descrizione \ Simulazione n.'
DATA_SERV=[]
DATA_SERV.append(N_LINES_MS)
DATA_SERV.append(T_UPSTREAM_MS)
DATA_SERV.append(METER_MS)
DATA_SERV.append(N_MS)
DATA_SERV.append(F_PREC_MS)
DATA_SERV.append(N_SIM_MS)
if WHO_CALLS=='multi_buff' or len(DATA_MS)==7:
   DATA_SERV.append(BUFFER_MS)
   F_NAME='%s_risultati finali su %d simulazioni, %d linee, %s %s tempo medio, lotto
        %d unità, %d buffer'%(FILE_NAME_COD_MS, N_SIM_MS, N_LINES_MS,
        mp.nstr(T_UPSTREAM_MS,4), METER_MS, N_MS, BUFFER_MS)
   DESCRIPTION=['Tempo medio di interarrivo','Varianza tempi medi interarrivo','Tempo
        medio di attesa','Varianza tempi medi di attesa','Tempo medio di interuscita','Varianza
        tempi medi interuscita','Livello di WIP medio','Varianza livelli di WIP','Numero
        medio di coppie entrate tra due saturazioni buffer']
   PRINT_TO_MULTI_FINAL_FILE('multi_buff', F_NAME, F_EXT_MS,
        F_DELIMITER, TABLE_TEXT, DESCRIPTION, FINAL_ARRAY,
```

```
                FINAL,DATA_SERV)
        elif WHO_CALLS=='multi_nb' or len(DATA_MS)==6:
            F_NAME='%s_risultati finali su %d simulazioni, %d linee, %s %s tempo medio, lotto
                %d unità'%(FILE_NAME_COD_MS, N_SIM_MS, N_LINES_MS,
                mp.nstr(T_UPSTREAM_MS,4), METER_MS, N_MS)
            DESCRIPTION=['Tempo medio di interarrivo','Varianza tempi medi interarrivo','Tempo
                medio di attesa','Varianza tempi medi di attesa','Tempo medio di interuscita','Varianza
                tempi medi interuscita','Livello di WIP medio','Varianza livelli di WIP']
            PRINT_TO_MULTI_FINAL_FILE('multi_n', F_NAME, F_EXT_MS, F_DELIMITER,
                TABLE_TEXT, DESCRIPTION, FINAL_ARRAY, FINAL, DATA_SERV)
        #end if

        if not(WHO_CALLS=='sched' or WHO_CALLS=='comp'):
            if not(CTRL_MS==-1):
                print('Operazione conclusa')
                SERV_MS=INPUT_MENU('er_sub',0)
                if SERV_MS[0]==1:
                    print('Raggiunto limite di errori in immissione')
                    CTRL_MS=-1
                else:
                    if SERV_MS[1]==1:
                        CTRL_MS=0
                    else:
                        CTRL_MS=-1
                    #end if
                #end if
            #end if
        else:
            CTRL_MS=-1
        #end if
    #end if
#end while
if not(WHO_CALLS=='sched' or WHO_CALLS=='comp'):
    input('Reindirizzamento a menù precedente')
    OUT_MS=[0,0]
elif WHO_CALLS=='sched':
    OUT_MS=[0,0]
#end if
return(OUT_MS)
#end MULTI_SIM()
def USER_SIMULATION_MAIN():
#Funzione custom adibita alla gestione del menù contenente le operazioni di simulazione eseguibili
#in modalità utente.
#Viene richiamata dalla funzione custom USER_MAIN.
#L'immissione dati avverrà tramite la funzione custom INPUT_MENU.
#In base alla scelta verranno richiamate le funzioni custom SINGLE_SIM o MULTI_SIM.
    CTRL_USM=0
    while not(CTRL_USM==-1):
        SERV_TEXT_USM=[]
        SERV_USM=INPUT_MENU('user_simulation',0)
```

```
        if SERV_USM[0]==1:
            print('Raggiunto limite di errori in immissione')
            CTRL_USM=-1
        else:
            CTRL_USM=SERV_USM[1]
            if CTRL_USM==1:
                SINGLE_SIM('single_nb',[])
            elif CTRL_USM==2:
                SINGLE_SIM('single_buff',[])
            elif CTRL_USM==3:
                MULTI_SIM('multi_nb',[])
            elif CTRL_USM==4:
                MULTI_SIM('multi_buff',[])
            elif CTRL_USM==0:
                CTRL_USM=-1
            #end if
        #end if
        if SERV_USM==-1:
            print('Operazione non completata correttamente')
        else:
            print('Operazione completata correttamente')
    #end while
    input('Reindirizzamento a menù precedente')
    return()
#end def USER_SIMULATION_MAIN()

def COMP_TABS(WHO_CALLS, DATA_CMT, DATA_TIMELESS_CMT, ORDERS_CMT):
#Funzione custom deputata alla comparazione dei dati prodotti tramite modellizzatore e simulatore
#di una famiglia di casi specifici in assenza di buffer.
#Può essere richiamata dalla modalità schedulatore o in modalità utente dalla funzione custom
#USER_COMPARATORE_MAIN.
#In modalità utente i dati di input necessari verranno immessi tramite la funzione INPUT_MENU,
#mentre in modalità schedulatore gli verranno forniti tramite il comando di chiamata.
#Il caricamento dei files necessari avverrà tramite la funzione custom
#GENERAL_MATRIX_LOADING.
#Durante l'elaborazione verranno richiamate le funzioni custom NEW_SET_ELABORATION, per
#la modellizzazione, e  MULTI_SIM per la simulazione.
#Tramite la funzione custom PRINT_FINAL_COMP verranno creati e salvati i files di output
#contenenti i parametri prodotti dai due metodi e le differenze percentuali del primo rispetto al
#secondo.
    OUT_CMT=[]
    OUT_CMT.append(0)
    CTRL_CMT=0
    giro=0
    while not(CTRL_CMT==-1):
        giro+=1
        if WHO_CALLS=='sched':
            DATA_LOAD_CMT=[ORDERS_CMT[1][0], ORDERS_CMT[1][1],
                    ORDERS_CMT[1][2]]
            NEW_LOAD=0
```

```
    LOAD_MENU_CTRL=0
    LOAD_ORDER_CMT=1
elif giro==1 and DATA_CMT[0]==0:
    N_TAB_TL_CMT=DATA_TIMELESS_CMT[0]
    TIME_VAR_TL_CMT=DATA_TIMELESS_CMT[1]
    TIME_VAR_MED_TL_CMT=DATA_TIMELESS_CMT[2]
    WIP_NO_BUFF_TL_CMT=DATA_TIMELESS_CMT[3]
    GLOBAL_TL_CMT=DATA_TIMELESS_CMT[4]
    NOBUFF_PARAMETERS_TL_CMT=DATA_TIMELESS_CMT[5]
    NEW_LOAD=1
    LOAD_MENU_CTRL=0
    LOAD_ORDER_CMT=0
elif giro==1 and not(DATA_CMT[0]==0):
    NEW_LOAD=0
    LOAD_MENU_CTRL=1
    LOAD_ORDER_CMT=0
elif giro>1 and DATA_CMT[0]==0:
    NEW_LOAD=1
    LOAD_MENU_CTRL=0
    LOAD_ORDER_CMT=0
elif giro>1 and not(DATA_CMT[0]==0):
    NEW_LOAD=0
    LOAD_MENU_CTRL=1
    LOAD_ORDER_CMT=0
else:
    input('ERRORE SELETTORE INIZIALE DENTRO CICLO WHILE IN COMP_TABS')
    CTRL_CMT=-1
#end if

if NEW_LOAD==1 and not(CTRL_CMT==-1):
    SERV_CMT=INPUT_MENU('new_load_mtx',0)
    if SERV_CMT[0]==1:
        print('Raggiunto limite di errori in immissione')
        CTRL_CMT=-1
    else:
        if SERV_CMT[1]==1:
            LOAD_MENU_CTRL=1
        else:
            LOAD_MENU_CTRL=0
        #end if
    #end if
#end if
if LOAD_MENU_CTRL==1 and not(WHO_CALLS=='sched') and not(CTRL_CMT==-1):
    SERV_CMT=INPUT_MENU('timeless_inp', [0])
    if SERV_CMT[0]==1:
        print('Raggiunto limite di errori in immissione')
        CTRL_CMT=-1
    else:
        DATA_LOAD_CMT=[SERV_CMT[1], SERV_CMT[2],0]
        LOAD_ORDER_CMT=1
```

```
   #end if
#end if
if LOAD_ORDER_CMT==1 and not(CTRL_CMT==-1):
   SERV_CMT=GENERAL_MATRIX_LOADING('comp',DATA_LOAD_CMT)
   if not(SERV_CMT[0]==1):
      CTRL_CMT=-1
      input('Errore in apertura files dati timeless')
   else:
      TIME_VAR_TL_CMT=SERV_CMT[1]
      TIME_VAR_MED_TL_CMT=SERV_CMT[2]
      WIP_NO_BUFF_TL_CMT=SERV_CMT[3]
      GLOBAL_TL_CMT=SERV_CMT[4]
      NOBUFF_PARAMETERS_TL_CMT=SERV_CMT[5]
      N_TAB_TL_CMT=TIME_VAR_MED_TL_CMT.cols
      DATA_TIMELESS_CMT=[N_TAB_TL_CMT, TIME_VAR_TL_CMT,
            TIME_VAR_MED_TL_CMT, WIP_NO_BUFF_TL_CMT, GLOBAL_TL_CMT,
            NOBUFF_PARAMETERS_TL_CMT]
   #end if
#end if
if not(CTRL_CMT==-1):
   if WHO_CALLS=='sched':
      N_START_CMT=ORDERS_CMT[2][0]
      T_UPSTREAM_CMT=ORDERS_CMT[2][1]
      METER_CMT=ORDERS_CMT[2][2]
      PREC_CMT=ORDERS_CMT[2][3]
      N_SIM_CMT=ORDERS_CMT[3]
      ROUNDS_CMT=ORDERS_CMT[4][0]
      STEP_CMT=ORDERS_CMT[4][1]
      DATA_TABS_CMT=[N_START_CMT, T_UPSTREAM_CMT, METER_CMT,
            PREC_CMT]
   else:
      SERV_CMT=INPUT_MENU('comp_data_tabs',0)
      if SERV_CMT[0]==1:
         print('Raggiunto limite di errori in immissione')
         CTRL_CMT=-1
      else:
         ROUNDS_CMT=SERV_CMT[1]
         STEP_CMT=SERV_CMT[2]
         print('INSERIMENTO DATI TABULATO INIZIALE')
         SERV_CMT=INPUT_MENU('tab_gen',[N_TAB_TL_CMT-((ROUNDS_CMT-1)
            *STEP_CMT)])
         if SERV_CMT[0]==1:
            print('Raggiunto limite di errori in immissione')
            CTRL_CMT=-1
         else:
            N_START_CMT=SERV_CMT[1]
            T_UPSTREAM_CMT=SERV_CMT[2]
            METER_CMT=SERV_CMT[3]
            PREC_CMT=SERV_CMT[4]
            DATA_TABS_CMT=[N_START_CMT, T_UPSTREAM_CMT, METER_CMT,
```

```
                PREC_CMT]
            SERV_CMT=INPUT_MENU('sim_data_m',0)
            if SERV_CMT[0]==1:
               print('Raggiunto limite di errori in immissione')
               CTRL_CMT=-1
            else:
               N_SIM_CMT=SERV_CMT[1]

            #end if
         #end if
      #end if
   #end if
#end if
if not(CTRL_CMT==-1):
   FILE_COD_NAME_CMT=('%sto%su_%s%s_%dsim'%(N_START_CMT,
         N_START_CMT+(STEP_CMT*(ROUNDS_CMT-1)),
         mp.nstr(T_UPSTREAM_CMT,5), METER_CMT, N_SIM_CMT))

   DATA_NSE_CMT=[]
   DATA_NSE_CMT.append(0)
   for i_data in range(len(DATA_TIMELESS_CMT)):
      DATA_NSE_CMT.append(DATA_TIMELESS_CMT[i_data])
   #end for i_data
   for i_data in range(len(DATA_TABS_CMT)):
      DATA_NSE_CMT.append(DATA_TABS_CMT[i_data])
   #end for i_data
   T_ENTRY_FINAL_CMT=mp.zeros(11,ROUNDS_CMT)
   T_WAIT_FINAL_CMT=mp.zeros(13,ROUNDS_CMT)
   T_EXIT_FINAL_CMT=mp.zeros(13,ROUNDS_CMT)
   WIP_FINAL_CMT=mp.zeros(10,ROUNDS_CMT)
   DATA_MS_CMT=[2, T_UPSTREAM_CMT, METER_CMT, N_START_CMT,
         PREC_CMT, N_SIM_CMT]
   for i_count in range(ROUNDS_CMT):
      SERV_CMT=NEW_SET_ELABORATION('sched',DATA_NSE_CMT)
      if SERV_CMT[0]==0:
         input('ERRORE DOPO CHIAMATA NEW SET')
         CTRL_CTM=-1
         break
      else:
         SERV2_CMT=MULTI_SIM('comp',DATA_MS_CMT)
         if SERV2_CMT[0]==0:
            input('ERRORE DOPO CHIAMATA MULTI_SIM')
            CTRL_CMT=-1
            break
      #end if
      SERV_MTX_CMT=SERV2_CMT[1]
      SERV_MTX2_CMT=SERV_CMT[6]
      T_ENTRY_FINAL_CMT[0,i_count]=DATA_NSE_CMT[7]
      T_ENTRY_FINAL_CMT[1,i_count]=SERV_CMT[3][0]
      T_ENTRY_FINAL_CMT[2,i_count]=SERV_CMT[5][0,0]
```

```
T_ENTRY_FINAL_CMT[3,i_count]=SERV_CMT[3][1]
T_ENTRY_FINAL_CMT[4,i_count]=SERV_CMT[5][0,1]
T_ENTRY_FINAL_CMT[5,i_count]=SERV_MTX_CMT[0]
T_ENTRY_FINAL_CMT[6,i_count]=SERV_MTX_CMT[1]

T_ENTRY_FINAL_CMT[7,i_count]=(T_ENTRY_FINAL_CMT[1,i_count]-
    T_ENTRY_FINAL_CMT[5,i_count])/T_ENTRY_FINAL_CMT[5,i_count]
T_ENTRY_FINAL_CMT[8,i_count]=(T_ENTRY_FINAL_CMT[2,i_count]-
    T_ENTRY_FINAL_CMT[5,i_count])/T_ENTRY_FINAL_CMT[5,i_count]
T_ENTRY_FINAL_CMT[9,i_count]=(T_ENTRY_FINAL_CMT[3,i_count]-
    T_ENTRY_FINAL_CMT[6,i_count])/T_ENTRY_FINAL_CMT[6,i_count]
T_ENTRY_FINAL_CMT[10,i_count]=(T_ENTRY_FINAL_CMT[4,i_count]-
    T_ENTRY_FINAL_CMT[6,i_count])/T_ENTRY_FINAL_CMT[6,i_count]

T_WAIT_FINAL_CMT[0,i_count]=DATA_NSE_CMT[7]
T_WAIT_FINAL_CMT[1,i_count]=SERV_CMT[3][2]
T_WAIT_FINAL_CMT[2,i_count]=SERV_MTX2_CMT[3,2]
T_WAIT_FINAL_CMT[3,i_count]=SERV_CMT[5][0,2]
T_WAIT_FINAL_CMT[4,i_count]=SERV_CMT[3][3]
T_WAIT_FINAL_CMT[5,i_count]=SERV_CMT[5][0,3]
T_WAIT_FINAL_CMT[6,i_count]=SERV_MTX_CMT[2]
T_WAIT_FINAL_CMT[7,i_count]=SERV_MTX_CMT[3]

T_WAIT_FINAL_CMT[8,i_count]=(T_WAIT_FINAL_CMT[1,i_count]-
    T_WAIT_FINAL_CMT[6,i_count])/T_WAIT_FINAL_CMT[6,i_count]
T_WAIT_FINAL_CMT[9,i_count]=(T_WAIT_FINAL_CMT[2,i_count]-
    T_WAIT_FINAL_CMT[6,i_count])/T_WAIT_FINAL_CMT[6,i_count]
T_WAIT_FINAL_CMT[10,i_count]=(T_WAIT_FINAL_CMT[3,i_count]-
    T_WAIT_FINAL_CMT[6,i_count])/T_WAIT_FINAL_CMT[6,i_count]
T_WAIT_FINAL_CMT[11,i_count]=(T_WAIT_FINAL_CMT[4,i_count]-
    T_WAIT_FINAL_CMT[7,i_count])/T_WAIT_FINAL_CMT[7,i_count]
T_WAIT_FINAL_CMT[12,i_count]=(T_WAIT_FINAL_CMT[5,i_count]-
    T_WAIT_FINAL_CMT[7,i_count])/T_WAIT_FINAL_CMT[7,i_count]

T_EXIT_FINAL_CMT[0,i_count]=DATA_NSE_CMT[7]
T_EXIT_FINAL_CMT[1,i_count]=SERV_CMT[3][4]
T_EXIT_FINAL_CMT[2,i_count]=1/SERV_MTX2_CMT[0,1]
T_EXIT_FINAL_CMT[3,i_count]=SERV_CMT[5][0,4]
T_EXIT_FINAL_CMT[4,i_count]=SERV_CMT[3][5]
T_EXIT_FINAL_CMT[5,i_count]=SERV_CMT[5][0,5]
T_EXIT_FINAL_CMT[6,i_count]=SERV_MTX_CMT[4]
T_EXIT_FINAL_CMT[7,i_count]=SERV_MTX_CMT[5]

T_EXIT_FINAL_CMT[8,i_count]=(T_EXIT_FINAL_CMT[1,i_count]-
    T_EXIT_FINAL_CMT[6,i_count])/T_EXIT_FINAL_CMT[6,i_count]
T_EXIT_FINAL_CMT[9,i_count]=(T_EXIT_FINAL_CMT[2,i_count]-
    T_EXIT_FINAL_CMT[6,i_count])/T_EXIT_FINAL_CMT[6,i_count]
T_EXIT_FINAL_CMT[10,i_count]=(T_EXIT_FINAL_CMT[3,i_count]-
    T_EXIT_FINAL_CMT[6,i_count])/T_EXIT_FINAL_CMT[6,i_count]
```

```
        T_EXIT_FINAL_CMT[11,i_count]=(T_EXIT_FINAL_CMT[4,i_count]-
            T_EXIT_FINAL_CMT[7,i_count])/T_EXIT_FINAL_CMT[7,i_count]
        T_EXIT_FINAL_CMT[12,i_count]=(T_EXIT_FINAL_CMT[5,i_count]-
            T_EXIT_FINAL_CMT[7,i_count])/T_EXIT_FINAL_CMT[7,i_count]

        WIP_FINAL_CMT[0,i_count]=DATA_NSE_CMT[7]
       WIP_FINAL_CMT[1,i_count]=SERV_MTX2_CMT[0,0]
        WIP_FINAL_CMT[2,i_count]=SERV_MTX2_CMT[1,0]
        WIP_FINAL_CMT[3,i_count]=SERV_MTX2_CMT[2,0]
        WIP_FINAL_CMT[4,i_count]=SERV_MTX2_CMT[4,0]
        WIP_FINAL_CMT[5,i_count]=SERV_MTX_CMT[6]

        WIP_FINAL_CMT[6,i_count]=(WIP_FINAL_CMT[1,i_count]-
            WIP_FINAL_CMT[5,i_count])/WIP_FINAL_CMT[5,i_count]
        WIP_FINAL_CMT[7,i_count]=(WIP_FINAL_CMT[2,i_count]-
            WIP_FINAL_CMT[5,i_count])/WIP_FINAL_CMT[5,i_count]
        WIP_FINAL_CMT[8,i_count]=(WIP_FINAL_CMT[3,i_count]-
            WIP_FINAL_CMT[5,i_count])/WIP_FINAL_CMT[5,i_count]
        WIP_FINAL_CMT[9,i_count]=(WIP_FINAL_CMT[4,i_count]-
            WIP_FINAL_CMT[5,i_count])/WIP_FINAL_CMT[5,i_count]

        DATA_NSE_CMT[7]+=STEP_CMT
        DATA_MS_CMT[3]+=STEP_CMT
    #end for i_count
#end if

if not(CTRL_CMT==-1):
    DATA_PRINT_CMT=[]
    DATA_PRINT_CMT.append(['comp_tabs_ta',[0,1,5,7],[0,2,5,8],[0,3,6,9],
            [0,4,6,10],T_ENTRY_FINAL_CMT])
    DATA_PRINT_CMT.append(['comp_tabs_tw',[0,1,6,8],[0,2,6,9],[0,3,6,10],[0,4,7,11],
            [0,5,7,12],T_WAIT_FINAL_CMT])
    DATA_PRINT_CMT.append(['comp_tabs_tu',[0,1,6,8],[0,2,6,9],[0,3,6,10],[0,4,7,11],
            [0,5,7,12],T_EXIT_FINAL_CMT])
    DATA_PRINT_CMT.append(['comp_tabs_wip',[0,1,5,6],[0,2,5,7],[0,3,5,8],
            [0,4,5,9],WIP_FINAL_CMT])
    PRINT_FINAL_COMP('comp_tabs', FILE_COD_NAME_CMT, DATA_PRINT_CMT)
    OUT_CMT[0]=1
    DATA_CMT[0]=0
#end if
if not(WHO_CALLS=='sched'):
    if not(CTRL_CMT==-1):
        print('Operazione conclusa')
        SERV_CMT=INPUT_MENU('er_sub',0)
        if SERV_CMT[0]==1:
            print('Raggiunto limite di errori in immissione')
            CTRL_CMT=-1
        else:
            if SERV_CMT[1]==1:
                CTRL_CMT=0
```

```
                else:
                    CTRL_CMT=-1
                #end if
            #end if
        #end if
    else:
        CTRL_CMT=-1
    #end if
#end while
if not(WHO_CALLS=='sched'):
    print('Reindirizzamento a menù precedente')
    if OUT_CMT[0]==1:
        OUT_CMT.append(DATA_CMT)
        OUT_CMT.append(DATA_TIMELESS_CMT)
    #end if
else:
    OUT_CMT[0]=0
#end if
return(OUT_CMT)
#end def COMP_TABS()

def COMP_BFV(WHO_CALLS, DATA_CBV, DATA_TABS_CBV, ORDERS_CBV):
#Funzione custom deputata alla comparazione dei dati prodotti tramite modellizzatore e simulatore
#di una famiglia di casi specifici in presenza di buffer.
#Può essere richiamata dalla modalità schedulatore o in modalità utente dalla funzione custom
#USER_COMPARATORE_MAIN.
#In modalità utente i dati di input necessari verranno immessi tramite la funzione INPUT_MENU,
#mentre in modalità schedulatore gli verranno forniti tramite il comando di chiamata.
#Il caricamento dei files necessari avverrà tramite la funzione custom
#GENERAL_MATRIX_LOADING.
#Durante l'elaborazione verranno richiamate le funzioni custom BUFFER_ELABORATION, per la
#modellizzazione, e  MULTI_SIM per la simulazione.
#Tramite la funzione custom PRINT_FINAL_COMP verranno creati e salvati i files di output
#contenenti i parametri prodotti dai due metodi e le differenze percentuali del primo rispetto al
#secondo.
    OUT_CBV=[]
    OUT_CBV.append(0)
    CTRL_CBV=0
    giro=0
    while not(CTRL_CBV==-1):
        giro+=1
        if WHO_CALLS=='sched':
            DATA_LOAD_CBV=[ORDERS_CBV[1][0], ORDERS_CBV[1][1], ORDERS_CBV[1][2],
                    ORDERS_CBV[1][3]]
            NEW_LOAD=0
            LOAD_MENU_CTRL=0
            LOAD_ORDER_CBV=1
        elif giro==1 and DATA_CBV[1]==0:
            N_T_M_CBV=DATA_TABS_CBV[0]
            N_T_M_CBV[0]=int(N_T_M_CBV[0])
```

```
    N_T_M_CBV[1]=mp.mpf(N_T_M_CBV[1])
    TIME_VAR_CBV=DATA_TABS_CBV[1]
    TIME_VAR_MED_CBV=DATA_TABS_CBV[2]
    WIP_NO_BUFF_CBV=DATA_TABS_CBV[3]
    GLOBAL_CBV=DATA_TABS_CBV[4]
    NOBUFF_PARAMETERS_CBV=DATA_TABS_CBV[5]
    NEW_LOAD=1
    LOAD_MENU_CTRL=0
    LOAD_ORDER_CBV=0
elif giro==1 and not(DATA_CBV[1]==0):
    NEW_LOAD=0
    LOAD_MENU_CTRL=1
    LOAD_ORDER_CBV=0
elif giro>1 and DATA_CBV[1]==0:
    NEW_LOAD=1
    LOAD_MENU_CTRL=0
    LOAD_ORDER_CBV=0
elif giro>1 and not(DATA_CBV[1]==0):
    NEW_LOAD=0
    LOAD_MENU_CTRL=1
    LOAD_ORDER_CBV=0
else:
    input('ERRORE SELETTORE INIZIALE DENTRO CICLO WHILE IN COMP_TABS')
    CTRL_CBV=-1
#end if

if NEW_LOAD==1 and not(CTRL_CBV==-1):
    SERV_CBV=INPUT_MENU('new_load_tabs',0)
    if SERV_CBV[0]==1:
        print('Raggiunto limite di errori in immissione')
        CTRL_CBV=-1
    else:
        if SERV_CBV[1]==1:
            LOAD_MENU_CTRL=1
        else:
            LOAD_MENU_CTRL=0
        #end if
    #end if
#end if
if LOAD_MENU_CTRL==1 and not(WHO_CALLS=='sched') and not(CTRL_CBV==-1):
    SERV_CBV=INPUT_MENU('tab_inp',[0])
    if SERV_CBV[0]==1:
        print('Raggiunto limite di errori in immissione')
        CTRL_CBV=-1
    else:
        LOAD_ORDER_CBV=1
        DATA_LOAD_CBV=[SERV_CBV[1], SERV_CBV[2], SERV_CBV[3], SERV_CBV[4]]
    #end if
#end if
if LOAD_ORDER_CBV==1 and not(CTRL_CBV==-1):
```

```
    SERV_CBV=GENERAL_MATRIX_LOADING('comp',DATA_LOAD_CBV)
    if not(SERV_CBV[0]==1):
        CTRL_CBV=-1
        input('errore in apertura matrici di base')
    else:
        N_T_M_CBV=SERV_CBV[1]
        N_T_M_CBV[0]=int(N_T_M_CBV[0])
        N_T_M_CBV[1]=mp.mpf(N_T_M_CBV[1])
        TIME_VAR_CBV=SERV_CBV[2]
        TIME_VAR_MED_CBV=SERV_CBV[3]
        WIP_NO_BUFF_CBV=SERV_CBV[4]
        GLOBAL_CBV=SERV_CBV[5]
        NOBUFF_PARAMETERS_CBV=SERV_CBV[6]
        DATA_TABS_CBV=[N_T_M_CBV, TIME_VAR_CBV, TIME_VAR_MED_CBV,
            WIP_NO_BUFF_CBV, GLOBAL_CBV, NOBUFF_PARAMETERS_CBV]
    #end if
#end if
if not(CTRL_CBV==-1):
    if WHO_CALLS=='sched':
        BUFF_START_CBV=ORDERS_CBV[2][0]
        PREC_CBV=ORDERS_CBV[2][1]
        N_SIM_CBV=ORDERS_CBV[3]
        ROUNDS_CBV=ORDERS_CBV[4][0]
        STEP_CBV=ORDERS_CBV[4][1]
    else:
        print('INSERIMENTO BUFFER INIZIALE E PRECISIONE DESIDERATA')
        SERV_CBV=INPUT_MENU('comp_data_buff1',[N_T_M_CBV[0]-4])
        if SERV_CBV[0]==1:
            print('Raggiunto limite di errori in immissione')
            CTRL_CBV=-1
        else:
            ROUNDS_CBV=SERV_CBV[1]
            SERV_CBV=INPUT_MENU('comp_data_buff2',[int((N_T_M_CBV[0]-4)/
              (ROUNDS_CBV)-1)])
            if SERV_CBV[0]==1:
                print('Raggiunto limite di errori in immissione')
                CTRL_CBV=-1
            else:
                STEP_CBV=SERV_CBV[1]
                SERV_CBV=INPUT_MENU('buff_gen',[(N_T_M_CBV[0]-4)-
                        ((ROUNDS_CBV-1)*STEP_CBV)])
                if SERV_CBV[0]==1:
                    print('Raggiunto limite di errori in immissione')
                    CTRL_CBV=-1
                else:
                    BUFF_START_CBV=SERV_CBV[1]
                    PREC_CBV=SERV_CBV[2]
                    SERV_CBV=INPUT_MENU('sim_data_m',0)
                    if SERV_CBV[0]==1:
                        print('Raggiunto limite di errori in immissione')
```

```
                CTRL_CBV=-1
            else:
                N_SIM_CBV=SERV_CBV[1]
            #end if
        #end if
    #end if
#end if
if not(CTRL_CBV==-1):
    FILE_COD_NAME_CBV=('%du_%sto%sbuff_%s%s_%dsim'%(N_T_M_CBV[0],
        BUFF_START_CBV, BUFF_START_CBV+(STEP_CBV*(ROUNDS_CBV-1)),
        mp.nstr(N_T_M_CBV[1],5), N_T_M_CBV[2], N_SIM_CBV))

    T_WAIT_FINAL_CBV=mp.zeros(4,ROUNDS_CBV)
    T_EXIT_FINAL_CBV=mp.zeros(4,ROUNDS_CBV)
    WIP_FINAL_CBV=mp.zeros(7,ROUNDS_CBV)
    DATA_BUFF_CBV=[0, N_T_M_CBV, TIME_VAR_CBV, TIME_VAR_MED_CBV,
        WIP_NO_BUFF_CBV, GLOBAL_CBV, NOBUFF_PARAMETERS_CBV,
        BUFF_START_CBV, PREC_CBV]
    DATA_MBUFF_CBV=[2, N_T_M_CBV[1], N_T_M_CBV[2], N_T_M_CBV[0],
        PREC_CBV, N_SIM_CBV, BUFF_START_CBV]
    for i_count in range(ROUNDS_CBV):
        SERV_CBV=BUFFER_ELABORATION('comp',DATA_BUFF_CBV)
        if SERV_CBV[0]==0:
            input('ERRORE DOPO CHIAMATA BUFFER_ELABORATION')
            CTRL_CBV=-1
            break
        else:
            SERV2_CBV=MULTI_SIM('comp',DATA_MBUFF_CBV)
            if not(SERV2_CBV[0]==1):
                input('ERRORE DOPO CHIAMATA MULTI_SIM')
                CTRL_CBV=-1
                break
        #end if
        SERV_BUFF_CBV=SERV_CBV[8]
        SERV_MBUFF_CBV=SERV2_CBV[1]

        T_WAIT_FINAL_CBV[0,i_count]=DATA_MBUFF_CBV[6]
        T_WAIT_FINAL_CBV[1,i_count]=SERV_BUFF_CBV[2]
        T_WAIT_FINAL_CBV[2,i_count]=SERV_MBUFF_CBV[2]

        T_WAIT_FINAL_CBV[3,i_count]=(T_WAIT_FINAL_CBV[1,i_count]-
            T_WAIT_FINAL_CBV[2,i_count])/T_WAIT_FINAL_CBV[2,i_count]

        T_EXIT_FINAL_CBV[0,i_count]=DATA_MBUFF_CBV[6]
        T_EXIT_FINAL_CBV[1,i_count]=1/SERV_BUFF_CBV[1]
        T_EXIT_FINAL_CBV[2,i_count]=SERV_MBUFF_CBV[4]

        T_EXIT_FINAL_CBV[3,i_count]=(T_EXIT_FINAL_CBV[1,i_count]-
            T_EXIT_FINAL_CBV[2,i_count])/T_EXIT_FINAL_CBV[2,i_count]
```

```
            WIP_FINAL_CBV[0,i_count]=DATA_MBUFF_CBV[6]
            WIP_FINAL_CBV[1,i_count]=SERV_BUFF_CBV[0]
            WIP_FINAL_CBV[2,i_count]=SERV_BUFF_CBV[3]
            WIP_FINAL_CBV[3,i_count]=SERV_MBUFF_CBV[6]
            WIP_FINAL_CBV[4,i_count]=SERV_MBUFF_CBV[8]
            WIP_FINAL_CBV[5,i_count]=(WIP_FINAL_CBV[1,i_count]-
                WIP_FINAL_CBV[3,i_count])/WIP_FINAL_CBV[3,i_count]
            WIP_FINAL_CBV[6,i_count]=(WIP_FINAL_CBV[2,i_count]-
                WIP_FINAL_CBV[4,i_count])/WIP_FINAL_CBV[4,i_count]
            DATA_BUFF_CBV[7]+=STEP_CBV
            DATA_MBUFF_CBV[6]+=STEP_CBV
        #end for i_count
    #end if
    if not(CTRL_CBV==-1):
        DATA_PRINT_CBV=[]
        DATA_PRINT_CBV.append(['comp_buff_tw',[0,1,2,3], T_WAIT_FINAL_CBV])
        DATA_PRINT_CBV.append(['comp_buff_tu',[0,1,2,3], T_EXIT_FINAL_CBV])
        DATA_PRINT_CBV.append(['comp_buff_wip',[0,1,3,5],[0,2,4,6], WIP_FINAL_CBV])
        PRINT_FINAL_COMP('comp_buff', FILE_COD_NAME_CBV, DATA_PRINT_CBV)
        OUT_CBV[0]=1
        DATA_CBV[1]=0
    #end if
    if not(WHO_CALLS=='sched'):
        if not(CTRL_CBV==-1):
            print('Operazione conclusa')
            SERV_CBV=INPUT_MENU('er_sub',0)
            if SERV_CBV[0]==1:
                print('Raggiunto limite di errori in immissione')
                CTRL_CBV=-1
            else:
                if SERV_CBV[1]==1:
                    CTRL_CBV=0
                else:
                    CTRL_CBV=-1
                #end if
            #end if
        #end if
    else:
        CTRL_CBV=-1
    #end if
#end while
if not(WHO_CALLS=='sched'):
    print('Reindirizzamento a menù precedente')
    if OUT_CBV[0]==1:
        OUT_CBV.append(DATA_CBV)
        OUT_CBV.append(DATA_TABS_CBV)
    #end if
else:
    OUT_CBV[0]=0
```

```
    #end if
    return(OUT_CBV)
#end def COMP_BFV(WHO_CALLS_SUB, DATA_CM, DATA_TABS_CM, ORDERS_CM)

def USER_COMPARATORE_MAIN(WHO_CALLS,ORDERS_CM):
#Funzione custom dedicata alla gestione del menù tramite il quale si sceglierà quale operazione di
#comparazione dati eseguire
#Può essere richiamata in modalità utente dalla funzione custom USER_MAIN o in modalità
#schedulatore.
#Mantiene al suo interno le matrici che possono essere utili al funzionamento dei blocchi interni
#fino all'abbandono della sezione da parte dell'utente o dello schedulatore.
#La parte grafica e di immissione dati del menù viene gestita dalla funzione custom
#INPUT_MENU.
#Se la scelta ricade sulla comparazione di una famiglia di casi specifici in assenza di buffer verrà
#richiamata la funzione custom  COMP_TABS, altrimenti  COMP_BFV per la comparazione di una
#famiglia di casi specifici in presenza di buffer.
    TIMELESS_STATUS_CM=1
    TABS_STATUS_CM=1
    TIME_TABS_CM=0
    METER_TABS_CM=''
    DATA_CM=[TIMELESS_STATUS_CM, TABS_STATUS_CM]
    DATA_TIMELESS_CM=[]
    DATA_TABS_CM=[]
    CTRL_CM=0
    while CTRL_CM==0:
        if WHO_CALLS=='sched':
            if ORDERS_CM[0]=='com_tab_nb':
                CTRL_CM=1
            elif ORDERS_CM[0]=='com_tab_bfvar':
                CTRL_CM=2
            elif ORDERS_CM[0]=='com_tab_bffix':
                CTRL_CM=3
            #end if
            WHO_CALLS_SUB='sched'
        else:
            SERV_CM=INPUT_MENU('main_comp',0)
            if SERV_CM[0]==1:
                print('Raggiunto limite errori in immissione dati')
                CTRL_CM=-1
            else:
                CTRL_CM=SERV_CM[1]
                WHO_CALLS_SUB=0
                ORDERS=0
            #end if
        #end if
        if not(CTRL_CM==-1):
            if CTRL_CM==1:
                SERV_CM=COMP_TABS(WHO_CALLS_SUB, DATA_CM, DATA_TIMELESS_CM,
                      ORDERS_CM)
                if SERV_CM[0]==1:
```

```
                DATA_CM=SERV_CM[1]
                DATA_TIMELESS_CM=SERV_CM[2]
                CTRL_CM=0
            #end if
            CTRL_CM=0
        elif CTRL_CM==2:
            SERV_CM=COMP_BFV(WHO_CALLS_SUB, DATA_CM, DATA_TABS_CM,
                ORDERS_CM)
            if SERV_CM[0]==1:
                CTRL_CM=0
                DATA_CM=SERV_CM[1]
                DATA_TABS_CM=SERV_CM[2]
            #end if
            CTRL_CM=0
        elif CTRL_CM==3:
            print('NON IMPLEMENTATO')
            CTRL_CM=0
        elif CTRL_CM==0:
            CTRL_CM=-1
        #end if
    #end if
    if WHO_CALLS=='sched':
        CTRL_CM=-1
    #end if
  #end while
  if not(WHO_CALLS=='sched'):
    print('Reindirizzamento a menù precedente')
  #end if
  return()
#end USER_COMPARATORE_MAIN()

def USER_MAIN():
#Funzione custom deputata alla gestione del menu contenente la scelta della famiglia di operazioni
#in modalità utente a cui si desidera accedere.
#La parte grafica e di immissione dati del menù viene gestita dalla funzione custom
#INPUT_MENU.
#Se la scelta ricade sulla sezione di modellizzazione teorica verrà richiamata la funzione custom
#USER_THEORY_MAIN, USER_SIMULATION_MAIN se si desidera eseguire delle operazioni
#di simulazione e USER_COMPARATORE_MAIN per le operazioni di comparazione.
  CTRL_UM=0
  while CTRL_UM==0:
    SERV=INPUT_MENU('user_main',0)
    ERROR_CTRL=SERV[0]
    if ERROR_CTRL==1:
        print('Limite errori di immissione dati raggiunto')
        CTRL_UM=-1
    elif ERROR_CTRL==0:
        CTRL_UM=SERV[1]
        if CTRL_UM==1:
            USER_THEORY_MAIN()
```

```
            CTRL_UM=0
        elif CTRL_UM==2:
            USER_SIMULATION_MAIN()
            CTRL_UM=0
        elif CTRL_UM==3:
            USER_COMPARATORE_MAIN(0,0)
            CTRL_UM=0
        elif CTRL_UM==0:
            CTRL_UM=-1
        #end if
      #end if
   #end while
   print('Reindirizzamento a menù precedente')
#end def USER_MAIN()


def SRC_SELECT_FIELD(TYPE_SSF,CHOICE_SSF):
#Funzione custom che gestisce alcuni parametri interni di SCHED_CYCLE_SIM.
   if CHOICE_SSF==1:
      OUT_SSF=4
   elif CHOICE_SSF==2:
      OUT_SSF=2
   elif CHOICE_SSF==3:
      OUT_SSF=1
   elif CHOICE_SSF==4 and (TYPE_SSF=='multi_nb' or TYPE_SSF=='multi_buff'):
      OUT_SSF=6
   elif CHOICE_SSF==4 and TYPE_SSF=='single_buff':
      OUT_SSF=6
   elif CHOICE_SSF==5 and TYPE_SSF=='multi_buff':
      OUT_SSF=7
   else:
      input('PROBLEMA IN SRC_SELECT_FIELD')
   #end if
   return(OUT_SSF)
#end SRC_SELECT_FIELD

def SCHED_CYCLE_SIM(TYPE_SC,DATA_SC):
#Funzione adibita a generare un ciclo di istruzioni della famiglia "comparazione" in maniera più
#rapida.
#Viene richiamata da SCHED_WRITE_SIM.
#Si avvale di INPUT_MENU per l'inserimento e controllo dei dati e di SRC_SELECT_FIELD per
#alcune operazioni specifiche.
   CTRL_SC=0
   print('\nINSERIMENTO DATI PER CREAZIONE ELABORAZIONE CICLO SIMULAZIONI:
         DATI SIMULAZIONE DI PARTENZA')
   DATA_SC=INPUT_MENU('sim_data',0)
   if DATA_SC[0]==0:
      if TYPE_SC=='single_buff':
         SERV2_SC=INPUT_MENU('sim_data_b',[DATA_SC[4]])
         if SERV2_SC[0]==0:
```

```
            DATA_SC.append(SERV2_SC[1])
         else:
            DATA_SC[0]=1
         #end if
      elif TYPE_SC=='multi_nb':
         SERV2_SC=INPUT_MENU('sim_data_m',0)
         if SERV2_SC[0]==0:
            DATA_SC.append(SERV2_SC[1])
         else:
            DATA_SC[0]=1
         #end if
      elif TYPE_SC=='multi_buff':
         SERV2_SC=INPUT_MENU('sim_data_m',0)
         if SERV2_SC[0]==0:
            DATA_SC.append(SERV2_SC[1])
            SERV2_SC=INPUT_MENU('sim_data_b',[DATA_SC[4]])
            if SERV2_SC[0]==0:
               DATA_SC.append(SERV2_SC[1])
            else:
               DATA_SC[0]=1
            #end if
         else:
            DATA_SC[0]=1
         #end if
      #end if
   #end if
   if DATA_SC[0]==1:
      print('Raggiunto limite massimo errori in immissione dati')
      OUT_SC=[1]
   else:
      TEXT_SC=TYPE_SC+'\n'
      SERV_FOR=[]

      if TYPE_SC=='single_nb':
         SERV2_SC=INPUT_MENU('scr_cyc_field_snb',[0])
      elif TYPE_SC=='single_buff':
         SERV2_SC=INPUT_MENU('scr_cyc_field_sbuff',[0])
      elif TYPE_SC=='multi_nb':
         SERV2_SC=INPUT_MENU('scr_cyc_field_mnb',[0])
      elif TYPE_SC=='multi_buff':
         SERV2_SC=INPUT_MENU('scr_cyc_field_mbuff',[0])
      #end if
      if SERV2_SC[0]==1:
         print('Raggiunto limite immissione dati')
         print('Creazione ciclo di comandi interrotta')
         CTRL_SC=-1
      else:
         SERV_FOR.append(SRC_SELECT_FIELD(TYPE_SC,SERV2_SC[1]))
         SERV2_SC=INPUT_MENU('scr_cyc_sim',[0])
         if SERV2_SC[0]==1:
```

```
            print('Raggiunto limite immissione dati')
            print('Creazione ciclo di comandi interrotta')
            CTRL_SC=-1
         else:
            SERV_FOR.append(DATA_SC[5])
            SERV_FOR.append(SERV2_SC[1])
            SERV_FOR.append(SERV2_SC[2])
            SERV_FOR.append(DATA_SC[SERV_FOR[0]])
         #end if
      #end if
      if CTRL_SC==0:
         SERV_OUT=[]
         INCREASING_FIELD=SERV_FOR[4]
         for i_write in range(SERV_FOR[2]):
            SERV_OUT.append(TEXT_SC)
            for j_write in range(len(DATA_SC)-1):
               if j_write+1==SERV_FOR[0] and not(SERV_FOR[0]==2):
                  SERV_OUT.append('%d%s'%(int(INCREASING_FIELD), F_DELIMITER))
               elif j_write+1==SERV_FOR[0] and SERV_FOR[0]==2:
                  SERV_OUT.append('%s%s'%
                        (mp.nstr(mp.mpf(INCREASING_FIELD),SERV_FOR[1]), F_DELIMITER))
               elif j_write+1==2 and not(j_write+1==SERV_FOR[0]):
                  SERV_OUT.append('%s%s'%
                        (mp.nstr(mp.mpf(DATA_SC[j_write+1]),SERV_FOR[1]), F_DELIMITER))
               elif j_write+1==3:
                  SERV_OUT.append('%s%s'%(DATA_SC[j_write+1],F_DELIMITER))
               else:
                  SERV_OUT.append('%d%s'%(int(DATA_SC[j_write+1]), F_DELIMITER))
               #end if
            #end for j_write
            SERV_OUT.append('\n')
            INCREASING_FIELD+=SERV_FOR[3]
         #end for i_write
         TEXT_OUT=''
         for i_write in range(len(SERV_OUT)):
            TEXT_OUT+=SERV_OUT[i_write]
         #end for i_write
         OUT_SC=[0,TEXT_OUT]
      #end if
   #end if
   return(OUT_SC)
#end def SCHED_CYCLE_SIM(TYPE_SC)

def SCHED_WRITE_SIM():
#Funzione custom che gestisce la selezione delle istruzioni della famiglia "simulazione" da
#immettere nel file di schedulazione, controllando la corretta immissione dei dati necessari alla
#futura esecuzione.
#Mantiene in memoria una serie di parametri in modo da garantire la coerenza tra le varie
#istruzioni.
#Viene richiamata da SCHED_WRITE_MAIN.
```

```
#Le varie sessioni di input verranno gestite dalla funzione custom INPUT_MENU.
#Può essere richiamata la funzione custom  SCHED_CYCLE_SIM.
   CTRL_SWS=0
   SERV_LINE=[]
   while CTRL_SWS==0:
      SERV_SWS=INPUT_MENU('scr_opr_sim',0)
      if SERV_SWS[0]==1:
         print('Limite errori in inserimento dati raggiunto')
         CTRL_SWS=-1
      else:
         if SERV_SWS[1]==1:
            SERV_OP_SWS='single_nb\n'
            SERV_SWS=INPUT_MENU('sim_data',0)
            if SERV_SWS[0]==0:
               ERROR_SWS=0
            else:
               ERROR_SWS=1
            #end if
         elif SERV_SWS[1]==2:
            SERV_OP_SWS='single_buff\n'
            SERV2_SWS=INPUT_MENU('sim_data',0)
            if SERV2_SWS[0]==0:
               SERV_SWS=SERV2_SWS
               SERV2_SWS=INPUT_MENU('sim_data_b',[SERV2_SWS[4]])
               if SERV2_SWS[0]==0:
                  SERV_SWS.append(SERV2_SWS[1])
                  if  SERV_SWS[2]=='s' or SERV_SWS[2]=='S' or SERV_SWS[2]=='sec':
                     SERV_SWS[2]='sec'
                  elif SERV_SWS[2]=='m'or SERV_SWS[2]=='M' or SERV_SWS[2]=='min':
                     SERV_SWS[2]='min'
                  elif SERV_SWS[2]=='h' or SERV_SWS[2]=='H':
                     SERV_SWS[2]='h'
                  #end if
                  ERROR_SWS=0
               else:
                  ERROR_SWS=1
               #end if
            else:
               ERROR_SWS=1
            #end if
         elif SERV_SWS[1]==3:
            SERV_OP_SWS='multi_nb\n'
            SERV2_SWS=INPUT_MENU('sim_data',0)
            if SERV2_SWS[0]==0:
               SERV_SWS=SERV2_SWS
               SERV2_SWS=INPUT_MENU('sim_data_m',0)
               if SERV2_SWS[0]==0:
                  SERV_SWS.append(SERV2_SWS[1])
                  if  SERV_SWS[2]=='s' or SERV_SWS[2]=='S' or SERV_SWS[2]=='sec':
                     SERV_SWS[2]='sec'
```

```
            elif SERV_SWS[2]=='m'or SERV_SWS[2]=='M' or SERV_SWS[2]=='min':
               SERV_SWS[2]='min'
            elif SERV_SWS[2]=='h' or SERV_SWS[2]=='H':
               SERV_SWS[2]='h'
            #end if
            ERROR_SWS=0
         else:
            ERROR_SWS=1
         #end if
      else:
         ERROR_SWS=1
      #end if
   elif SERV_SWS[1]==4:
      SERV_OP_SWS='multi_buff\n'
      SERV2_SWS=INPUT_MENU('sim_data',0)
      if SERV2_SWS[0]==0:
         SERV_SWS=SERV2_SWS
         SERV2_SWS=INPUT_MENU('sim_data_m',0)
         if SERV2_SWS[0]==0:
            SERV_SWS.append(SERV2_SWS[1])
            SERV2_SWS=INPUT_MENU('sim_data_b',[SERV_SWS[4]])
            if SERV2_SWS[0]==0:
               SERV_SWS.append(SERV2_SWS[1])
               if  SERV_SWS[2]=='s' or SERV_SWS[2]=='S' or SERV_SWS[2]=='sec':
                  SERV_SWS[2]='sec'
               elif SERV_SWS[2]=='m'or SERV_SWS[2]=='M' or SERV_SWS[2]=='min':
                  SERV_SWS[2]='min'
               elif SERV_SWS[2]=='h' or SERV_SWS[2]=='H':
                  SERV_SWS[2]='h'
               #end if
               ERROR_SWS=0
            else:
               ERROR_SWS=1
            #end if
         else:
            ERROR_SWS=1
         #end if
      else:
         ERROR_SWS=1
      #end if
   elif SERV_SWS[1]==5:
      SERV_SWS=SCHED_CYCLE_SIM('single_nb',0)
      if SERV_SWS[0]==0:
         ERROR_SWS=3
      else:
         ERROR_SWS=1
      #end if
   elif SERV_SWS[1]==6:
      SERV_SWS=SCHED_CYCLE_SIM('single_buff',0)
      if SERV_SWS[0]==0:
```

```
                ERROR_SWS=3
            else:
                ERROR_SWS=1
            #end if
        elif SERV_SWS[1]==7:
            SERV_SWS=SCHED_CYCLE_SIM('multi_nb',0)
            if SERV_SWS[0]==0:
                ERROR_SWS=3
            else:
                ERROR_SWS=1
            #end if
        elif SERV_SWS[1]==8:
            SERV_SWS=SCHED_CYCLE_SIM('multi_buff',0)
            if SERV_SWS[0]==0:
                ERROR_SWS=3
            else:
                ERROR_SWS=1
            #end if
        elif SERV_SWS[1]==0:
            ERROR_SWS=0
            CTRL_SWS=-1
        else:
            input('ERRORE ALL INTERNO MENU SELEZIONE COMANDI')
        #end if
        if ERROR_SWS==1:
            print('Limite errori in inserimento dati raggiunto')
            input('Linea di comando non inserita')
        elif ERROR_SWS==0 and not(CTRL_SWS==-1):
            SERV_LINE.append('%s'%(SERV_OP_SWS))
            for i_write in range(len(SERV_SWS)-1):
                SERV_LINE.append(str(SERV_SWS[i_write+1]))
                SERV_LINE.append(F_DELIMITER)
            #end for
            SERV_LINE.append('\n')
        elif ERROR_SWS==3 and not(CTRL_SWS==-1):
            for i_write in range(len(SERV_SWS[1])):
                SERV_LINE.append(SERV_SWS[1][i_write])
            #end for i_write
        #end if
    #end if
  #end while
  return(SERV_LINE)
#end SCHED_WRITE_SIM()

def SCHED_CYCLE_THEO(TYPE_SC,DATA_SC):
#Funzione adibita a generare un ciclo di istruzioni della famiglia "comparazione" in maniera più
#rapida.
#Viene richiamata da SCHED_WRITE_THEO.
#Si avvale di INPUT_MENU per l'inserimento e controllo dei dati.
  OUT_SC=[]
```

```
SERV_OUT_SC=[]
if TYPE_SC=='mtx':
   print('CREAZIONE COMANDI PER ESEGUIRE UN CICLO DI GENERAZIONE DI
      MATRICI DI BASE\n')
   TYPE2_SC=INPUT_MENU('mtx_cyc_load_gen',0)
   if TYPE2_SC[0]==1:
      print('Limite errori di immissione dati raggiunto')
      CTRL_SC=-1
   else:
      if TYPE2_SC[1]==1:
         SERV_SC=INPUT_MENU('mtx_gen',[0,0])
         if SERV_SC[0]==0:
            SERV_OUT_SC.append('mtx_gen\n')
            SERV_OUT_SC.append('%d%s'%(SERV_SC[1],F_DELIMITER))
            DATA_SC[0]=1
            DATA_SC[1]=SERV_SC[1]
            SERV_MTX_SC=SERV_SC
            CTRL_SC=0
         else:
            print('Limite errori di immissione dati raggiunto')
            CTRL_SC=-1
         #end if
      elif TYPE2_SC[1]==2:
         SERV_SC=INPUT_MENU('mtx_inp',0)
         if SERV_SC[0]==0:
            SERV_OUT_SC.append('mtx_load\n')
            SERV_OUT_SC.append('%d%s%d%s\n'%(SERV_SC[1], F_DELIMITER,
               SERV_SC[2], F_DELIMITER))
            DATA_SC[0]=1
            DATA_SC[1]=SERV_SC[1]
            SERV_MTX_SC=SERV_SC
            CTRL_SC=0
         else:
            print('Limite errori di immissione dati raggiunto')
            CTRL_SC=-1
         #end if
      else:
         CTRL_SC=-2
      #end if
      if not(CTRL_SC==-1 or CTRL_SC==-2):
         SERV_SC=INPUT_MENU('mtx_cyc',0)
         if SERV_SC[0]==1:
            CTRL_SC=-1
            print('Limite errori di immissione dati raggiunto')
         else:
            CTRL_SC=0
            if TYPE2_SC[1]==1:
               SERV_SC[2]-=1
               SERV_OUT_SC.append('%d%s\n'%(SERV_SC[1], F_DELIMITER))
            #end if
```

```
            #end if
          #end if
        #end if
    elif TYPE_SC=='timeless':
        print('CREAZIONE COMANDI PER ESEGUIRE UN CICLO DI GENERAZIONE
          TABULATI TIMELESS\n')
        TYPE2_SC=INPUT_MENU('timeless_cyc_load_gen',0)
        if TYPE2_SC[0]==1:
          print('Limite errori di immissione dati raggiunto')
          CTRL_SC=-1
        else:
          if TYPE2_SC[1]==1:
            SERV_SC=INPUT_MENU('timeless_gen',[DATA_SC[2]-1,0])
            if SERV_SC[0]==0:
              SERV_START=[]
              SERV_OUT_SC.append('timeless_gen\n')
              SERV_OUT_SC.append('%d%s'%(SERV_SC[1], F_DELIMITER))
              SERV_TML_SC=SERV_SC #SERV_TML=[0, N_tml]
              CTRL_SC=0
            else:
              CTRL_SC=-1
              print('Limite errori di immissione dati raggiunto')
            #end if
          elif TYPE2_SC[1]==2:
            SERV_SC=INPUT_MENU('tab_inp',[DATA_SC[2]-1,0])
            if SERV_SC[0]==0:
              SERV_START=[]
              SERV_OUT_SC.append('timeless_load\n')
              SERV_OUT_SC.append('%d%s%d%s\n'%(SERV_SC[1], F_DELIMITER,
                SERV_SC[2], F_DELIMITER))
              SERV_TML_SC=SERV_SC  #SERV_TML=[0, N_tml, prec_dec_open]
              CTRL_SC=0
            else:
              print('Limite errori di immissione dati raggiunto')
              CTRL_SC=-1
            #end if
          else:
            CTRL_SC=-2
          #end if
        #end if
        if not(CTRL_SC==-1 or CTRL_SC==-2):
          SERV_SC=INPUT_MENU('timeless1_cyc',[DATA_SC[2]-SERV_SC[1]])
          if SERV_SC[0]==1:
            CTRL_SC=-1
            print('Limite errori di immissione dati raggiunto')
          else:   #SERV_SC=[0,prec_dec_save, N_cyc]
            if TYPE2_SC[1]==1:
              SERV_SC[2]-=1
              SERV_OUT_SC.append('%d%s\n'%(SERV_SC[1], F_DELIMITER))
            #end if
```

```
            SERV2_SC=INPUT_MENU('timeless2_cyc',[int((DATA_SC[2]-
                SERV_TML_SC[1])/SERV_SC[2])])
            if SERV2_SC[0]==1:
               CTRL_SC=-1
               print('Limite errori di immissione dati raggiunto')
            else:
               CTRL_SC=0
               SERV_SC.append(SERV2_SC[1]) #SERV_SC=[0, prec_dec_save, N_cyc, step]
            #end if
         #end if
      #end if

   elif TYPE_SC=='tabs':
      print('CREAZIONE COMANDI PER ESEGUIRE UN CICLO DI GENERAZIONE DI
                MATRICI DI BASE\n')
      FIELD_SC=INPUT_MENU('tab1_cyc',DATA_SC)
      if FIELD_SC[0]==1:
         print('Limite errori di immissione dati raggiunto')
         CTRL_SC=-1
      else:
         CTRL_SC=0
      #end if
      if CTRL_SC==0:
         TYPE2_SC=INPUT_MENU('tab_cyc_load_gen',0)
         if TYPE2_SC[0]==1:
            print('Limite errori di immissione dati raggiunto')
            CTRL_SC=-1
         else:
            if TYPE2_SC[1]==1:
               SERV_SC=INPUT_MENU('tab_gen',[DATA_SC[2]-1,0])
               if SERV_SC[0]==0:
                  SERV_START=[]
                  if  SERV_SC[3]=='s' or SERV_SC[3]=='S' or SERV_SC[3]=='sec':
                     SERV_SC[3]='sec'
                  elif SERV_SC[3]=='m'or SERV_SC[3]=='M' or SERV_SC[3]=='min':
                     SERV_SC[3]='min'
                  elif SERV_SC[3]=='h' or SERV_SC[3]=='H':
                     SERV_SC[3]='h'
                  #end if
                  SERV_OUT_SC.append('tabs_gen\n')
                  SERV_OUT_SC.append('%d%s%s%s%s%s'%(SERV_SC[1], F_DELIMITER,
                        str(SERV_SC[2]), F_DELIMITER, SERV_SC[3], F_DELIMITER))
                  SERV_TAB_SC=SERV_SC
                  CTRL_SC=0
               else:
                  CTRL_SC=-1
                  print('Limite errori di immissione dati raggiunto')
               #end if
            elif TYPE2_SC[1]==2:
               SERV_SC=INPUT_MENU('tab_inp',[DATA_SC[2]-1,0])
```

```
            if SERV_SC[0]==0:
               SERV_START=[]
               if  SERV_SC[3]=='s' or SERV_SC[3]=='S' or SERV_SC[3]=='sec':
                  SERV_SC[3]='sec'
               elif SERV_SC[3]=='m'or SERV_SC[3]=='M' or SERV_SC[3]=='min':
                  SERV_SC[3]='min'
               elif SERV_SC[3]=='h' or SERV_SC[3]=='H':
                  SERV_SC[3]='h'
               #end if
               SERV_OUT_SC.append('tabs_load\n')
               SERV_OUT_SC.append('%d%s%s%s%s%s%d%s\n'%(SERV_SC[1],
                        F_DELIMITER, str(SERV_SC[2]), F_DELIMITER, SERV_SC[3],
                        F_DELIMITER, SERV_SC[4], F_DELIMITER))
               SERV_TAB_SC=SERV_SC
               CTRL_SC=0
            else:
               print('Limite errori di immissione dati raggiunto')
               CTRL_SC=-1
            #end if
         else:
            CTRL_SC=-2
         #end if
      #end if
   #end if
   if not(CTRL_SC==-1 or CTRL_SC==-2):
      SERV_SC=INPUT_MENU('tab22_cyc',[DATA_SC[2]-SERV_SC[1]])
      if SERV_SC[0]==1:
         CTRL_SC=-1
         print('Limite errori di immissione dati raggiunto')
      else:
         if TYPE2_SC[1]==1:
            SERV_SC[2]-=1
            SERV_OUT_SC.append('%d%s\n'%(SERV_SC[1], F_DELIMITER))
         #end if
         SERV2_SC=INPUT_MENU('tab3_cyc',[int((DATA_SC[2]-
               SERV_TAB_SC[1])/SERV_SC[2])])
         if SERV2_SC[0]==1:
            CTRL_SC=-1
            print('Limite errori di immissione dati raggiunto')
         else:
            CTRL_SC=0
            SERV_SC.append(SERV2_SC[1])
         #end if
      #end if
   #end if
elif TYPE_SC=='buff':
   SERV_SC=INPUT_MENU('buff_cyc',0)
   if SERV_SC[0]==1:
      print('Limite errori di immissione dati raggiunto')
      CTRL_SC=-1
```

```
    else:
      CTRL_SC=0
    #end if
#end if

if not(CTRL_SC==-1 or CTRL_SC==-2):
    if TYPE_SC=='mtx':
      SERV_FOR=[1, SERV_SC[1], SERV_SC[2], SERV_SC[3], SERV_MTX_SC[1]]
      TEXT_SC='mtx_gen\n'
    elif TYPE_SC=='timeless':
      TEXT_SC='timeless_gen\n'
      SERV_FOR=[1, SERV_SC[1], SERV_SC[2], SERV_SC[3], SERV_TML_SC[1]]
    elif TYPE_SC=='tabs':
      TEXT_SC='tabs_gen\n'
      if FIELD_SC[1]==1:
        SERV_FOR=[FIELD_SC[1], SERV_SC[1], SERV_SC[2], SERV_SC[3],
            SERV_TAB_SC[1]]
      elif FIELD_SC[1]==2:
        SERV_FOR=[FIELD_SC[1], SERV_SC[1], SERV_SC[2], SERV_SC[3],
            SERV_TAB_SC[2]]
      #end if
    elif TYPE_SC=='buff':
      SERV_FOR=[1, SERV_SC[1], SERV_SC[2], SERV_SC[3], SERV_SC[4]]
      TEXT_SC='buff_gen\n'
      SERV_OUT_SC.append(TEXT_SC)
      SERV_OUT_SC.append('%d%s%d%s\n'%(SERV_FOR[4], F_DELIMITER,
            SERV_FOR[1], F_DELIMITER))
    #end if
    start=mp.mpf(SERV_FOR[4])
    for i_write in range(SERV_FOR[2]):
      start+=mp.mpf(SERV_FOR[3])
      SERV_OUT_SC.append(TEXT_SC)
      if TYPE_SC=='mtx' or TYPE_SC=='timeless' or TYPE_SC=='buff':
        SERV_OUT_SC.append('%d%s%d%s\n'%
            (start,F_DELIMITER,SERV_FOR[1],F_DELIMITER))
      elif TYPE_SC=='tabs':
        if SERV_FOR[0]==1:
          SERV_OUT_SC.append('%d%s%s%s%s%s%d%s\n'%(start, F_DELIMITER,
            str(SERV_TAB_SC[2]), F_DELIMITER, SERV_TAB_SC[3], F_DELIMITER,
            SERV_FOR[1], F_DELIMITER))
        elif SERV_FOR[0]==2:
          SERV_OUT_SC.append('%d%s%s%s%s%s%d%s\n'%(SERV_TAB_SC[1],
            F_DELIMITER, str(start), F_DELIMITER, SERV_TAB_SC[3], F_DELIMITER,
            SERV_FOR[1], F_DELIMITER))
        #end if
      #end if
    #end for i_write
    CTRL_SC=0
#end if
if CTRL_SC==0:
```

```
      OUT_SC.append(0)
      OUT_SC.append(SERV_OUT_SC)
   else:
      OUT_SC.append(1)
   #end if
   return(OUT_SC)
#end def SCHED_CYCLE_THEO(TYPE_SC)

def SCHED_WRITE_THEO(DATA_SWT):
#Funzione custom che gestisce la selezione delle istruzioni della famiglia "modellizzazione" da
#immettere nel file di schedulazione, controllando la corretta immissione dei dati necessari alla
#futura esecuzione.
#Mantiene in memoria una serie di parametri in modo da garantire la coerenza tra le varie
#istruzioni.
#Viene richiamata da SCHED_WRITE_MAIN.
#Le varie sessioni di input verranno gestite dalla funzione custom INPUT_MENU.
#Può essere richiamata la funzione custom  SCHED_CYCLE_THEO.
   CTRL_SWT=0
   SERV_LINE=[]
   MTX_OPEN_SWT=DATA_SWT[0]
   TIMELESS_OPEN_SWT=DATA_SWT[1]
   TABS_OPEN_SWT=DATA_SWT[2]
   LAST_N_MTX_SWT=DATA_SWT[3]
   LAST_N_TIMELESS_SWT=DATA_SWT[4]
   LAST_N_TAB_SWT=DATA_SWT[5]
   LAST_TUP_TAB_SWT=DATA_SWT[6]
   LAST_METER_TAB_SWT=DATA_SWT[7]
   LAST_USER_PREC_SWT=DATA_SWT[8]

   while CTRL_SWT==0:
      SERV_SWT=INPUT_MENU('scr_opr_theo',0)
      if SERV_SWT[0]==1:
         print('Limite errori in inserimento dati raggiunto')
         CTRL_SWT=-1
      else:
         if SERV_SWT[1]==1:
            SERV_OP_SWT='mtx_load\n'
            SERV_SWT=INPUT_MENU('mtx_inp',[0])
            if SERV_SWT[0]==0:
               MTX_OPEN_SWT=1
               LAST_N_MTX_SWT=SERV_SWT[1]
               ERROR_SWT=0
            else:
               ERROR_SWT=1
            #end if
         elif SERV_SWT[1]==2:
            SERV_OP_SWT='timeless_load\n'
            SERV_SWT=INPUT_MENU('timeless_inp',[0])
            if SERV_SWT[0]==0:
               TIMELESS_OPEN_SWT=1
```

```
      LAST_N_TIMELESS_SWT=SERV_SWT[1]
      ERROR_SWT=0
    else:
      ERROR_SWT=1
    #end if
  elif SERV_SWT[1]==3:
    SERV_OP_SWT='tabs_load\n'
    SERV_SWT=INPUT_MENU('tab_inp',[0])
    if SERV_SWT[0]==0:
      TABS_OPEN_SWT=1
      LAST_N_TAB_SWT=SERV_SWT[1]
      LAST_TUP_TAB_SWT=SERV_SWT[2]
      if SERV_SWT[0]==0:
        TABS_OPEN_SWT=1
        LAST_N_TAB_SWT=SERV_SWT[1]
        if  SERV_SWT[3]=='s' or SERV_SWT[3]=='S' or SERV_SWT[3]=='sec':
          SERV_SWT[3]='sec'
        elif SERV_SWT[3]=='m'or SERV_SWT[3]=='M' or SERV_SWT[3]=='min':
          SERV_SWT[3]='min'
        elif SERV_SWT[3]=='h' or SERV_SWT[3]=='H':
          SERV_SWT[3]='h'
        #end if

      LAST_METER_TAB_SWT=SERV_SWT[3]
      ERROR_SWT=0
    else:
      ERROR_SWT=1
    #end if
  elif SERV_SWT[1]==4:
    SERV_OP_SWT='mtx_gen\n'
    SERV_SWT=INPUT_MENU('mtx_gen',[0])
    if SERV_SWT[0]==0:
      MTX_OPEN_SWT=1
      LAST_N_MTX_SWT=SERV_SWT[1]
      ERROR_SWT=0
    else:
      ERROR_SWT=1
    #end if
  elif SERV_SWT[1]==5:
    if MTX_OPEN_SWT==1:
      SERV_OP_SWT='timeless_gen\n'
      SERV_SWT=INPUT_MENU('timeless_gen',[LAST_N_MTX_SWT])
      if SERV_SWT[0]==0:
        TIMELESS_OPEN_SWT=1
        LAST_N_TIMELESS_SWT=SERV_SWT[1]
        ERROR_SWT=0
      else:
        ERROR_SWT=1
      #end if
    else:
```

```
        print('Linea di comando non inserita')
        input('Incorretto inserire comando generazione tabulati timeless senza aver
          preventivamente caricato in memoria idonee matrici di base')
        ERROR_SWT=2
      #end if
    elif SERV_SWT[1]==6:
      if TIMELESS_OPEN_SWT==1:
        SERV_OP_SWT='tabs_gen\n'
        SERV_SWT=INPUT_MENU('tab_gen',[LAST_N_TIMELESS_SWT])
        if SERV_SWT[0]==0:
          TABS_OPEN_SWT=1
          LAST_N_TAB_SWT=SERV_SWT[1]
          LAST_TUP_TAB_SWT=SERV_SWT[2]
          if  SERV_SWT[3]=='s' or SERV_SWT[3]=='S' or SERV_SWT[3]=='sec':
            SERV_SWT[3]='sec'
          elif SERV_SWT[3]=='m'or SERV_SWT[3]=='M' or SERV_SWT[3]=='min':
            SERV_SWT[3]='min'
          elif SERV_SWT[3]=='h' or SERV_SWT[3]=='H':
            SERV_SWT[3]='h'
          #end if
          LAST_METER_TAB_SWT=SERV_SWT[3]
          ERROR_SWT=0
        else:
          ERROR_SWT=1
        #end if
      else:
        print('Linea di comando non inserita')
        input('Incorretto inserire comando generazione tabulati caso specifico senza aver
          preventivamente caricato in memoria idonei tabulati timeless')
        ERROR_SWT=2
      #end if
    elif SERV_SWT[1]==7:
      if TABS_OPEN_SWT==1:
        SERV_OP_SWT='buff_gen\n'
        SERV_SWT=INPUT_MENU('buff_gen',[LAST_N_TAB_SWT])
        if SERV_SWT[0]==0:
          ERROR_SWT=0
        else:
          ERROR_SWT=1
        #end if
      else:
        print('Linea di comando non inserita')
        input('Incorretto inserire comando generazione tabulati in presenza di buffer senza aver
          preventivamente caricato in memoria i tabulati di un caso idoneo senza buffer')
        ERROR_SWT=2
      #end if
    elif SERV_SWT[1]==8:
      SERV_SWT=SCHED_CYCLE_THEO('mtx',[MTX_OPEN_SWT,
          LAST_N_MTX_SWT])
      if SERV_SWT[0]==0:
```

```
            ERROR_SWT=3
        else:
            ERROR_SWT=1
            print('Linea di comando non inserita')
        #end if
    elif SERV_SWT[1]==9:
        if MTX_OPEN_SWT==1:
            SERV_SWT=SCHED_CYCLE_THEO('timeless',[MTX_OPEN_SWT,
              LAST_N_MTX_SWT])
            if SERV_SWT[0]==0:
                ERROR_SWT=3
            else:
                ERROR_SWT=1
                print('Linea di comando non inserita')
            #end if
        else:
            print('Linea di comando non inserita')
            input('Incorretto inserire comando generazione CICLO TABULATI TIMELESS senza
              aver preventivamente caricato in memoria le matrici di base')
            ERROR_SWT=2
        #end if
    elif SERV_SWT[1]==10:
        if TIMELESS_OPEN_SWT==1:
            SERV_SWT=SCHED_CYCLE_THEO('tabs',[TIMELESS_OPEN_SWT,
              TABS_OPEN_SWT, LAST_N_TIMELESS_SWT, LAST_N_TAB_SWT,
              LAST_TUP_TAB_SWT, LAST_METER_TAB_SWT])
            if SERV_SWT[0]==0:
                ERROR_SWT=3
            else:
                ERROR_SWT=1
            #end if
        else:
            print('Linea di comando non inserita')
            input('Incorretto inserire comando generazione CICLO TABULATI SENZA BUFFER
              senza aver preventivamente caricato in memoria i tabulati timeless')
            ERROR_SWT=2
        #end if
    elif SERV_SWT[1]==11:
        if TABS_OPEN_SWT==1:
            SERV_SWT=SCHED_CYCLE_THEO('buff',[0])
            if SERV_SWT[0]==0:
                ERROR_SWT=3
            else:
                ERROR_SWT=1
            #end if
        else:
            print('Linea di comando non inserita')
            input('Incorretto inserire comando generazione CICLO TABULATI CON BUFFER
              senza aver preventivamente caricato in memoria i tabulati di un caso idoneo senza
              buffer')
```

```
                ERROR_SWT=2
            #end if
        elif SERV_SWT[1]==0:
            ERROR_SWT=0
            CTRL_SWT=-1
        else:
            input('ERRORE ALL INTERNO MENU SELEZIONE COMANDI')
        #end if
        if ERROR_SWT==1:
            print('Limite errori in inserimento dati raggiunto')
            input('Linea di comando non inserita')
        elif ERROR_SWT==0 and not(CTRL_SWT==-1):
            SERV_LINE.append('%s'%(SERV_OP_SWT))
            for i_write in range(len(SERV_SWT)-1):
                SERV_LINE.append(str(SERV_SWT[i_write+1]))
                SERV_LINE.append(F_DELIMITER)
            #end for
            SERV_LINE.append('\n')
        elif ERROR_SWT==3 and not(CTRL_SWT==-1):
            for i_write in range(len(SERV_SWT[1])):
                SERV_LINE.append(SERV_SWT[1][i_write])
            #end for i_write
        #end if
    #end if
  #end while
  DATA_OUT_SWT=[MTX_OPEN_SWT, TABS_OPEN_SWT, LAST_N_MTX_SWT,
      LAST_N_TAB_SWT, LAST_TUP_TAB_SWT, LAST_METER_TAB_SWT,
      LAST_USER_PREC_SWT]
  OUT_SWT=[DATA_OUT_SWT, SERV_LINE]
  return(OUT_SWT)
#end def SCHED_WRITE_THEO

def SCHED_CYCLE_COMP(TYPE_SCC):
#Funzione adibita a generare un ciclo di istruzioni della famiglia "comparazione" in maniera più
#rapida.
#Viene richiamata da SCHED_WRITE_COMP
#Si avvale di INPUT_MENU per l'inserimento e controllo dei dati.
  OUT_SCC=[]
  SERV_OUT_SCC=[]
  if TYPE_SCC=='comp_tab':
    print('CREAZIONE COMANDI PER ESEGUIRE UN CICLO DI GENERAZIONE DI
      COMPARAZIONI IN ASSENZA DI BUFFER\n')
    SERV_SCC=INPUT_MENU('timeless_inp',0)
    if SERV_SCC[0]==1:
      CTRL_SCC=-1
      print('Limite errori di immissione dati raggiunto')
    else:
      CTRL_SCC=0
      SERV2_SCC=INPUT_MENU('comp_tab_cyc',[0])
      if SERV2_SCC[0]==1:
```

```
            CTRL_SCC=-1
            print('Limite errori di immissione dati raggiunto')
        else:
            SERV3_SCC=INPUT_MENU('comp_data_tabs', [0])
            if SERV3_SCC[0]==1:
                CTRL_SCC=-1
                print('Limite errori di immissione dati raggiunto')
            else:
                LIM_TAB=SERV_SCC[1]-((SERV3_SCC[1]-1)*SERV3_SCC[2])
                SERV4_SCC=INPUT_MENU('tab_gen',[LIM_TAB, 0])
                if SERV4_SCC[0]==1:
                    CTRL_SCC=-1
                    print('Limite errori di immissione dati raggiunto')
                else:
                    if SERV4_SWT[3]=='s' or SERV4_SWT[3]=='S' or SERV4_SWT[3]=='sec':
                        SERV4_SWT[3]='sec'
                    elif SERV4_SWT[3]=='m'or SERV4_SWT[3]=='M' or SERV4_SWT[3]=='min':
                        SERV4_SWT[3]='min'
                    elif SERV4_SWT[3]=='h' or SERV4_SWT[3]=='H':
                        SERV4_SWT[3]='h'
                    #end  if
                    SERV5_SCC=INPUT_MENU('sim_data_m',0)
                    if SERV5_SCC[0]==1:
                        CTRL_SCC=-1
                        print('Limite errori di immissione dati raggiunto')
                    else:
                        CTRL_SCC=0
                    #end if
                #end if
            #end if
        #end if
    #end if
    if not(CTRL_SCC==-1):
        T_CYC_SCC=SERV4_SCC[2]
        for i_cyc in range(SERV2_SCC[1]):
            T_CYC_SCC+=i_cyc*SERV2_SCC[2]
            SERV_OUT_SCC.append('comp_tab\n')
            SERV_OUT_SCC.append('%d%s%d%s'%(SERV_SCC[1], F_DELIMITER,
                SERV_SCC[2], F_DELIMITER))
            SERV_OUT_SCC.append('%d%s%d%s'%(SERV3_SCC[1], F_DELIMITER,
                SERV3_SCC[2], F_DELIMITER))
            SERV_OUT_SCC.append('%d%s%s%s%s%s%d%s'%(SERV4_SCC[1], F_DELIMITER,
                str(T_CYC_SCC), F_DELIMITER, SERV4_SCC[3], F_DELIMITER,
                SERV2_SCC[3], F_DELIMITER))
            SERV_OUT_SCC.append('%d%s'%(SERV5_SCC[1], F_DELIMITER))
            SERV_OUT_SCC.append('\n')
        #end for i_cyc
        OUT_SCC=[0, SERV_OUT_SCC]
    else:
        OUT_SCC=[1]
```

```
    #end if

elif TYPE_SCC=='comp_buff':
    print('CREAZIONE COMANDI PER ESEGUIRE UN CICLO DI GENERAZIONE DI
      COMPARAZIONI IN PRESENZA DI BUFFER\n')
    SERV_SCC=INPUT_MENU('timeless_inp',[0])
    if SERV_SCC[0]==1:
      CTRL_SCC=-1
      print('Limite errori di immissione dati raggiunto')
    else:
      CTRL_SCC=0
      SERV2_SCC=INPUT_MENU('comp_buff_cyc',[0])
      if SERV2_SCC[0]==1:
        CTRL_SCC=-1
        print('Limite errori di immissione dati raggiunto')
      else:
        SERV3_SCC=INPUT_MENU('tab_gen',[SERV_SCC[1],0])
        if SERV3_SCC[0]==1:
          CTRL_SCC=-1
          print('Limite errori di immissione dati raggiunto')
        else:
          if SERV3_SWT[3]=='s' or SERV3_SWT[3]=='S' or SERV3_SWT[3]=='sec':
            SERV3_SWT[3]='sec'
          elif SERV3_SWT[3]=='m'or SERV3_SWT[3]=='M' or SERV3_SWT[3]=='min':
            SERV3_SWT[3]='min'
          elif SERV3_SWT[3]=='h' or SERV3_SWT[3]=='H':
            SERV3_SWT[3]='h'
          #end if
          SERV4_SCC=INPUT_MENU('comp_data_buff1', [SERV3_SCC[1]-4])
          if SERV4_SCC[0]==1:
            CTRL_SCC=-1
            print('Limite errori di immissione dati raggiunto')
          else:
            SERV5_SCC=INPUT_MENU('comp_data_buff2', [(SERV3_SCC[1]-4)/
                  (SERV4_SCC[1]-1)])
            if SERV5_SCC[0]==1:
              CTRL_SCC=-1
              print('Limite errori di immissione dati raggiunto')
            else:
              SERV6_SCC=INPUT_MENU('buff_gen',[(N_T_M_CBV[0]-4)-
                  ((SERV4_SCC[1]-1)*SERV5_SCC[1]),0])
              if SERV6_SCC[0]==1:
                CTRL_SCC=-1
                print('Limite errori di immissione dati raggiunto')
              else:
                SERV7_SCC=INPUT_MENU('sim_data_m',0)
                if SERV7_SCC[0]==1:
                  CTRL_SCC=-1
                  print('Limite errori di immissione dati raggiunto')
                else:
```

```
                        CTRL_SCC=0
                    #end if
                #end if
            #end if
        #end if
    #end if
    #end if
    if not(CTRL_SCC==-1):
        T_CYC_SCC=SERV3_SCC[2]
        SERV_OUT_SCC.append('timeless_load\n')
        SERV_OUT_SCC.append('%d%s%d%s\n'%(SERV_SCC[1], F_DELIMITER,
            SERV_SCC[2], F_DELIMITER))
        for i_cyc in range(SERV2_SCC[1]):
            T_CYC_SCC+=i_cyc*SERV2_SCC[2]
            SERV_OUT_SCC.append('tabs_gen\n')
            SERV_OUT_SCC.append('%d%s%s%s%s%s\n'%(SERV3_SCC[1], F_DELIMITER,
                str(T_CYC_SCC), F_DELIMITER, SERV3_SCC[3], F_DELIMITER,
                SERV2_SCC[3], F_DELIMITER))
            SERV_OUT_SCC.append('comp_buff\n')
            SERV_OUT_SCC.append('%d%s%s%s%s%s'%(SERV3_SCC[1], F_DELIMITER,
                str(SERV3_SCC[2]), F_DELIMITER, SERV3_SCC[3], F_DELIMITER,
                F_PREC_PRG, F_DELIMITER))
            SERV_OUT_SCC.append('%d%s%d%s'%(SERV4_SCC[1], F_DELIMITER,
                SERV5_SCC[1], F_DELIMITER))
            SERV_OUT_SCC.append('%d%s%d%s'%(SERV6_SCC[1], F_DELIMITER,
                SERV2_SCC[3], F_DELIMITER))
            SERV_OUT_SCC.append('%d%s'%(SERV7_SCC[1], F_DELIMITER))
            SERV_OUT_SCC.append('\n')
        #end for i_cyc
        OUT_SCC=[0, SERV_OUT_SCC]
    else:
        OUT_SCC=[1]
    #end if
  return(OUT_SCC)
#end def SCHED_CYCLE_COMP(TYPE_SC)

def SCHED_WRITE_COMP(DATA_SWC):
#Funzione custom che gestisce la selezione delle istruzioni della famiglia "comparazione" da
#immettere nel file di schedulazione, controllando la corretta immissione dei dati necessari alla
#futura esecuzione.
#Mantiene in memoria una serie di parametri in modo che un'istruzione inserita successivamente
#non possa essere in contrasto con una precedente.
#Viene richiamata da SCHED_WRITE_MAIN.
#Le varie sessioni di input verranno gestite dalla funzione custom INPUT_MENU.
#Può essere richiamata la funzione custom  SCHED_CYCLE_COMP.
  CTRL_SWC=0
  SERV_LINE=[]
  MTX_OPEN_SWC=DATA_SWC[0]
  TABS_OPEN_SWC=DATA_SWC[1]
```

```
LAST_N_MTX_SWC=DATA_SWC[2]
LAST_N_TAB_SWC=DATA_SWC[3]
LAST_TUP_TAB_SWC=DATA_SWC[4]
LAST_METER_TAB_SWC=DATA_SWC[5]
LAST_USER_PREC_SWC=DATA_SWC[6]
LAST_ROUNDS_SWC=DATA_SWC[7]
LAST_STEP_SWC=DATA_SWC[8]

while CTRL_SWC==0:
   SERV_SWC=INPUT_MENU('scr_opr_comp',[0])
   if SERV_SWC[0]==1:
      print('Limite errori in inserimento dati raggiunto')
      CTRL_SWC=-1
   else:
      if SERV_SWC[1]==1:
         SERV_OP_SWC='comp_tab\n'
         SERV_SWC=INPUT_MENU('timeless_inp',[0])
         if SERV_SWC[0]==0:
            ERROR_SWC=0
         else:
            print('Limite errori in inserimento dati raggiunto')
            ERROR_SWC=1
         #end if
         if ERROR_SWC==0:
            SERV2_SWC=INPUT_MENU('comp_data_tabs',[0])
            if SERV2_SWC[0]==1:
               print('Limite errori in inserimento dati raggiunto')
               ERROR_SWC=1
            else:
               SERV_SWC.append(SERV2_SWC[1])
               SERV_SWC.append(SERV2_SWC[2])
               ERROR_SWC=0
            #end if
         #end if
         if ERROR_SWC==0:
            print('INSERIMENTO DATI TABULATO INIZIALE')
            SERV2_SWC=INPUT_MENU('tab_gen',[SERV_SWC[1]-((SERV_SWC[3]-1)
              *SERV_SWC[4])])
            if SERV2_SWC[0]==1:
               print('Limite errori in inserimento dati raggiunto')
               ERROR_SWC=1
            else:
               SERV_SWC.append(SERV2_SWC[1])
               SERV_SWC.append(SERV2_SWC[2])
               SERV_SWC.append(SERV2_SWC[3])
               SERV_SWC.append(SERV2_SWC[4])
               ERROR_SWC=0
            #end if
         #end if
         if ERROR_SWC==0:
```

```
        SERV2_SWC=INPUT_MENU('sim_data_m',[0])
        if SERV2_SWC[0]==1:
            print('Limite errori in inserimento dati raggiunto')
            ERROR_SWC=1
        else:
            SERV_SWC.append(SERV2_SWC[1])
            ERROR_SWC=0
        #end if
    #end if

elif SERV_SWC[1]==2:
    SERV_OP_SWC='comp_buff\n'
    SERV_SWC=INPUT_MENU('tab_inp',[0])
    if SERV_SWC[0]==0:
        if  SERV_SWC[3]=='s' or SERV_SWC[3]=='S' or SERV_SWC[3]=='sec':
            SERV_SWC[3]='sec'
        elif SERV_SWC[3]=='m'or SERV_SWC[3]=='M' or SERV_SWC[3]=='min':
            SERV_SWC[3]='min'
        elif SERV_SWC[3]=='h' or SERV_SWC[3]=='H':
            SERV_SWC[3]='h'
        #end if
        ERROR_SWC=0
    else:
        ERROR_SWC=1
    #end if
    if ERROR_SWC==0:
        print('INSERIMENTO BUFFER INIZIALE E PRECISIONE DESIDERATA')
        SERV2_SWC=INPUT_MENU('comp_data_buff1',[SERV_SWC[1]-4])
        if SERV2_SWC[0]==1:
            print('Limite errori in inserimento dati raggiunto')
            ERROR_SWC=1
        else:
            SERV3_SWC=INPUT_MENU('comp_data_buff2',[int((SERV_SWC[1]-4)/
                    (SERV2_SWC[1]-1))])
            if SERV3_SWC[0]==1:
                print('Limite errori in inserimento dati raggiunto')
                ERROR_SWC=1
            else:
                SERV_SWC.append(SERV2_SWC[1])
                SERV_SWC.append(SERV3_SWC[1])
                ERROR_SWC=0
            #end if
        #end if
    #end if
    if ERROR_SWC==0:
        SERV4_SWC=INPUT_MENU('buff_gen',[SERV_SWC[1]-((SERV2_SWC[1]-1)
          *SERV3_SWC[1])])
        if SERV4_SWC[0]==1:
            print('Limite errori in inserimento dati raggiunto')
            ERROR_SWC=1
```

```
        else:
            SERV_SWC.append(SERV4_SWC[1])
            SERV_SWC.append(SERV4_SWC[2])
            ERROR_SWC=0
        #end if
    #end if
    if ERROR_SWC==0:
        SERV5_SWC=INPUT_MENU('sim_data_m',[0])
        if SERV5_SWC[0]==1:
            print('Limite errori in inserimento dati raggiunto')
            ERROR_SWC=1
        else:
            SERV_SWC.append(SERV5_SWC[1])
            ERROR_SWC=0
        #end if
    #end if
elif SERV_SWC[1]==3:
    SERV_SWC=SCHED_CYCLE_COMP('comp_tab')
    if SERV_SWC[0]==1:
        ERROR_SWC=1
    else:
        ERROR_SWC=3
    #end if
elif SERV_SWC[1]==4:
    SERV_SWC=SCHED_CYCLE_COMP('comp_buff')
    if SERV_SWC[0]==1:
        ERROR_SWC=1
    else:
        ERROR_SWC=3
    #end if
elif SERV_SWC[1]==0:
    ERROR_SWC=0
    CTRL_SWC=-1
else:
    input('ERRORE ALL INTERNO MENU SELEZIONE COMANDI')
#end if
if not(ERROR_SWC==1) and not(CTRL_SWC==-1):
    if SERV_OP_SWC=='comp_tab\n':
        MTX_OPEN_SWC=1
        LAST_N_MTX_SWC=SERV_SWC[1]
        LAST_ROUNDS_SWC=SERV_SWC[2]
        LAST_STEP_SWC=SERV_SWC[3]
    elif SERV_OP_SWC=='comp_buff\n':
        TABS_OPEN_SWC=1
        LAST_N_TAB_SWC=SERV_SWC[1]
        LAST_TUP_TAB_SWC=SERV_SWC[2]
        LAST_METER_TAB_SWC=SERV_SWC[3]
        LAST_ROUNDS_SWC=SERV_SWC[5]
        LAST_STEP_SWC=SERV_SWC[6]
    #end if
```

```
            #end if

            if ERROR_SWC==1:
               print('Limite errori in inserimento dati raggiunto')
               input('Linea di comando non inserita')
            elif ERROR_SWC==0 and not(CTRL_SWC==-1):
               SERV_LINE.append('%s'%(SERV_OP_SWC))
               for i_write in range(len(SERV_SWC)-1):
                  SERV_LINE.append('%s'%(str(SERV_SWC[i_write+1])))
                  SERV_LINE.append('%s'%(F_DELIMITER))
               #end for i_write
               SERV_LINE.append('\n')
            elif ERROR_SWC==3 and not(CTRL_SWC==-1):
               for i_write in range(len(SERV_SWC[1])):
                  SERV_LINE.append(SERV_SWC[1][i_write])
               #end for i_write
            #end if
         #end if
      #end while
      DATA_OUT_SWC=[MTX_OPEN_SWC, TABS_OPEN_SWC, LAST_N_MTX_SWC,
            LAST_N_TAB_SWC, LAST_TUP_TAB_SWC, LAST_METER_TAB_SWC,
            LAST_USER_PREC_SWC, LAST_ROUNDS_SWC, LAST_STEP_SWC]
      OUT_SWC=[DATA_OUT_SWC, SERV_LINE]
      return(OUT_SWC)
   #end SCHED_WRITE_COMP()

   def SCHED_WRITE_MAIN():
   #Funzione custom adibita all'apertura del file di schedulazione e alla gestione del menù di scelta
   #principale riguardo alla famiglia dell'operazione che si intende inserirvi.
   #Viene richiamata da SCHED_CREATE_MAIN.
   #Il nome del file e le scelte del menù verranno immesso tramite la funzione custom
   #INPUT_MENU.
   #In corso di elaborazione potranno essere richiamate le seguenti funzioni custom:
   #SCHED_WRITE_THEO, SCHED_WRITE_SIM e SCHED_WRITE_COMP.
      SERV_SWM=INPUT_MENU('sched_name_save',[0])
      if SERV_SWM[0]==1:
         print('Limite errori in immissione dati raggiunto')
         CTRL_SWM=-1
      else:
         CTRL_SWM=0
         f=open("%s%s%s.%s" %(F_BASE_NAME[3][0],SERV_SWM[1],F_BASE_NAME[3][1],
            F_EXT_S),"w+")

         MTX_OPEN_SWM=0
         TIMELESS_OPEN_SWM=0
         TABS_OPEN_SWM=0
         LAST_N_MTX_SWM=0
         LAST_N_TIMELESS_SWM=0
         LAST_N_TAB_SWM=0
         LAST_TUP_TAB_SWM=0
```

```
LAST_METER_TAB_SWM=0
LAST_USER_PREC_SWM=0
COMP_MTX_OPEN_SWM=0
COMP_TIMELESS_OPEN_SWM=0
COMP_TABS_OPEN_SWM=0
COMP_LAST_N_MTX_SWM=0
COMP_LAST_N_TIMELESS_MTX_SWM=0
COMP_LAST_N_TAB_SWM=0
COMP_LAST_TUP_TAB_SWM=0
COMP_LAST_METER_TAB_SWM=0
COMP_LAST_USER_PREC_SWM=0
COMP_LAST_ROUNDS_SWC=0
COMP_LAST_STEP_SWC=0
DATA_SWM=[MTX_OPEN_SWM, TIMELESS_OPEN_SWM, TABS_OPEN_SWM,
   LAST_N_MTX_SWM, LAST_N_TIMELESS_SWM, LAST_N_TAB_SWM,
   LAST_TUP_TAB_SWM, LAST_METER_TAB_SWM, LAST_USER_PREC_SWM]
COMP_DATA_SWM=[COMP_MTX_OPEN_SWM, COMP_TIMELESS_OPEN_SWM,
   COMP_TABS_OPEN_SWM, COMP_LAST_N_MTX_SWM,
   COMP_LAST_N_TIMELESS_MTX_SWM, COMP_LAST_N_TAB_SWM,
   COMP_LAST_TUP_TAB_SWM, COMP_LAST_METER_TAB_SWM,
   COMP_LAST_USER_PREC_SWM, COMP_LAST_ROUNDS_SWC,
   COMP_LAST_STEP_SWC]
CTRL_SWM=0
while CTRL_SWM==0:
   SERV_SWM=INPUT_MENU('scr_opr_family',[0])
   if SERV_SWM[0]==1:
      print('Limite errori in inserimento dati raggiunto')
      CTRL_SWM=-1
   else:
      if SERV_SWM[1]==1:
         SERV_SWM=SCHED_WRITE_THEO(DATA_SWM)
         DATA_SWM=SERV_SWM[0]
         for i in range(len(SERV_SWM[1])):
            f.write(SERV_SWM[1][i])
         #end for i
      elif SERV_SWM[1]==2:
         SERV_SWM=SCHED_WRITE_SIM()
         for i in range(len(SERV_SWM)):
            f.write(SERV_SWM[i])
         #end for i
      elif SERV_SWM[1]==3:
         SERV_SWM=SCHED_WRITE_COMP(COMP_DATA_SWM)
         COMP_DATA_SWM=SERV_SWM[0]
         for i in range(len(SERV_SWM[1])):
            f.write(SERV_SWM[1][i])
         #end for i
      elif SERV_SWM[1]==0:
         CTRL_SWM=-1
         f.write('end\n')
      #end if
```

```
            #end if
        #end while
        f.close()
        return()
#end def SCHED_WRITE_MAIN()

def SCHED_CREATE_MAIN():
#Funzione custom adibita alla gestione del menù delle opzioni possibili nella creazione e modifica
#di un file di schedulazione.
#Viene richiamata dalla funzione SCHEDULER_MAIN.
#L'input viene gestito tramite la funzione custom INPUT_MENU.
#L'unica opzione attualmente implementata richiamerà la funzione SCHED_WRITE_MAIN.
    CTRL_SC=0
    print('\n')
    print('PROCEDURA CREAZIONE FILE DI SCHEDULAZIONE')
    while CTRL_SC==0:
        SERV_SC=INPUT_MENU('scr_main',[0])
        if SERV_SC[0]==1:
            print('Limite errori in inserimento dati raggiunto')
            CTRL_SC=-1
        else:
            if SERV_SC[1]==1:
                SERV_SC=SCHED_WRITE_MAIN()
            elif SERV_SC[1]==2:
                input('Procedura non implementata')
            elif SERV_SC[1]==0:
                CTRL_SC=-1
            #end if
        #end if
    #end while
    input('Reindirizzamento a menù precedente')
    return()
#end def SCHED_CREATE_MAIN


def SCHED_DATA_DECODER(LINE_SDD,COD_SDD):
#Funzione custom adibita alla decodifica dei dati presenti nei file di schedulazione con successiva
#riformulazione in formato standard per il programma.
#Viene richiamata dalla funzione custom SCHED_EXEC.
#Restituisce i dati rielaborati.
    OUT_SDD=[]
    DEL_SDD=F_DELIMITER
    ERROR_SDD=0
    if COD_SDD=='mtx':
        field_number=2
    elif COD_SDD=='timeless':
        field_number=2
    elif COD_SDD=='tabs':
        field_number=4
    elif COD_SDD=='buff':
```

```
    field_number=2
elif COD_SDD=='single_nb':
    field_number=5
elif COD_SDD=='single_buff':
    field_number=6
elif COD_SDD=='multi_nb':
    field_number=6
elif COD_SDD=='multi_buff':
    field_number=7
elif COD_SDD=='comp_tab':
    field_number=9
elif COD_SDD=='comp_buff':
    field_number=9
else:
    input('PROBLEMA AL SELETTORE INIZIALE IN DATA_DECODER')
    ERROR_SDD=1
    OUT_SDD.append(1)
#end if
if ERROR_SDD==0:
    OUT_SDD.append(0)
    pointer=0
    field_count=0
    start_pointer=0
    SERV_SDD=[]
    while pointer<(len(LINE_SDD)-1) and field_count<field_number and ERROR_SDD==0:
        if not(LINE_SDD[pointer]==DEL_SDD):
            pointer+=1
        else:
            field_count+=1
            try:
                SERV_SDD.append(LINE_SDD[start_pointer:pointer])
                start_pointer=pointer+1
                pointer+=1
            except:
                ERROR_SDD=1
                print('ERRORE ESTRAZIONE DATI DENTRO SCHED_DATA_DECODER 1')
            #end try
        #end if
    #end while
    if ERROR_SDD==1:
        OUT_SDD[0]=1
    else:
        if COD_SDD=='mtx' or COD_SDD=='buff' or COD_SDD=='timeless':
            for i in range(field_number):
                try:
                    OUT_SDD.append(int(SERV_SDD[i]))
                except:
                    print('ERRORE ESTRAZIONE DATI DENTRO SCHED_DATA_DECODER 2')
                    ERROR_SDD=1
                    OUT_SDD[0]=1
```

```
        #end try
    #end for i
    if COD_SDD=='timeless':
        OUT_SDD.append(0)
    #end if
elif COD_SDD=='tabs':
    try:
        SERV_SDD[0]=int(SERV_SDD[0])
        SERV_SDD[1]=mp.mpf(SERV_SDD[1])
        if SERV_SDD[1]-int(SERV_SDD[1])==0:
            SERV_SDD[1]='%d.0'%(SERV_SDD[1])
        #end if
        SERV_SDD[3]=int(SERV_SDD[3])
        if not(SERV_SDD[2] in ['sec','min','h']):
            ERROR_SDD=1
            OUT_SDD[0]=1
            print('ERRORE ESTRAZIONE DATI DENTRO SCHED_DATA_DECODER 3')
        #end if
    except:
        ERROR_SDD=1
        OUT_SDD[0]=1
        print('ERRORE ESTRAZIONE DATI DENTRO SCHED_DATA_DECODER 4')
    #end try
elif COD_SDD=='comp_tab':
    try:
        OUT2_SDD=[]
        for i in range(2):
            OUT2_SDD.append(int(SERV_SDD[i]))
        #end for i
        OUT2_SDD.append(0)
        for i in range(3):
            OUT2_SDD.append(int(SERV_SDD[i+2]))
        #end for i
        OUT2_SDD.append(mp.mpf(SERV_SDD[5]))
        if SERV_SDD[6] in ['s','m','h','sec','min','h','S','M','H']:
            if SERV_SDD[6] in ['s','S','sec']:
                SERV_SDD[6]='sec'
            elif SERV_SDD[6] in ['m','M','min']:
                SERV_SDD[6]='min'
            elif SERV_SDD[6] in ['h','H']:
                SERV_SDD[6]='h'
            #end if
            OUT2_SDD.append(SERV_SDD[6])
        else:
            ERROR_SDD=1
            OUT2_SDD[0]=1
            print('ERRORE ESTRAZIONE DATI DENTRO SCHED_DATA_DECODER
                  5')
        #end if
        OUT2_SDD.append(int(SERV_SDD[7]))
```

```
            OUT2_SDD.append(int(SERV_SDD[8]))
        except:
            ERROR_SDD=1
            OUT_SDD[0]=1
            print('ERRORE ESTRAZIONE DATI DENTRO SCHED_DATA_DECODER 6')
        #end try
        if not(ERROR_SDD==1):
            SERV_SDD=[]
            SERV_SDD.append('com_tab_nb')
            SERV_SDD.append([OUT2_SDD[0],OUT2_SDD[1], OUT2_SDD[2]])
            SERV_SDD.append([OUT2_SDD[5],OUT2_SDD[6],OUT2_SDD[7],OUT2_SDD[8]])
            SERV_SDD.append(OUT2_SDD[9])
            SERV_SDD.append([OUT2_SDD[3],OUT2_SDD[4]])
        #end if

    elif COD_SDD=='comp_buff':
        try:
            OUT2_SDD=[]
            OUT2_SDD.append(int(SERV_SDD[0]))
            SERV_SDD[1]=mp.mpf(SERV_SDD[1])
            if SERV_SDD[1]-int(SERV_SDD[1])==0:
                SERV_SDD[1]='%d.0'%(SERV_SDD[1])
            #end if
            OUT2_SDD.append(SERV_SDD[1])
            if SERV_SDD[2] in ['s','m','h','sec','min','h','S','M','H']:
                if SERV_SDD[2] in ['s','S','sec']:
                    SERV_SDD[2]='sec'
                elif SERV_SDD[2] in ['m','M','min']:
                    SERV_SDD[2]='min'
                elif SERV_SDD[2] in ['h','H']:
                    SERV_SDD[2]='h'
                #end if
                OUT2_SDD.append(SERV_SDD[2])
            else:
                ERROR_SDD=1
                OUT_SDD[0]=1
                print('ERRORE ESTRAZIONE DATI DENTRO SCHED_DATA_DECODER 7')
            #end if
            OUT2_SDD.append(int(SERV_SDD[3]))
            OUT2_SDD.append(int(SERV_SDD[4]))
            OUT2_SDD.append(int(SERV_SDD[5]))
            OUT2_SDD.append(int(SERV_SDD[6]))
            OUT2_SDD.append(int(SERV_SDD[7]))
            OUT2_SDD.append(int(SERV_SDD[8]))
            ERROR_SDD=0
        except:
            ERROR_SDD=1
            OUT_SDD[0]=1
            print('ERRORE ESTRAZIONE DATI DENTRO SCHED_DATA_DECODER 8')
        #end try
```

```
        if not(ERROR_SDD==1):
            SERV_SDD=[]
            SERV_SDD.append('com_tab_bfvar')
            SERV_SDD.append([OUT2_SDD[0],OUT2_SDD[1],OUT2_SDD[2],OUT2_SDD[3]])
            SERV_SDD.append([OUT2_SDD[6],OUT2_SDD[7]])
            SERV_SDD.append(OUT2_SDD[8])
            SERV_SDD.append([OUT2_SDD[4],OUT2_SDD[5]])
        #end if
    else:
        try:
            SERV_SDD[0]=int(SERV_SDD[0])
            SERV_SDD[1]=mp.mpf(SERV_SDD[1])
            if SERV_SDD[1]-int(SERV_SDD[1])==0:
                SERV_SDD[1]='%d.0'%(int(SERV_SDD[1]))
            #end if
            if SERV_SDD[2] in ['s','m','h','sec','min','h','S','M','H']:
                if SERV_SDD[2] in ['s','S','sec']:
                    SERV_SDD[2]='sec'
                elif SERV_SDD[2] in ['m','M','min']:
                    SERV_SDD[2]='min'
                elif SERV_SDD[2] in ['h','H']:
                    SERV_SDD[2]='h'
                #end if
            else:
                ERROR_SDD=1
                OUT_SDD[0]=1
                print('ERRORE ESTRAZIONE DATI DENTRO SCHED_DATA_DECODER 9')
            #end if
            SERV_SDD[3]=int(SERV_SDD[3])
            SERV_SDD[4]=int(SERV_SDD[4])
            if not(COD_SDD=='single_nb'):
                SERV_SDD[5]=int(SERV_SDD[5])
                if COD_SDD=='multi_buff':
                    SERV_SDD[6]=int(SERV_SDD[6])
                #end if
            #end if
        except:
            ERROR_SDD=1
            OUT_SDD[0]=1
            print('ERRORE ESTRAZIONE DATI DENTRO SCHED_DATA_DECODER 10')
        #end try
    #end if
#end if
if ERROR_SDD==1:
    OUT_SDD=[]
    OUT_SDD.append(1)
else:
    OUT_SDD=[]
    OUT_SDD.append(0)
    OUT_SDD.append(SERV_SDD)
```

```
      #end if
   #end if
   return(OUT_SDD)
#end def SCHED_DATA_DECODER(LINE_SDR,COD_SDR)

def SCHED_EXEC():
#Funzione custom adibita all'esecuzione di un file di schedulazione preesistente.
#Mantiene in memoria per tutta l'esecuzione i dati che necessitano essere trasportati da una
#funzione alle successive.
#L'immissione del nome del file da caricare avverrà tramite la funzione custom INPUT_MENU.
#L'apertura sarà eseguita tramite GENERAL_MATRIX_LOADING.
#La trasformazione delle linee del file di schedulazione in parametri operativi sarà eseguita da
#SCHED_DATA_DECODER.
#In base alle istruzioni in esso contenuto potranno essere richiamate le seguenti funzioni custom:
#GENERAL_MATRIX_LOADING, GENERAL_MATRIX_GENERATION, TIMELESS,
#NEW_SET_ELABORATION, BUFFER_ELABORATION, SINGLE_SIM, MULTI_SIM e
#USER_COMPARATORE_MAIN
   STATUS_MTX_SCHED=0
   STATUS_TIMELESS_SCHED=0
   STATUS_TABS_SCHED=0
   SERV_SCHED=INPUT_MENU('sched_name_load',[0])
   if SERV_SCHED[0]==1:
      CTRL_SCHED=-1
      input('Limite errori di immissione dati raggiunto')
   else:
      CTRL_SCHED=0
      SERV_SCHED=GENERAL_MATRIX_LOADING('sched',[SERV_SCHED[1]])
      if SERV_SCHED[0]==-1:
         CTRL_SCHED=-1
      else:
         CTRL_SCHED=0
      #end if
   #end if
   if CTRL_SCHED==0:
      SCHED=SERV_SCHED[1]
      line=0
      count_empty=0
      while CTRL_SCHED==0 and line<len(SCHED) and count_empty<5:
         if SCHED[line]=='mtx_load\n':
            count_empty=0
            line+=1
            SERV_SCHED=SCHED_DATA_DECODER(SCHED[line],'mtx')
            ERROR_SCHED=SERV_SCHED[0]
            if ERROR_SCHED==1:
               print('PROGRAMMA DI SCHEDULAZIONE ERRATO 1')
               input('DATI APERTURA MATRICE BASE ERRATI, %s DOPO %s'%
                  (SCHED[line],SCHED[line-1]))
               CTRL_SCHED=-1
            else:
               DATA_SCHED=SERV_SCHED[1]
```

```
        SERV_SCHED=GENERAL_MATRIX_LOADING('sched',DATA_SCHED)
        if not(SERV_SCHED[0]==1):
            CTRL_SCHED=-1
            print('PROGRAMMA DI SCHEDULAZIONE ERRATO 2')
            input('errore in apertura matrici di base')
        else:
            CTRL_SCHED=0
            STATUS_MTX_SCHED=0
            DELTA_Z_SCHED=SERV_SCHED[1]
            CINT_Z_SCHED=SERV_SCHED[2]
            CF_Z_SCHED=SERV_SCHED[3]
            CINT_TA_SCHED=SERV_SCHED[4]
            CF_TA_SCHED=SERV_SCHED[5]
            N_MTX_SCHED=DELTA_Z_SCHED.cols
            CF_TU_SCHED=SERV_SCHED[6]
            CINT_TA_GLBL_SCHED=SERV_SCHED[7]
            CDEN_TA_GLBL_SCHED=SERV_SCHED[8]
            CINT_TW_GLBL_SCHED=SERV_SCHED[9]
            CDEN_TW_GLBL_SCHED=SERV_SCHED[10]
            CDEN_TU_GLBL_SCHED=SERV_SCHED[11]
            DATA_MTX=[DELTA_Z_SCHED, CINT_Z_SCHED, CF_Z_SCHED,
                    CINT_TA_SCHED, CF_TA_SCHED, N_MTX_SCHED, CF_TU_SCHED,
                    CINT_TA_GLBL_SCHED, CDEN_TA_GLBL_SCHED,
                    CINT_TW_GLBL_SCHED, CDEN_TW_GLBL_SCHED,
                    CDEN_TU_GLBL_SCHED]
        #end if
    #end if
elif SCHED[line]=='timeless_load\n':
    count_empty=0
    line+=1
    SERV_SCHED=SCHED_DATA_DECODER(SCHED[line],'timeless')
    ERROR_SCHED=SERV_SCHED[0]
    if ERROR_SCHED==1:
        print('PROGRAMMA DI SCHEDULAZIONE ERRATO 3')
        input('DATI APERTURA TIMELESS ERRATI, %s DOPO %s'%
          (SCHED[line],SCHED[line-1]))
        CTRL_SCHED=-1
    else:
        DATA_SCHED=SERV_SCHED[1]
        DATA_SCHED.append(0)
        SERV_SCHED=GENERAL_MATRIX_LOADING('sched',DATA_SCHED)
        if not(SERV_SCHED[0]==1):
            CTRL_SCHED=-1
            print('PROGRAMMA DI SCHEDULAZIONE ERRATO 4')
            input('errore in apertura timeless')
        else:
            CTRL_SCHED=0
            TIME_VAR_TIMELESS_SCHED=SERV_SCHED[1]
            TIME_VAR_TIMELESS_MED_SCHED=SERV_SCHED[2]
            WIP_NO_BUFF_TIMELESS_SCHED=SERV_SCHED[3]
```

```
            GLOBAL_TIMELESS_SCHED=SERV_SCHED[4]
            NOBUFF_TIMELESS_PARAMETERS_SCHED=SERV_SCHED[5]
            N_TABS_TIMELESS_SCHED=TIME_VAR_TIMELESS_MED_SCHED.cols
            DATA_TIMELESS=[N_TABS_TIMELESS_SCHED,
                    TIME_VAR_TIMELESS_SCHED,
                    TIME_VAR_TIMELESS_MED_SCHED,
                    WIP_NO_BUFF_TIMELESS_SCHED, GLOBAL_TIMELESS_SCHED,
                    NOBUFF_TIMELESS_PARAMETERS_SCHED]
        #end if
    #end if

elif SCHED[line]=='tabs_load\n':
    count_empty=0
    line+=1
    SERV_SCHED=SCHED_DATA_DECODER(SCHED[line],'tabs')
    ERROR_SCHED=SERV_SCHED[0]
    if ERROR_SCHED==1:
        print('PROGRAMMA DI SCHEDULAZIONE ERRATO 5')
        input('DATI APERTURA TABULATI CASO SPECIFICO ERRATI, %s DOPO %s'%
          (SCHED[line],SCHED[line-1]))
        CTRL_SCHED=-1
    else:
        DATA_SCHED=SERV_SCHED[1]
        SERV_SCHED=GENERAL_MATRIX_LOADING('sched',DATA_SCHED)
        if not(SERV_SCHED[0]==1):
            CTRL_SCHED=-1
            print('PROGRAMMA DI SCHEDULAZIONE ERRATO 6')
            input('errore in apertura tabulati caso specifico')
        else:
            CTRL_SCHED=0
            STATUS_TABS_SCHED=0
            N_TABS_SCHED=int(SERV_SCHED[1][0])
            T_UPSTREAM_SCHED=mp.mpf(SERV_SCHED[1][1])
            METER_SCHED=SERV_SCHED[1][2]
            TIME_VAR_SCHED=SERV_SCHED[2]
            TIME_VAR_ME_SCHED=SERV_SCHED[3]

            WIP_NO_BUFF_SCHED=SERV_SCHED[4]
            GLOBAL_SCHED=SERV_SCHED[5]
            NOBUFF_PARAMETERS_SCHED=SERV_SCHED[6]
            DATA_TABS=[[N_TABS_SCHED, T_UPSTREAM_SCHED, METER_SCHED],
                    TIME_VAR_SCHED, TIME_VAR_ME_SCHED, WIP_NO_BUFF_SCHED,
                    GLOBAL_SCHED, NOBUFF_PARAMETERS_SCHED]
        #end if
    #end if
elif SCHED[line]=='mtx_gen\n':
    count_empty=0
    line+=1
    SERV_SCHED=SCHED_DATA_DECODER(SCHED[line],'mtx')
    ERROR_SCHED=SERV_SCHED[0]
```

```
    if ERROR_SCHED==1:
        print('PROGRAMMA DI SCHEDULAZIONE ERRATO 7')
        input('DATI GENERAZIONE MATRICI DI BASE ERRATI, %s DOPO %s'%
          (SCHED[line],SCHED[line-1]))
        CTRL_SCHED=-1
    else:
        DATA_SCHED=SERV_SCHED[1]
        SERV_SCHED=GENERAL_MATRIX_GENERATION('sched',DATA_SCHED)
        if not(SERV_SCHED[0]==1):
            CTRL_SCHED=-1
            print('PROGRAMMA DI SCHEDULAZIONE ERRATO 8')
            input('errore in generazione matrici di base')
        else:
            CTRL_SCHED=0
            STATUS_MTX_SCHED=0
            DELTA_Z_SCHED=SERV_SCHED[1]
            CINT_Z_SCHED=SERV_SCHED[2]
            CF_Z_SCHED=SERV_SCHED[3]
            CINT_TA_SCHED=SERV_SCHED[4]
            CF_TA_SCHED=SERV_SCHED[5]
            N_MTX_SCHED=DELTA_Z_SCHED.cols
            CF_TU_SCHED=SERV_SCHED[6]
            CINT_TA_GLBL_SCHED=SERV_SCHED[7]
            CDEN_TA_GLBL_SCHED=SERV_SCHED[8]
            CINT_TW_GLBL_SCHED=SERV_SCHED[9]
            CDEN_TW_GLBL_SCHED=SERV_SCHED[10]
            CDEN_TU_GLBL_SCHED=SERV_SCHED[11]
            DATA_MTX=[DELTA_Z_SCHED, CINT_Z_SCHED, CF_Z_SCHED,
                    CINT_TA_SCHED, CF_TA_SCHED, N_MTX_SCHED, CF_TU_SCHED,
                    CINT_TA_GLBL_SCHED, CDEN_TA_GLBL_SCHED,
                    CINT_TW_GLBL_SCHED, CDEN_TW_GLBL_SCHED,
                    CDEN_TU_GLBL_SCHED]
        #end if
    #end if
elif SCHED[line]=='timeless_gen\n':
    count_empty=0
    line+=1
    SERV_SCHED=SCHED_DATA_DECODER(SCHED[line],'timeless')
    ERROR_SCHED=SERV_SCHED[0]
    if ERROR_SCHED==1:
        print('PROGRAMMA DI SCHEDULAZIONE ERRATO 9')
        input('DATI GENERAZIONE TABULATI TIMELESS, %s DOPO %s'%
          (SCHED[line],SCHED[line-1]))
        CTRL_SCHED=-1
    else:
        DATA_SCHED=[]
        DATA_SCHED.append(0)
        for i in range(len(DATA_MTX)):
            DATA_SCHED.append(DATA_MTX[i])
        #end for
```

```
            DATA_SCHED.append(SERV_SCHED[1][0])
            DATA_SCHED.append(SERV_SCHED[1][1])
            SERV_SCHED=TIMELESS('sched',DATA_SCHED)
            if SERV_SCHED[0]==0:
               CTRL_SCHED=-1
               print('PROGRAMMA DI SCHEDULAZIONE ERRATO 10')
               input('errore in generazione tabulati timeless')
            else:
               CTRL_SCHED=0
               STATUS_TIMELESS_SCHED=0
               N_TABS_TIMELESS_SCHED=SERV_SCHED[1]
               TIME_VAR_TIMELESS_SCHED=SERV_SCHED[2]
               TIME_VAR_TIMELESS_MED_SCHED=SERV_SCHED[3]
               WIP_NO_BUFF_TIMELESS_SCHED=SERV_SCHED[4]
               GLOBAL_TIMELESS_SCHED=SERV_SCHED[5]
               NOBUFF_TIMELESS_PARAMETERS_SCHED=SERV_SCHED[6]
               DATA_TIMELESS=[N_TABS_TIMELESS_SCHED,
                        TIME_VAR_TIMELESS_SCHED,
                        TIME_VAR_TIMELESS_MED_SCHED,
                        WIP_NO_BUFF_TIMELESS_SCHED, GLOBAL_TIMELESS_SCHED,
                        NOBUFF_TIMELESS_PARAMETERS_SCHED]
            #end if
         #end if
      elif SCHED[line]=='tabs_gen\n':
         count_empty=0
         line+=1
         SERV_SCHED=SCHED_DATA_DECODER(SCHED[line],'tabs')
         ERROR_SCHED=SERV_SCHED[0]
         if ERROR_SCHED==1:
            print('PROGRAMMA DI SCHEDULAZIONE ERRATO 11')
            input('DATI GENERAZIONE TABULATI CASO SPECIFICO ERRATI, %s DOPO
              %s'%(SCHED[line],SCHED[line-1]))
            CTRL_SCHED=-1
         else:
            DATA_SCHED=[]
            DATA_SCHED.append(0)
            for i_data in range(len(DATA_TIMELESS)):
               DATA_SCHED.append(DATA_TIMELESS[i_data])
            #end for
            for i_data in range(len(SERV_SCHED[1])):
               DATA_SCHED.append(SERV_SCHED[1][i_data])
            #end for
            SERV_SCHED=NEW_SET_ELABORATION('sched',DATA_SCHED)
            if not(SERV_SCHED[0]==1):
               CTRL_SCHED=-1
               print('PROGRAMMA DI SCHEDULAZIONE ERRATO 12')
               input('errore in apertura tabulati caso specifico')
            else:
               CTRL_SCHED=0
               STATUS_TABS_SCHED=0
```

```
            N_TABS_SCHED=int(SERV_SCHED[1][0])
            T_UPSTREAM_SCHED=mp.mpf(SERV_SCHED[1][1])
            METER_SCHED=SERV_SCHED[1][2]
            TIME_VAR_SCHED=SERV_SCHED[2]
            TIME_VAR_ME_SCHED=SERV_SCHED[3]
            WIP_NO_BUFF_SCHED=SERV_SCHED[4]
            GLOBAL_SCHED=SERV_SCHED[5]
            NOBUFF_PARAMETERS_SCHED=SERV_SCHED[6]
            DATA_TABS=[[N_TABS_SCHED, T_UPSTREAM_SCHED, METER_SCHED],
                  TIME_VAR_SCHED, TIME_VAR_ME_SCHED, WIP_NO_BUFF_SCHED,
                  GLOBAL_SCHED, NOBUFF_PARAMETERS_SCHED]
         #end if
      #end if
   elif SCHED[line]=='buff_gen\n':
      count_empty=0
      line+=1
      SERV_SCHED=SCHED_DATA_DECODER(SCHED[line],'buff')
      ERROR_SCHED=SERV_SCHED[0]
      if ERROR_SCHED==1:
         print('PROGRAMMA DI SCHEDULAZIONE ERRATO 13')
         input('DATI GENERAZIONE TABULATI CASO SPECIFICO ERRATI, %s DOPO
           %s'%(SCHED[line],SCHED[line-1]))
         CTRL_SCHED=-1
      else:
         DATA_SCHED=[]
         DATA_SCHED.append(0)
         for i_data in range(len(DATA_TABS)):
            DATA_SCHED.append(DATA_TABS[i_data])
         #end for i_data
         for i_data in range(len(SERV_SCHED[1])):
            DATA_SCHED.append(SERV_SCHED[1][i_data])
         #end for i_data
         SERV_SCHED=BUFFER_ELABORATION('sched',DATA_SCHED)
         if not(SERV_SCHED[0]==1):
            CTRL_SCHED=-1
            print('PROGRAMMA DI SCHEDULAZIONE ERRATO 14')
            input('errore in apertura tabulati caso specifico')
         else:
            CTRL_SCHED=0
         #end if
      #end if
   elif SCHED[line]=='single_nb\n':
      count_empty=0
      line+=1
      SERV_SCHED=SCHED_DATA_DECODER(SCHED[line],'single_nb')
      ERROR_SCHED=SERV_SCHED[0]
      if ERROR_SCHED==1:
         print('PROGRAMMA DI SCHEDULAZIONE ERRATO 15')
         input('DATI GENERAZIONE SIMULAZIONE SENZA BUFFER ERRATI')
         CTRL_SCHED=-1
```

```
    else:
        CTRL_SCHED=0
        DATA_SCHED=SERV_SCHED[1]
        SINGLE_SIM('sched',DATA_SCHED)
    #end if
elif SCHED[line]=='single_buff\n':
    count_empty=0
    line+=1
    SERV_SCHED=SCHED_DATA_DECODER(SCHED[line],'single_buff')
    ERROR_SCHED=SERV_SCHED[0]
    if ERROR_SCHED==1:
        print('PROGRAMMA DI SCHEDULAZIONE ERRATO 16')
        input('DATI GENERAZIONE SIMULAZIONE CON BUFFER ERRATI')
        CTRL_SCHED=-1
    else:
        CTRL_SCHED=0
        DATA_SCHED=SERV_SCHED[1]
        SINGLE_SIM('sched',DATA_SCHED)
    #end if
elif SCHED[line]=='multi_nb\n':
    count_empty=0
    line+=1
    SERV_SCHED=SCHED_DATA_DECODER(SCHED[line],'multi_nb')
    ERROR_SCHED=SERV_SCHED[0]
    if ERROR_SCHED==1:
        print('PROGRAMMA DI SCHEDULAZIONE ERRATO 17')
        input('DATI GENERAZIONE BLOCCO SIMULAZIONI SENZA BUFFER')
        CTRL_SCHED=-1
    else:
        CTRL_SCHED=0
        DATA_SCHED=SERV_SCHED[1]
        MULTI_SIM('sched',DATA_SCHED)
    #end if
elif SCHED[line]=='multi_buff\n':
    count_empty=0
    line+=1
    SERV_SCHED=SCHED_DATA_DECODER(SCHED[line],'multi_buff')
    ERROR_SCHED=SERV_SCHED[0]
    if ERROR_SCHED==1:
        print('PROGRAMMA DI SCHEDULAZIONE ERRATO 18')
        input('DATI GENERAZIONE BLOCCO SIMULAZIONI CON BUFFER')
        CTRL_SCHED=-1
    else:
        CTRL_SCHED=0
        DATA_SCHED=SERV_SCHED[1]
        MULTI_SIM('sched',DATA_SCHED)
    #end if
elif SCHED[line]=='comp_tab\n':
    count_empty=0
     line+=1
```

```
            SERV_SCHED=SCHED_DATA_DECODER(SCHED[line],'comp_tab')
            ERROR_SCHED=SERV_SCHED[0]
            if ERROR_SCHED==1:
               print('PROGRAMMA DI SCHEDULAZIONE ERRATO 19')
               input('DATI GENERAZIONE BLOCCO COMPARAZIONI SENZA BUFFER')
               CTRL_SCHED=-1
            else:
               CTRL_SCHED=0
               DATA_SCHED=SERV_SCHED[1]
               USER_COMPARATORE_MAIN('sched',DATA_SCHED)
            #end if
         elif SCHED[line]=='comp_buff\n':
            count_empty=0
            line+=1
            SERV_SCHED=SCHED_DATA_DECODER(SCHED[line],'comp_buff')
            ERROR_SCHED=SERV_SCHED[0]
            if ERROR_SCHED==1:
               print('PROGRAMMA DI SCHEDULAZIONE ERRATO 20')
               input('DATI GENERAZIONE BLOCCO COMPARAZIONI CON BUFFER')
               CTRL_SCHED=-1
            else:
               CTRL_SCHED=0
               DATA_SCHED=SERV_SCHED[1]
               USER_COMPARATORE_MAIN('sched',DATA_SCHED)
            #end if
         elif SCHED[line]=='end\n':
            CTRL_SCHED=2
         else:
            count_empty+=1
         #end if
         line+=1
      #end while
      if count_empty>4:
         print('Raggiunto livello eccessive linee vuote')
         input('PROGRAMMA DI SCHEDULAZIONE ERRATO 21')
      elif CTRL_SCHED==2:
         input('Schedulazione eseguita correttamente')
      else:
         input('Schedulazione interrotta causa assenza di linee di comando')
      #end if
   return()
#end def SCHED_EXEC


def SCHEDULER_MAIN():
#Funzione custom adibita alla gestione del menù di scelta delle operazioni eseguibili in modalità
#schedulazione.
#L'input viene gestito dalla funzione custom INPUT_MENU.
#In base ad esso verranno richiamate le funzioni custom SCHED_EXEC e
#SCHED_CREATE_MAIN.
```

```
    MENU_CTRL_MAIN=0
    while MENU_CTRL_MAIN==0:
        SERV_MAIN=INPUT_MENU('main_sched',0)   #main_sched ha 3 opzioni
        ERROR_CTRL_MAIN=SERV_MAIN[0]
        if ERROR_CTRL_MAIN==1:
            print('Limite errori di immissione dati raggiunto')
            MENU_CTRL_MAIN=-1
        elif ERROR_CTRL_MAIN==0:
            MENU_CTRL_MAIN=SERV_MAIN[1]
            if MENU_CTRL_MAIN==1:
                SCHED_EXEC()
                MENU_CTRL_MAIN=0
            elif MENU_CTRL_MAIN==2:
                SCHED_CREATE_MAIN()
                MENU_CTRL_MAIN=0
            elif MENU_CTRL_MAIN==0:
                MENU_CTRL_MAIN=-1
            #end if
        #end if
    #end while
    return()
#end def SCHEDULER_MAIN()

def PROD_MAPPING():
#Funzione custom che crea tutte le produttorie fino ad un valore massimo stabilito nei parametri
#iniziali del programma.
#Viene richiamata dal programma principale.
    N_PM=PRODUTTORIA_MAX_VALUE
    if PROD_KIND==0:
        MATRIX=mp.zeros(1,N_PM+1)
        MATRIX[0,0]=1
        MATRIX[0,1]=1
        j=2
        while j<N_PM+1:
            print(j)
            MATRIX[0,j]=MATRIX[0,j-1]*j
            j+=1
        #end while
        print('Elaborazione dati iniziali terminata')
        input('Premere un tasto per continuare')
        if PROD_PRINT_ENABLE==1:
            print('Salvataggio su file')
            f=open('produttorie_line_%du.%s'%(N_PM,F_EXT),'w+')
            f.write('%d\n'%(N_PM+1))
            for j in range(N_PM+1):
                print(i)
                f.write('%s%s'%(mp.nstr(MATRIX[0,j],F_PREC_PRG),F_DELIMITER))
            #end for j
            f.write('\n')
            f.close()
```

```
            print('Salvataggio terminato')
        #end if
    elif PROD_KIND==1:
        MATRIX=mp.zeros(N_PM+1,N_PM+1)
        MATRIX[0,0]=1
        MATRIX[1,1]=1
        j=2
        while j<N_PM+1:
            MATRIX[1,j]=MATRIX[1,j-1]*j
            j+=1
        #end while
        i=2
        while i<N_PM+1:
            print(i)
            j=i
            while j<N_PM+1:
                MATRIX[i,j]=MATRIX[i-1,j]/MATRIX[i-1,i-1]
                j+=1
            #end while
            i+=1
        #end while
        print('Elaborazione dati iniziali terminata')
        input('Premere un tasto per continuare')
        if PROD_PRINT_ENABLE==1:
            print('Salvataggio su file')
            f=open('produttorie_%du.%s'%(N_PM,F_EXT),'w+')
            f.write('%d\n'%(N_PM+1))
            for i in range(N_PM+1):
                print(i)
                for j in range(N_PM+1):
                    f.write('%s%s'%(mp.nstr(MATRIX[i,j],F_PREC_PRG),F_DELIMITER))
                #end for j
                f.write('\n')
            #end for i
            f.close()
            print('Salvataggio terminato')
        #end if
    #end if
    return(MATRIX)
#end PROD_MAPPING(PRODUTTORIA_MAX_VALUE, PROD_PRINT_ENABLE)
#PROGRAMMA PRINCIPALE
#Definizione parametri di funzionameno del programma.
#Definizione di alcuni paratri di scrittura nel salvataggio file.
#Definizione estensioni dei files di salvataggio.
#Calcolo produttorie necessarie ai calcoli tramite funzione PROD_MAPPING
#Definizione degli array contenenti i dati necessari alla costruzione dei nomi dei files di
#salvataggio.
#Ciclo WHILE contenente il menu iniziale di scelta gestito tramite INPUT_MENU ed una IF in
#cascata.
#In base alla scelta dell'utente verranno richiamate le funzioni custom USER_MAIN e
```

```
#SCHEDULER_MAIN.

mp.dps=2000
mp.pretty=False

F_DELIMITER=','
F_DELIMITER_OUT='.'
F_EXT='csv'
F_EXT_S='scd'

F_PREC_PRG=100
EXTRACTION_PREC=1000000000000
EXTRACTION_RANGE=10000000000000000

PRODUTTORIA_MAX_VALUE=int(1500)
PROD_KIND=0 #=0 calcola una singola linea, =1 mappa ogni combinazione
PROD_PRINT_ENABLE=0 #0=disable 1=enable

PROD_MAIN=PROD_MAPPING()

F_BASE_NAME=[]
F_BASE_NAME.append(['MTX_','u_','d_DELTA','d_CINT_Z','d_CF_Z','d_CINT_ARR',
        'd_CF_ARR','d_CF_EXIT','d_CINT_GLOB_TARR','d_CF_GLOB_TARR',
        'd_CINT_GLOB_WAIT','d_CF_GLOB_WAIT','d_CF_GLOB_EXIT'])
F_BASE_NAME.append(['TAB_','u_','_','d_TIME_VAR','d_TIME_VAR_MED',
        'd_NOBUFF_WIP_PROBABILITY_MTX','d_GLOBAL_TV',
        'd_NOBUFF_PARAMETERS'])
F_BASE_NAME.append(['BUFF_','u_','_','buff_','d_BUFF_WIP_PROBABILITY_MTX',
        'd_BUF_PARAMETERS'])
F_BASE_NAME.append(['SCHED_(',')'])
F_BASE_NAME.append(['TL_TAB_','u_','d_TIME_VAR','d_TIME_VAR_MED',
        'd_NOBUFF_WIP_PROBABILITY_MTX','d_GLOBAL_TV',
        'd_NOBUFF_PARAMETERS'])

F_NAME_USER=[]
F_NAME_USER.append(['USER_','u_TABELLA COEFICIENTI DELTA', 'u_COEFICIENTI
        FUNZIONI CUMULATA DI Z','u_COEFICIENTI FUNZIONI DI DENSITA DI
        Z','u_COEFICIENTI FUNZIONI CUMULATE TEMPI DI ARRIVO','u_COEFICIENTI
        FUNZIONI DI DENSITA TEMPI DI ARRIVO'])
F_NAME_USER.append(['USER_','_Tempi medi di arrivo, di attesa, di uscita, loro varianze e
        risultati finali','_TABELLA DELLE PROBABILITA DEI LIVELLI DI WIP'])
F_NAME_USER.append(['USER_','_calcoli con buffer='])
F_NAME_USER.append(['USER_', 'u_TIMELESS(Tempi medi di arrivo, di attesa, di uscita, loro
        varianze su singola coppia)', 'u_TIMELESS(MEDIE su N pezzi dei Tempi medi di arrivo, di
        attesa, di uscita, loro varianze su singole coppie', 'u_TIMELESS(Tempi medi di arrivo, di
        attesa, di uscita, loro varianze globali su N coppie)', 'u_TIMELESS(tabella delle probabilità
        di WIP in assenza di buffer)', '_TIMELESS(parametri su N coppie)'])

F_NAME_COMP=[]
F_NAME_COMP.append(['COMP_','_TEMPI ARRIVO','TEMPI ATTESA','TEMPI
```

```
        USCITA','LIVELLI WIP'])
F_NAME_COMP.append(['COMP_','TEMPI ATTESA','TEMPI USCITA','LIVELLI WIP'])

MENU_RANGE=5

MENU_CTRL=0
while MENU_CTRL==0:
   SERV=INPUT_MENU('main',0)
   ERROR_CTRL=SERV[0]
   if ERROR_CTRL==1:
      print('Limite errori di immissione dati raggiunto')
      MENU_CTRL=-1
   elif ERROR_CTRL==0:
      MENU_CTRL=SERV[1]
      if MENU_CTRL==1:
         USER_MAIN()
         MENU_CTRL=0
      elif MENU_CTRL==2:
         SCHEDULER_MAIN()
         MENU_CTRL=0
      elif MENU_CTRL==0:    #uscita
         MENU_CTRL=-1
      #end if
   #end if
#end while
input('Chiusura programma')
print('Arrivederci')
#End Program
```

BIBLIOGRAFIA

- [Buzacott.JA,Shabthikumar.JG]-Stochastic Models Of Manufactruing Systems-1993-PrenticeHall
- [Zipkin.PH]-Foundations Of Inventory Management-2009-McGrawHill
- [Hopp.WJ, Spearman.ML]-Factory Physics, third edition-2008-McGrawHill
- [Sheldon.MR]-Probabilità e statistica per l'ingegneria e le scienze, seconda edizione-2008-APOGEO

Ringraziamenti

Oltre ai doverosi ringraziamenti al Prof. Chiabert che, nell'arco degli ultimi due anni, ha avuto la pazienza di sopportarmi e guidarmi nel corso di questo lavoro, vorrei ringraziare tutte le persone che questa "fortuna" l'hanno avuta per più tempo ancora.

In particolar modo:

- mia Madre e mio Padre, per l'Amore, la dedizione, la coerenza, l'abnegazione, le energie e un infinità di emozioni e speranze riversate nel crescermi, nonostante la sorte loro avversa, nonostante la cocciutaggine e la tardiva comprensione del soggetto coinvolto;
- Fabio Mininno, senza l'aiuto del quale non avrei potuto terminare gli studi, per essere diventato quel Fratello maggiore che auguro a chiunque di trovare nell'ora più buia, che sono fiero di avere al fianco in questo traguardo;
- Giorgio Gherlone, per l'ospitalità offertami quando non avevo modo di provvedere diversamente, per avermi accolto come un figlio, per le prediche per bene e non per male;
- i miei Cugini, per l'affetto che mi hanno sempre fatto pervenire;
- Famiglia Piton, per le ripetute spronate, senza mai demordere;
- Marco Manighetti e Famiglia, per esser stati Amici fedeli;
- Franco, per la quieta energia che infonde;
- Famiglia Divaia, per la solidarietà;
- tutti gli Amici che in momenti diversi mi sono stati vicini;
- ai miei Zii;
- tutti i Professori e gli Insegnanti di cui ho avuto l'onore di essere allievo.

A tutti Loro dedico la frase che ero solito dire a mia Madre ogni volta che raggiungevo un obiettivo: **"Abbiamo vinto"**.