Question 1.1

Write the formulas for the average time spent by a customer in the system and the system occupancy of an M/M/1 queue.

The average time spent by a customer in the system:

$$T = \frac{\bar{N}}{\lambda} = \frac{1}{\mu - \lambda} \tag{1}$$

where μ is the service rate and λ is the arrival rate.

Question 1.2

What is the relationship between total time spent in the system, queueing delay, and service time? Similarly, what is the difference between the number of customers (packets) in the queue, that are currently serviced, and the total system occupancy?

$$T = \bar{x} + W \tag{2}$$

Total time: T
Queuing delay (waiting time): W
Service time: $\bar{x}$

Question 1.3

Estimate the expected system size and the expected time spent in system based on your simulations. Then, compute the expected theoretical values (in Python or Matlab) for these.

$$T = \frac{1}{\mu - \lambda} = \frac{1}{4-3} = 1 \tag{3}$$

$$p = \frac{\lambda}{\mu} = \frac{3}{4} \tag{4}$$

$$N = \frac{p}{1-p} = 3 \tag{5}$$

Mean theoretical delay: 1.000000
Mean theoretical size : 3.000000

Question 1.4

Compare the simulation results with theoretical values. For what number of arrivals do these two become similar (answer in terms of order of magnitude)?

Mean practical delay: 0.990398
Mean practical size : 2.982140

When the number of arrivals reaches 10^5, the theoretical and empirical values become similar.

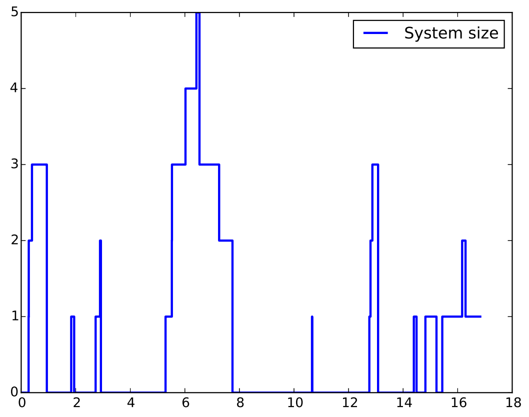

Figure 1: M/M/1 System Size

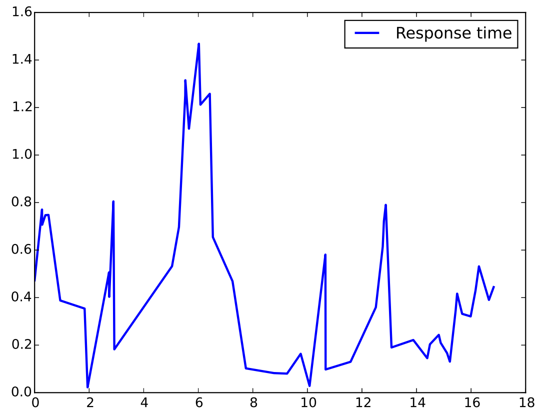

Figure 2: M/M/1 Response Time

Question 1.5

Experiment with other types of interarrival and service time combinations, e.g. M/D/1, D/M/1, D/D/1. You only need to provide practical calculation values.

M/D/1

Mean practical delay: 0.639085
Mean practical size : 1.936220

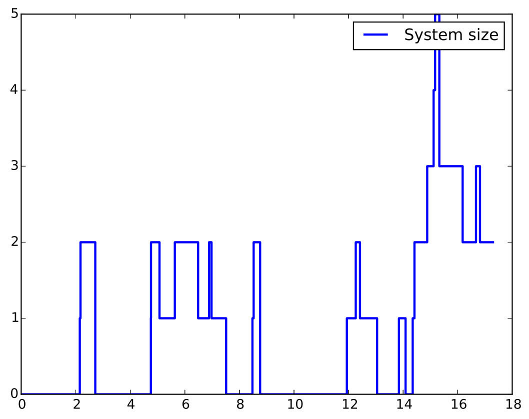

Figure 3: M/D/1 System Size

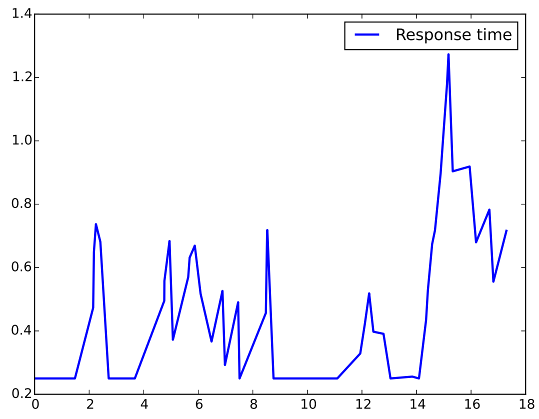

Figure 4: M/D/1 Response Time

D/M/1

Mean practical delay: 0.548419
Mean practical size : 1.197356

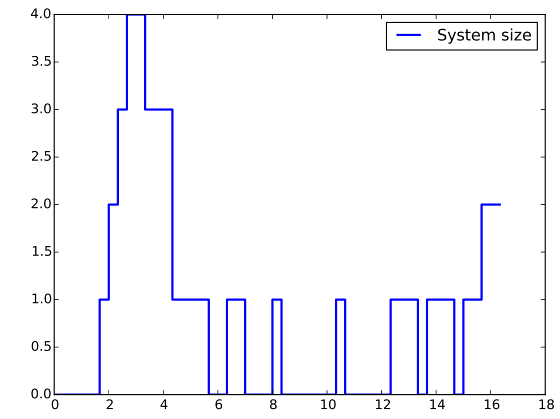

Figure 5: D/M/1 System Size

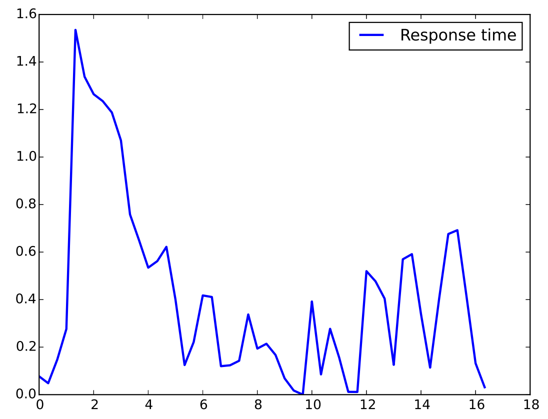

Figure 6: D/M/1 Response Time

D/D/1

Mean practical delay: 0.250000
Mean practical size : 0.000000

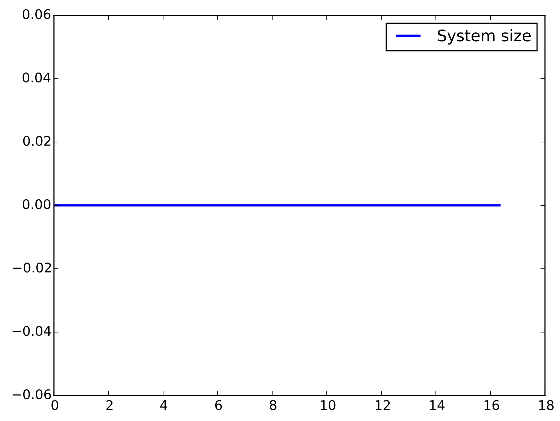

Figure 7: D/D/1 System Size

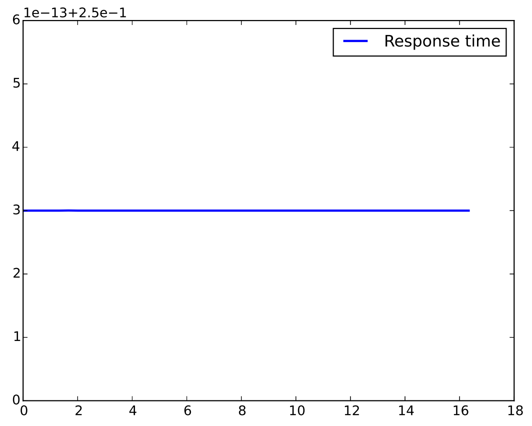

Figure 8: D/D/1 Response Time

Question 1.6

Provide a working copy of the programs you have written in Step 1.2 according to the guidelines for full credit.

M/M/1	`section1/mm1.py`
M/D/1	`section1/md1.py`
D/M/1	`section1/dm1.py`
D/D/1	`section1/dd1.py`

Question 2.1

Provide a working copy of the program you write in this step according to the guidelines for full credit.

Mean practical delay: 1.067071
Mean practical size : 5.327433

Mean theoretical delay: 1.090909
Mean theoretical size : 5.454545

Program: `section2/mmc.py`

Bonus Question

Experiment with other queue types, e.g. M/D/2, D/M/2, D/D/2. Especially, investigate queue stability for various parameters and briefly interpret your observations.

m the number of servers
λ arrival rate
μ service rate per server

$\lambda \leq m\mu$ is the stability requirement. When $\lambda > m\mu$, the queuing system becomes unstable.

M/D/2	`section2/mdc.py`
D/M/2	`section2/dmc.py`
D/D/2	`section2/ddc.py`

Question 3.1

Calculate the theoretical expected system size and the expected time spent in system for each case above (in Python or Matlab) using standard formulas.

Case 1

Mean theoretical delay: 1.000000
Mean theoretical size : 5.000000

Case 2

Mean theoretical delay: 2.000000
Mean theoretical size : 10.000000

Case 3

Mean theoretical delay: 1.090909
Mean theoretical size : 5.454545

Question 3.2

You have already simulated two of the three cases above. Simulate the third case and provide your working program according to the guidelines.

Program: `section3/Q31.py`

Question 3.3

Calculate the performance of each option based on your simulations and check your results against theoretical values.

Case 1

Mean practical delay: $1.039124 \approx 1$
Mean practical size : $5.186733 \approx 5$

Case 2

Q0 mean practical delay: 2.116894
Q0 mean practical size : 5.257232
Q1 mean practical delay: 1.882052
Q1 mean practical size : 4.686604
Combined mean practical delay: $1.999473 \approx 2$
Combined mean practical size : $9.943836 \approx 10$

Case 3

Mean practical delay: $1.067071 \approx 1.090909$
Mean practical size : $5.327433 \approx 5.454545$

Simulation results are similar to theoretical values.

Question 3.4

Which of the three options above would you choose? Why? Discuss briefly based on the simulation and theoretical analysis you have conducted.

Time spent in option 1 and option 3 is much shorter than in option 2. The goal is to minimize the time, so option 2 will not be adopted. Option 1 and option 3 have relatively equal waiting time, however, option 3 consisting of 3 routers is more complicated than option 1. To conclude, option 1 is the best solution.

Appendix

section1/mm1.py

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Jul 15 12:16:53 2014

HINTS:
1) Analyse the script from bottom to top
2) Fill in the blanks marked with ???

@author: alpcan
"""

import numpy as np
import matplotlib.pylab as plt
from wsmm1helper import *



def Theoreticalmm1(srate,arate):
    meandelay = 1/(srate-arate)
    rho = float(arate)/float(srate)
    meansize = rho/(1-rho)

    # Hint for display...
    print 'Mean theoretical delay: {:4f} \n'.format(meandelay)
    print 'Mean theoretical size : {:4f} \n'.format(meansize)


############################################################
## Main program

# parameters

maxsteps=int(1E5)    # simulation steps
srate=4              # service rate
arate=3              # arrival rate


# create simulation
simulation=DESmm1(srate,arate,maxsteps)

# main loop
for i in range(maxsteps):
    intarrive=np.random.exponential(1.0/arate)  # interarrival time
    simulation.packetarrival(intarrive)
    servetime=np.random.exponential(1.0/srate)  # service time
    simulation.nextstep(servetime)

# calculate and print theoretical values
# you can also do this in Matlab if you prefer!
Theoreticalmm1(srate,arate)

# calculate and print practical delay, size values
# optionally visualise
simulation.practicalcalc(True,True)
```

section1/md1.py

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Jul 15 12:16:53 2014

HINTS:
1) Analyse the script from bottom to top
2) Fill in the blanks marked with ???

@author: alpcan
"""

import numpy as np
import matplotlib.pylab as plt
from wsmm1helper import *



def Theoreticalmm1(srate,arate):
    '''
    Theoretically calculates and prints mean delay and system size.

    Inputs: service and arrival rates.
    '''

    '''
    ??? 'You can optionally do this in Matlab, if you wish!'
    '''

    # Hint for display...
    #print 'Mean theoretical delay: {:4f} \n'.format(meandelay)
    #print 'Mean theoretical size : {:4f} \n'.format(meansize)


############################################################
## Main program

# parameters

maxsteps=int(1E5)   # simulation steps
srate=4             # service rate
arate=3             # arrival rate


# create simulation
simulation=DESmm1(srate,arate,maxsteps)

# main loop
for i in range(maxsteps):
    intarrive=np.random.exponential(1.0/arate)  # interarrival time
    simulation.packetarrival(intarrive)
    servetime=1.0/srate  # service time
    simulation.nextstep(servetime)

# calculate and print theoretical values
# you can also do this in Matlab if you prefer!
Theoreticalmm1(srate,arate)

# calculate and print practical delay, size values
# optionally visualise
simulation.practicalcalc(True,True)
```

section1/dm1.py

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jul 15 12:16:53 2014

HINTS:
1) Analyse the script from bottom to top
2) Fill in the blanks marked with ???

@author: alpcan
"""

import numpy as np
import matplotlib.pylab as plt
from wsmm1helper import *



def Theoreticalmm1(srate,arate):
    '''
    Theoretically calculates and prints mean delay and system size.

    Inputs: service and arrival rates.
    '''

    '''
    ??? 'You can optionally do this in Matlab, if you wish!'
    '''

    # Hint for display...
    #print 'Mean theoretical delay: {:4f} \n'.format(meandelay)
    #print 'Mean theoretical size : {:4f} \n'.format(meansize)


#############################################################
## Main program

# parameters

maxsteps=int(1E5)   # simulation steps
srate=4             # service rate
arate=3             # arrival rate


# create simulation
simulation=DESmm1(srate,arate,maxsteps)

# main loop
for i in range(maxsteps):
    intarrive=1.0/arate  # interarrival time
    simulation.packetarrival(intarrive)
    servetime=np.random.exponential(1.0/srate)  # service time
    simulation.nextstep(servetime)

# calculate and print theoretical values
# you can also do this in Matlab if you prefer!
Theoreticalmm1(srate,arate)

# calculate and print practical delay, size values
# optionally visualise
simulation.practicalcalc(True,True)
```

section1/dd1.py

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Jul 15 12:16:53 2014

HINTS:
1) Analyse the script from bottom to top
2) Fill in the blanks marked with ???

@author: alpcan
"""

import numpy as np
import matplotlib.pylab as plt
from wsmm1helper import *



def Theoreticalmm1(srate,arate):
    '''
    Theoretically calculates and prints mean delay and system size.

    Inputs: service and arrival rates.
    '''

    '''
    ??? 'You can optionally do this in Matlab, if you wish!'
    '''

    # Hint for display...
    #print 'Mean theoretical delay: {:4f} \n'.format(meandelay)
    #print 'Mean theoretical size : {:4f} \n'.format(meansize)


#############################################################
## Main program

# parameters

maxsteps=int(1E5)    # simulation steps
srate=4              # service rate
arate=3              # arrival rate


# create simulation
simulation=DESmm1(srate,arate,maxsteps)

# main loop
for i in range(maxsteps):
    intarrive=1.0/arate  # interarrival time
    simulation.packetarrival(intarrive)
    servetime=1.0/srate  # service time
    simulation.nextstep(servetime)

# calculate and print theoretical values
# you can also do this in Matlab if you prefer!
Theoreticalmm1(srate,arate)

# calculate and print practical delay, size values
# optionally visualise
simulation.practicalcalc(True,True)
```

section2/mmc.py

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 14 18:42:28 2014

HINTS:
1) Analyse the script from bottom to top
2) Fill in the blanks marked with ???


@author: alpcan
"""

import math
import numpy as np
from wsmmchelper import *


def Theoreticalmmk(srate,arate,m):

    rho = float(arate) / (m * srate)

    tmp = 0.0
    for k in range(m):
        tmp += (m*rho)**k / math.factorial(k)

    tmp += (m*rho)**m / math.factorial(m) / (1 - rho)
    p0 = 1.0 / tmp

    Nq = (m*rho)**m * rho / math.factorial(m) / (1-rho)**2 * p0
    W = Nq / arate
    meandelay = 1.0 / srate + W
    meansize = arate * meandelay

    # Hint for display...
    print 'Mean theoretical delay: {:4f} \n'.format(meandelay)
    print 'Mean theoretical size : {:4f} \n'.format(meansize)


#parameters

maxsteps=int(1E5)        # simulation steps
srate=3                  # service rate
arate=5                  # arrrival rate
nbrservers=2             # number of servers

simulation=DESmmc(srate,arate,nbrservers,maxsteps)

for i in range(maxsteps):
    intarrive=np.random.exponential(1.0/arate)  # interarrival time
    simulation.packetarrival(simulation.Q,intarrive)
    servetime=np.random.exponential(1.0/srate)  # service time
    simulation.nextstep(servetime)

simulation.practicalcalc()
Theoreticalmmk(srate,arate,nbrservers)
```

section2/mdc.py

```python
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 14 18:42:28 2014

HINTS:
1) Analyse the script from bottom to top
2) Fill in the blanks marked with ???


@author: alpcan
"""

import math
import numpy as np
from wsmmchelper import *


def Theoreticalmmk(srate,arate,k):

    '''
    ??? 'You can optionally do this in Matlab, if you wish!'
    '''

    # Hint for display...
    #print 'Mean theoretical delay: {:4f} \n'.format(meandelay)
    #print 'Mean theoretical size : {:4f} \n'.format(meansize)


#parameters

maxsteps==int(1E5)       # simulation steps
srate=3                  # service rate
arate=5                  # arrrival rate
nbrservers=2             # number of servers

simulation=DESmmc(srate,arate,nbrservers,maxsteps)

for i in range(maxsteps):
    intarrive=np.random.exponential(1.0/arate)  # interarrival time
    simulation.packetarrival(simulation.Q,intarrive)
    servetime=1.0/srate   # service time
    simulation.nextstep(servetime)

simulation.practicalcalc()
Theoreticalmmk(srate,arate,nbrservers)
```

section2/dmc.py

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 14 18:42:28 2014

HINTS:
1) Analyse the script from bottom to top
2) Fill in the blanks marked with ???


@author: alpcan
"""

import math
import numpy as np
from wsmmchelper import *


def Theoreticalmmk(srate,arate,k):

    '''
    ??? 'You can optionally do this in Matlab, if you wish!'
    '''

    # Hint for display...
    #print 'Mean theoretical delay: {:4f} \n'.format(meandelay)
    #print 'Mean theoretical size : {:4f} \n'.format(meansize)


#parameters

maxsteps==int(1E5)       # simulation steps
srate=3                  # service rate
arate=5                  # arrrival rate
nbrservers=2             # number of servers

simulation=DESmmc(srate,arate,nbrservers,maxsteps)

for i in range(maxsteps):
    intarrive=1.0/arate  # interarrival time
    simulation.packetarrival(simulation.Q,intarrive)
    servetime=np.random.exponential(1.0/srate)  # service time
    simulation.nextstep(servetime)

simulation.practicalcalc()
Theoreticalmmk(srate,arate,nbrservers)
```

section2/ddc.py

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 14 18:42:28 2014

HINTS:
1) Analyse the script from bottom to top
2) Fill in the blanks marked with ???


@author: alpcan
"""

import math
import numpy as np
from wsmmchelper import *


def Theoreticalmmk(srate,arate,k):

    '''
    ??? 'You can optionally do this in Matlab, if you wish!'
    '''

    # Hint for display...
    #print 'Mean theoretical delay: {:4f} \n'.format(meandelay)
    #print 'Mean theoretical size : {:4f} \n'.format(meansize)


#parameters

maxsteps==int(1E5)       # simulation steps
srate=3                  # service rate
arate=5                  # arrrival rate
nbrservers=2             # number of servers

simulation=DESmmc(srate,arate,nbrservers,maxsteps)

for i in range(maxsteps):
    intarrive=1.0/arate  # interarrival time
    simulation.packetarrival(simulation.Q,intarrive)
    servetime=1.0/srate  # service time
    simulation.nextstep(servetime)

simulation.practicalcalc()
Theoreticalmmk(srate,arate,nbrservers)
```

section3/Q31.py

```
import math


######### Case 1 #########

srate=6         # service rate
arate=5         # arrrival rate

meandelay = 1/(srate-arate)
meansize = arate * meandelay

print 'Case 1'
print 'Mean theoretical delay: {:4f}'.format(meandelay)
print 'Mean theoretical size : {:4f}'.format(meansize)


######### Case 2 #########

srate=3         # service rate
arate=5         # arrrival rate

arate_new = arate * 0.5     # new arrrival rate

meandelay = 1/(srate-arate_new)
meansize = arate * meandelay

print ''
print 'Case 2'
print 'Mean theoretical delay: {:4f}'.format(meandelay)
print 'Mean theoretical size : {:4f}'.format(meansize)


######### Case 3 #########

srate=3         # service rate
arate=5         # arrrival rate
m=2             # number of servers

rho = float(arate) / (m * srate)

tmp = 0.0
for k in range(m):
    tmp += (m*rho)**k / math.factorial(k)

tmp += (m*rho)**m / math.factorial(m) / (1 - rho)
p0 = 1.0 / tmp

Nq = (m*rho)**m * rho / math.factorial(m) / (1-rho)**2 * p0
W = Nq / arate
meandelay = 1.0 / srate + W
meansize = arate * meandelay

print ''
print 'Case 3'
print 'Mean theoretical delay: {:4f}'.format(meandelay)
print 'Mean theoretical size : {:4f}'.format(meansize)
```

section3/mm1p.py

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 14 18:42:28 2014

HINTS:
1) Analyse the script from bottom to top
2) Fill in the blanks marked with ???



@author: alpcan
"""

import numpy as np
from wsmmphelper import *
#parameters

maxsteps=int(1E5)    # simulation steps
srate=3              # service rate
arate=5              # arrival rate
parallel=2           # nbr of parallel mm1's

simulation=DESmm1parallel(srate,arate,parallel,maxsteps)

for i in range(maxsteps):
    intarrive=np.random.exponential(1.0/arate)  # interarrival time
    simulation.packetarrival(intarrive)
    servetime=np.random.exponential(1.0/srate)  # service time
    simulation.nextstep(servetime)

simulation.practicalcalc(parallel)
```