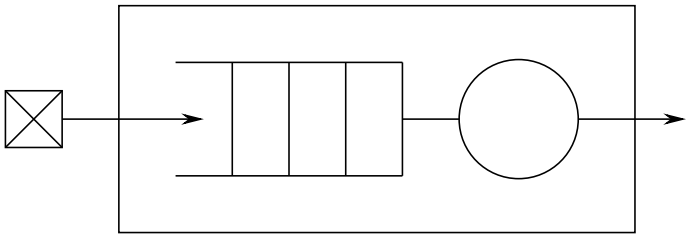
Sistema MM1

Simulazione di un sistema a coda

Pietro Casavecchia

October 30, 2022



Contents

1 Obiettivo

L'obiettivo è quello di simulare il sistema a coda MM1.
Si è cercato di rimanere fedeli al reale funzionamento cercando di interpretare il significato fisico delle componenti del sistema e implementare i funzionamenti in modo da potere essere vicini ad una rappresentazione reale.

2 Struttura

Il programma è stutturato in classi corrispondenti agli oggetti del sistema che sono:

- Class Package
- Class Buffer
- Class Server
- Class System

2.1 Funzione delle Classi

L'obiettivo delle Classi Package, Buffer e Server è quello di creare, nel caso del pacchetto più istanze quindi più oggetti definiti come pacchetti mentre le altre due Classi sono chiamate solo una volta così da creare un oggetto che rappresenti un Buffer e un Server.

La Classe System rappresenta l'intero sistema, gestisce le altre classi e nel caso del Buffer e del Server li chiama una volta sola, mentre chiama la Classe Package ogni volta che un nuovo pacchetto si deve generare.
Inoltre System genera i pacchetti, siccome un Metodo di generazione nella Classe Package non avrebbe senso perchè sarebbe legato a tutti i pacchetti, mentre è necessario avere un Metodo di generazione che comprenda tutti i Pacchetti quindi deve essere un Metodo della Classe System.

2.2 Tempo nella Simulazione

Il tempo è rappresentato da UT quindi Unità di Tempo indefinita, ogni UT è un ciclo di un While che dura fino al tempo di osservazione del sistema.
Per ogni UT vengono chiamate tutte funzioni principali.

3 Codice

Il codice è diviso in Classi e Funzioni per statistiche e plot del Sistema.

3.1 Class Package

```
class Package:
    def __init__(self):
        self.id_number = None
```

Listing 1: Class Package

3.2 Class Buffer

```
class Buffer:
    def __init__(self):
        self.queue = []

    def calculate_buffer_size(self):
        return len(self.queue)
```

Listing 2: Class Buffer

3.3 Class Server init

```
class Server:
    def __init__(self):
        self.pkg_serving = None
        self.status = 0
        self.serving_time = None
        self.service_progression = None

        # parameters and feedback
        self.array_serving_time = []
```

Listing 3: Class Server init

3.4 Server Methods

```
def from_buffer_to_server(self, buffer):
    # FIFO buffer
    # get first pkg
    pkg = buffer.queue[0]
    # eliminate from queue
    buffer.queue.pop(0)
    # move into the server change server status
    self.status = 1
    self.pkg_serving = pkg

    # generate a serving time > 0
    while True:
        self.serving_time = math.floor(np.random.exponential(
            S))
        if self.serving_time > 0:
            break
    # initiate service progression
    self.service_progression = 0

    # append serving time for feedback
    self.array_serving_time.append(self.serving_time)

def service(self, buffer, pkgs_served):
    # if status of the server is zero and a pkg in
    #   the buffer exits then move the pkg
    #   into the server
    if self.status == 0 and buffer.calculate_buffer_size() ==
        0:
        # erase pkg given exited the server
        self.pkg_serving = None
        # empty the server
        self.serving_time = None
        self.service_progression = None
    elif self.status == 0 and buffer.calculate_buffer_size()
       > 0:
        self.from_buffer_to_server(buffer)

    # if the serving time == 0 then after the
    #   from_buffer...() call the status will be 1
    if self.status == 1:
        self.service_progression += 1
        if self.service_progression > self.serving_time:
            # append in pkgs served
            pkgs_served.append(self.pkg_serving)
            self.status = 0
```

Listing 4: Server Methods

3.5 Class System init

```
class System():
    def __init__(self, run_time, IA):
        self.IA = IA
        self.run_time = run_time
        self.current_time = 0
        self.n_pkgs = 0
        self.pkgs_served = []

        # generation of pkg variabiles
        self.inter_arrival_time = None
        self.generation_progression = None

        # generate a buffer and server
        self.buffer = Buffer()
        self.server = Server()

        # parameters and feedback
        # number of pkgs nel sistema for every unit of time
        self.array_inter_arrival_time = []
        self.pkgs_sys_for_unitTime = []
        self.pkgs_queue_for_unitTime = []
        self.dict_pkgsTime_queue = {}
        self.dict_pkgsTime_system = {}
        # array counter for how many pkgs in the
        #       system [a, b]
        # a: how many time there are none and b:
        #       how many time there > 0
        self.array_P01 = [0, 0]

        # start simulation call
        self.simulation()
```

Listing 5: Class System init

3.6 System Method Pkg Generation

```
def pkg_generation(self):
    # initialize pkg to none
    pkg = None
    # run the generation only for the run time time
    if self.current_time < self.run_time:
        # generate pkg zero
        if (
            (self.current_time == 0) or
            (self.generation_progression >= self.
               inter_arrival_time)
```

```
            ):
            pkg = Package()
            pkg.id_number = self.n_pkgs
            self.n_pkgs += 1
            # inter arrival time > 0
            while True:
                self.inter_arrival_time = math.floor(np.
                    random.exponential(self.IA))
                if self.inter_arrival_time > 0:
                    break
            # set generation progression to zero for
                initialize it
            self.generation_progression = 0

            # append gen time for feedback
            self.array_inter_arrival_time.append(self.
                inter_arrival_time)

        # increment only if self.inter_arrival_time != 0
        # because the pkg gen fn will be called again
        if self.inter_arrival_time != 0:
            self.generation_progression += 1
    else:
        self.inter_arrival_time = None
        self.generation_progression = None

    return pkg
```

Listing 6: System Method Pkg Generation

3.7 System Method Simulation

```
def simulation(self):
    # do not continue the process until all pks are served
    #   end with run time
    while (self.current_time < self.run_time):

        # --- --- --- start sys call --- --- ---

        # menage generation
        pkg = self.pkg_generation()
        # manage buffer
        if pkg != None: self.buffer.queue.append(pkg)
        # manage server
        self.server.service(self.buffer, self.pkgs_served)

        # --- --- --- end sys call --- --- ---
```

```
16
17                  # --- --- --- start print df --- --- ---
18          # print pkg
19          if pkg == None: data_list.append(None)
20          elif pkg != None: data_list.append(pkg.id_number)
21
22          # print buffer
23          queue_pkgs = []
24          for i in range(len(self.buffer.queue)):
25              queue_pkgs.append(self.buffer.queue[len(self.
                    buffer.queue) - i - 1].id_number)
26          data_list.append(queue_pkgs)
27          data_list.append(self.buffer.calculate_buffer_size())
28
29          # print server
30          printServerStatus = "-->" if self.server.status == 1
                else "None"
31          data_list.append(printServerStatus)
32
33          # print pkg served
34          if self.server.pkg_serving == None: data_list.append(
                None)
35          elif self.server.pkg_serving != None: data_list.
                append(self.server.pkg_serving.id_number)
36          data_list.append(self.server.serving_time)
37          data_list.append(self.server.service_progression)
38
39          # print sys
40          last_served_pkgs = []
41          for i in range(len(self.pkgs_served)):
42              if i >= 5: break
43              last_served_pkgs.append(self.pkgs_served[len(self
                    .pkgs_served) - i - 1].id_number)
44          data_list.append(len(self.pkgs_served))
45          data_list.append(last_served_pkgs)
46
47          # append to the last index of df that is the len(df)
48          df.loc[len(df)] = data_list
49          # --- --- --- end print df --- --- ---
50
51          # --- --- --- start feedback --- --- ---
52          # Ls Lq
53          self.pkgs_sys_for_unitTime.append(self.n_pkgs - len(
                self.pkgs_served))
54          self.pkgs_queue_for_unitTime.append(self.n_pkgs - len
                (self.pkgs_served) - self.server.status)
55          # calculate how long a pkg in the queue
56          for i in range(len(self.buffer.queue)):
57              # get value of  pkg
```

```
58              value_queue_pkg = self.dict_pkgsTime_queue.get(
                    self.buffer.queue[i].id_number)
59              # if is none set to 1 else increment by 1
60              if value_queue_pkg == None:
61                  self.dict_pkgsTime_queue.update({self.buffer.
                        queue[i].id_number: 1})
62              else:
63                  self.dict_pkgsTime_queue.update({self.buffer.
                        queue[i].id_number: value_queue_pkg + 1})
64          # calculate how long a pkg in the server
65          # read the value in the server
66          #   check if already exits in the dict then
67          #               add 1 else increase by one
68          if self.server.pkg_serving != None:
69              # search it in the
70              value_server_pkg = self.dict_pkgsTime_system.get(
                    self.server.pkg_serving.id_number)
71              if value_server_pkg == None:
72                  self.dict_pkgsTime_system.update({self.server
                        .pkg_serving.id_number: 1})
73              else:
74                  self.dict_pkgsTime_system.update({self.server
                        .pkg_serving.id_number: value_server_pkg +
                         1})
75          # calculate value probability 0 add it to array
76          #       first index is when queue
77          if len(self.buffer.queue) == 0 and self.server.status
                == 0:
78              self.array_P01[0] += 1
79          else:
80              self.array_P01[1] += 1
81          # --- --- --- end feedback --- --- ---
82
83          print(self.n_pkgs, end="\r")
84
85          # increment simulation time
86          self.current_time += 1
```

Listing 7: System Method Simulation

3.8 System Method Parameters

```
def calculate_parameters(self):
    # take the value from the dict of waiting
    #   time in the server / queue
    array_waitingTime_queue = []
    array_waitingTime_server = []
    for values_q in self.dict_pkgsTime_queue.values():
        array_waitingTime_queue.append(values_q)
    # adjust with adding 1 for mean for missing pks
    for _ in range(self.n_pkgs - len(array_waitingTime_queue)
       ):
        array_waitingTime_queue.append(0)

    for values_s in self.dict_pkgsTime_system.values():
        array_waitingTime_server.append(values_s)
    # adjust with adding 1 for mean for missing pks
    for _ in range(self.n_pkgs - len(array_waitingTime_server
       )):
        array_waitingTime_server.append(0)


    return (
        self.array_inter_arrival_time,
        self.server.array_serving_time,
        self.pkgs_sys_for_unitTime,
        self.pkgs_queue_for_unitTime,
        array_waitingTime_server,
        array_waitingTime_queue,
        self.array_P01
        )
```

Listing 8: System Method Parameters

4 Sezione della Simulazione

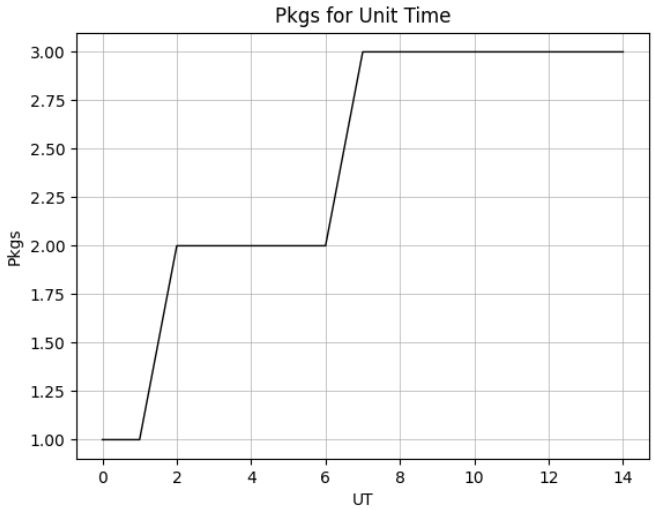
```
    UT  interTime  genProc  pkgIdGen       queue  buffDim  \
0    0       None     None         0          []         0
1    1          1        1         1         [1]         1
2    2          1        1         2         [2]         1
3    3          3        1      None         [2]         1
4    4          3        2      None         [2]         1
5    5          3        3         3      [3, 2]         2
6    6          1        1         4      [4, 3]         2
7    7          1        1         5      [5, 4]         2
8    8          2        1      None      [5, 4]         2
9    9          2        2         6   [6, 5, 4]         3
10  10          3        1      None      [6, 5]         2
11  11          3        2      None      [6, 5]         2
12  12          3        3         7   [7, 6, 5]         3
13  13          5        1      None      [7, 6]         2
14  14          5        2      None      [7, 6]         2

   servStatus  pkgIdServ  servTime  servProc  nPkgsServ  \
0          -->          0         2         1          0
1         None          0         2         2          1
2          -->          1         4         1          1
3          -->          1         4         2          1
4          -->          1         4         3          1
5         None          1         4         4          2
6         None          2         1         1          3
7          -->          3         3         1          3
8          -->          3         3         2          3
9         None          3         3         3          4
10         -->          4         3         1          4
11         -->          4         3         2          4
12        None          4         3         3          5
13         -->          5         4         1          5
14         -->          5         4         2          5

          pkgsServed
0                 []
1                [0]
2                [0]
3                [0]
4                [0]
5             [1, 0]
6          [2, 1, 0]
7          [2, 1, 0]
8          [2, 1, 0]
9       [3, 2, 1, 0]
10      [3, 2, 1, 0]
```

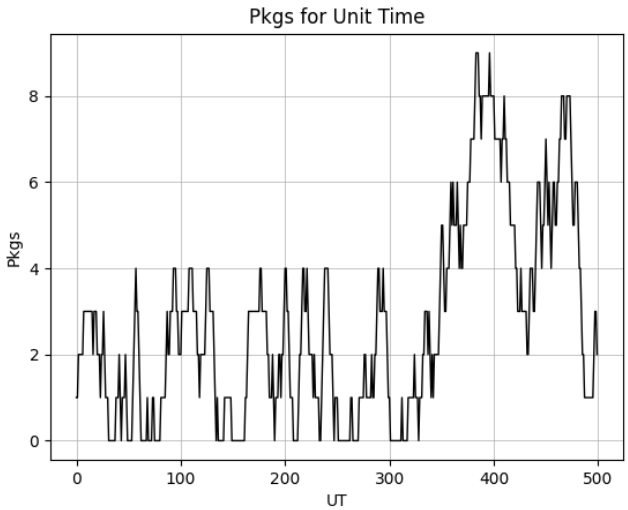
```
11      [3, 2, 1, 0]
12   [4, 3, 2, 1, 0]
13   [4, 3, 2, 1, 0]
14   [4, 3, 2, 1, 0]
```

Listing 9: System Section

4.1 Grafici Pkgs nel Sistema per Unità di Tempo

Il primo rappresenta la sezione fino a 15ut, invece il secondo una simulazione con lo stesso Seed fino a 500ut





5 Parametri del Sistema

Vengono calcolati i parametri teorici usando le seguenti equazioni, poi vengono calcolati i parametri sperimentali, concretizzando il significato pratico delle equazioni mettendo degli appositi Flag nella simulazione, così da avere per ogni parametro teorico il corrispettivo sperimentale.
Viene calcolato anche λ e μ sperimentale osservando il Tempo di Interarrivo (IA) e il Tempo di Servizio (S) generati dalle opportune distribuzioni esponenziali.

5.1 Equazioni

Dato il Tempo di Interarrivo, IA e il Tempo di Servizio, S :

$$\lambda = \frac{1}{IA} \qquad\qquad \mu = \frac{1}{S}$$

Steady State:

$$\lambda_k = \lambda \qquad\qquad \mu_k = \mu$$

Fattore di utilizzo o percentuale del tempo che tutti i Servers sono occupati:

$$\rho = \frac{\lambda}{\mu} \qquad\qquad \rho = 1 - P_0$$

P_0 Probabilità di avere 0 Pkgs nel Sistema:

$$P_0 = 1 - \frac{\lambda}{\mu} = 1 - \rho$$

P_k Probabilità di avere k Pkgs nel Sistema:

$$P_k = P_0 \left(\frac{\lambda}{\mu}\right)^k = \rho^k \cdot P_0$$

W_S Tempo medio di un Pkg nel Sistema, W_q tempo medio di un Pkg nel Buffer:

$$W_S = \frac{L_S}{\lambda} = \frac{1}{\mu - \lambda} \qquad\qquad W_q = W_S - \frac{1}{\mu} = \frac{\lambda}{\mu(\mu - \lambda)}$$

L_S Numero medio di Pkgs nel Sistema, L_q Numero medio di Pkgs nel Buffer:

$$L_S = \frac{\rho}{1 - \rho} \qquad\qquad L_q = \frac{\lambda^2}{\mu(\mu - \lambda)} = \lambda W_q = \frac{\rho^2}{1 - \rho}$$

5.2 Esempio Output con λ e μ Fissati

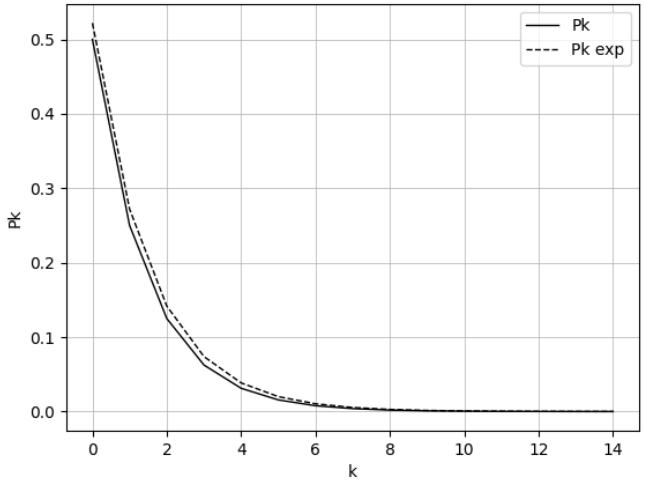
Viene fissato il Tempo di Interarrivo e il Tempo di Servizio quindi di conseguenza λ e μ poi vengono fatte n simulazione per avere un valore medio dei parametri, in questo caso il numero di simulazioni era 5 con un Tempo di Simulazione di 40000ut.

```
IA theo: 18.5 IA out: 21.816 error: 17
S theo: 5 S out: 5.532 error: 10
Lambda theo: 0.054 Lambda out: 0.046 error: -15
Mu theo: 0.2 Mu out: 0.181 error: -10
Rho theo: 0.27 Rho out: 0.256 error: -6
Ls theo: 0.37 Ls out: 0.35 error: -6
Lq theo: 0.1 Lq out: 0.094 error: -7
Ws theo: 6.852 Ws out: 8.546 error: 24
Wq theo: 1.852 Wq out: 2.015 error: 8
P0 theo: 0.73 P0 out: 0.732 error: 0
```

Listing 10: Output Parameters Sample

5.3 Parametro P

Grafico teorico e sperimentale dell'andamento di Pk in funzione di k tenendo ρ costante, il massimo k in questo caso si è scelto arbitrariamente per semplicià ma si potrebbe scegliere il massimo k osservato nella simulazione.



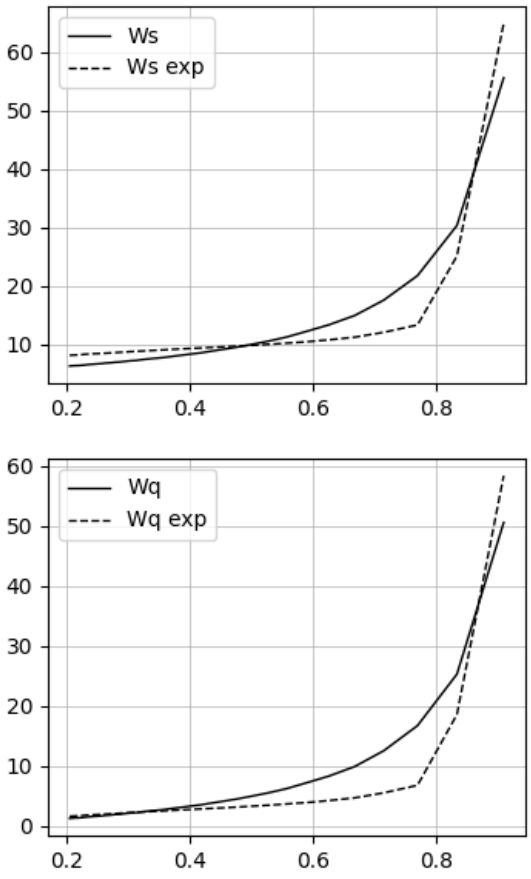
5.4 Output con Variazione di ρ

Si è fissato il Tempo di Servizio e si è fatto variare in modo incrementale il Tempo di Interarrivo mantenendo sempre valida la relazione:

$$\lambda < \mu \qquad (IA) > S \qquad 0 < \rho < 1$$

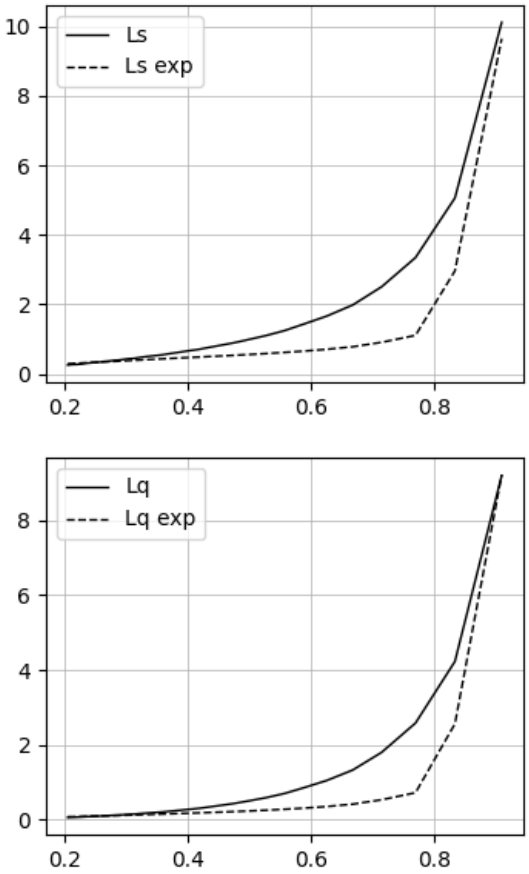
5.5 Parametro W

Grafico teorico e sperimentale dell'andamento di W_S e W_q in funzione di ρ



5.6 Parametro L

Grafico teorico e sperimentale dell'andamento di L_S e L_q in funzione di ρ



6 Considerazioni

Di seguito vengono elencate le principali considerazioni fatte per la simulazione.

6.1 Scarto dello Zero

Usando appropriate distribuzioni esponenziali per la generazione di λ e μ, vengono eliminati i casi nei quali restituiscono valore 0 perchè essendo la simulazione in tempi discreti si avrebbero multipli arrivi e servizi nella stessa Unità di Tempo, anche se potrebbe essere una buona approssimazione della realtà, si intende un UT molto piccola che quindi non permette piu di un arrivo o servizio.

6.2 Osservazione della Simulazione

Il tempo di osservazione della simulazione parte da un tempo $t = 0$ e finisce in un tempo $t = end$ quindi non si ferma la generazione dei Pkgs e poi si aspetta il loro servizio ma si smette di osservare il sistema quando ancora i Pkgs si stanno generando. Più ρ è piccolo più la differenza tra questi due approcci è meno significativa perchè c'è meno utilizzo dei Server e quindi meno Pkgs da servire dopo la fine della generazione.

6.3 Lunghezza della Simulazione

Più la simulazione è lunga in UT più si da spazio a tempi di generazione e servizio più lunghi.

6.4 Errore in funzione di ρ

Più ρ è grande più l'errore su i parametri aumenta.

Listings