# test_queueing_model

October 20, 2023

## 0.1 Modelling flow through the homeless services system with an M(t)/M/s(t) queue

The DES model of flow through the homeless services system (Singham et al. (2023) illustrates the effects of different investment policies (building a mix of temporary shelter and permanent accommodation) on the homeless population involved. A key metric for measuring the success of a policy is the number of homeless people who remain unsheltered throughout the modelled 6 year time window. There are two "versions" of this model - a general one with a homogeneous population and homogeneous accommodation and a more detailed one where the population is categorised in terms of specific housing need, the accommodation to meet that need is also categorised and different categories of the population have different pathways through the system.

Here we look specifically at the general version of the model and we attempt to provide an alternative way of modelling the system using an exact analytical model of approximation of the DES model. The motivation for this is two-fold. Firstly, two different models of the same system can add extra insight on and understanding of the underlying system. Secondly, a future simulation optimisation approach may well benefit from an analytical model to guide the search of the simulation solution space.

### 0.1.1 The analytical model

The analytical model we use is an M(t)/M/s(t) queue with the following features:

- The time-dependent Poisson arrival process is a direct equivalent of the arrival process used in a modified version of the DES model (modifications discussed later in this document).
- The exponential distribution we use to model service time (time spent in permanent accommodation) is a direct equivalent of exponential service time distribution used in the modified DES model. The exponential distribution is favourable for the analytical model because it enables the use of numerical integration to analyse the dynamics of the system, because the exponential dist is memoryless, we can model the system as a discrete time Markov chain.
- The time-dependent number of housing servers (i.e. number permanent accommodation units) is a direct equivalent of the time-dependent number of housing units in the DES model.
- The number of shelters (which is directly modelled in the DES model) is not directly modelled in the analytical model, but it is incorporated in the analysis of the dynamics of the queue - the queue for the permanent accommodation at a given time can be "split out" into the sheltered and the unsheltered population, where the point in the queue where the split is made is time dependent.

The state space is the number of people in the system, and we compute the probability of being in each state at any point in discretised time by using numerical integration. using this method,

within sufficiently small intervals, we can assume the arrival rate is constant and also assume that at most one state change can occur in that time interval. Therefore, if we know the probabilities $p_n(t)$ of being in all states $n \in \{0, 1, ..., N-1, N\}$ at some time $t \in \{0, 1, ..., T-1, T\}$, then at time $t + d$, where $d$ is the small time-interval, we can give the probability $p_n(t + d)$ of being in some state $n$ as:

$$p_n(t + d) = p_{n-1}(t)[\lambda(t)d] + p_n(t)[1 - \lambda(t)d - \mu(t)m_n d] + p_{n+1}(t)[\mu(t)m_{n+1}d]$$

where $\lambda(t)$ is the arrival rate at time $t$, $\mu(t)$ is the service rate at time $t$ and $m_n$ is the number of busy servers when the system is in state $n$.

Given an initial number of people in the system, and a fixed maximum number of people in the system which we set to be very large, we can iteratively compute all state probabilites in the time interval $\{d, 2d, ..., T\}$. These probabilities can then be analysed to determine the following time-dependent metrics:

- the expected number of people in the system,
- the expected number of people in the queue (both sheltered and non-sheltered),
- the expected number of unsheltered people.

The python module accompanying this notebook, called "queueing_model.py" contains the code required (wrapped up in a "queue" class) to model such a queue and calculate the desired metrics. It also contains contains a function "mms_steadystate($\lambda$, $s$, $\mu$)" which gives the steady state results for an M/M/s queue and it includes various other auxiliary functions for including plotting functions.

[5]:
```
import queueing_model as qm
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import math
from scipy import stats
```

### 0.1.2 Steady state behaviour of M/M/s queue

First, it is helpful to test our queueing model against the known steady state behaviour of a time-homogeneous queue. Steady state behaviour is only observed when the arrival rate is less than the service rate multiplied by the number of servers.

As an example, we use the following parameters:

[6]:
```
# steady state parameters for testing
arrival_rate = 35
service_rate = 12
n_servers = 4
print("Arrival rate:", arrival_rate, "people enter system per year")
print("Service rate:", service_rate, "people exit each accommodation unit per␣
 ↪year")
print("Number of accommodation units", n_servers, "units")
```

```
Arrival rate: 35 people enter system per year
Service rate: 12 people exit each accommodation unit per year
```

```
Number of accommodation units 4 units
```

The known steady-state behaviour (number in the system, number in the queue) is computed below:

[7]:
```
n_sys, n_queue = qm.mms_steadystate(arrival_rate, n_servers, service_rate)
print("Steady state number in the system: ", round(n_sys,3))
print("Steady state number in queue (sheltered + unsheltered): ",␣
 ↪round(n_queue,3))
```

```
Steady state number in the system:  4.196
Steady state number in queue (sheltered + unsheltered):  1.279
```

Now, we check that our queueing model (analysed using numerical integration, as described above) is working correctly by using it to model the dynamics of a queue with the same steady state set up for arrival rate, service rate and the number of servers.

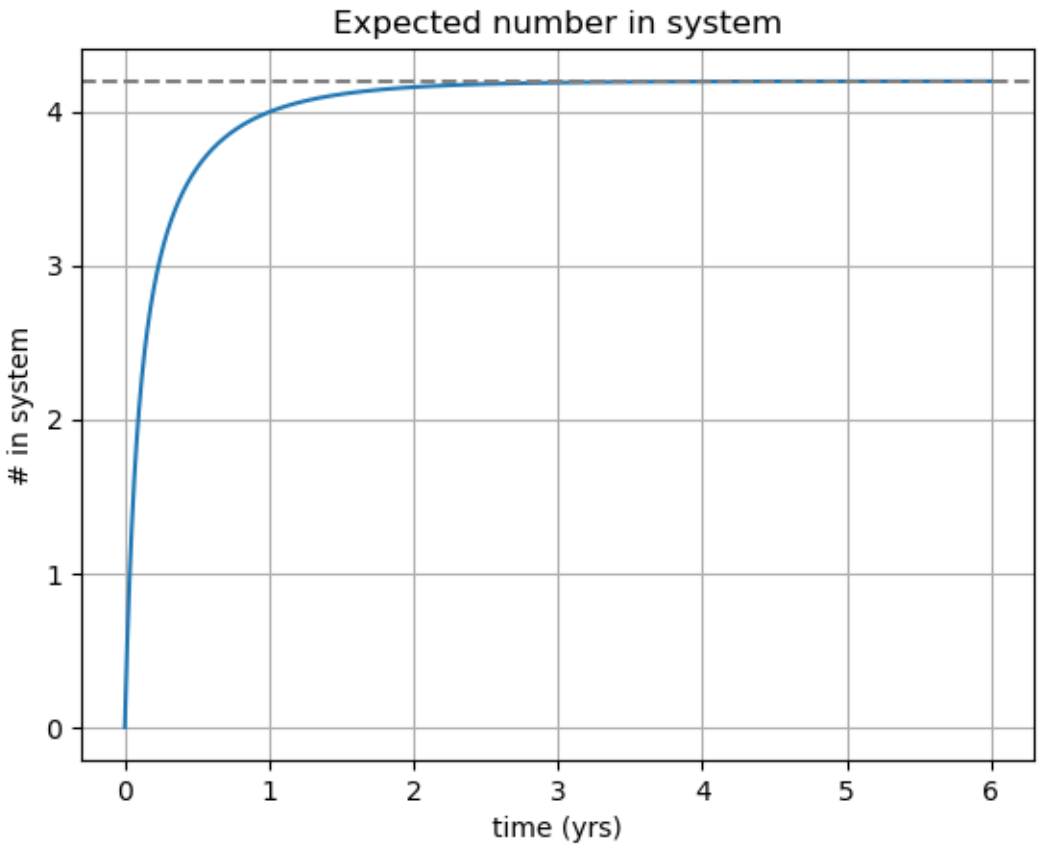
[8]:
```
# parameters to model queue (MMs steady state)
annual_arrival_rate = [35, 35, 35, 35, 35, 35] # six entries, one for each year␣
 ↪over 6 years.
mean_service_time = 1/12 # per year
servers_initial = 4
shelter_initial= 0
server_build_rate = [0,0,0,0,0,0] # six entries, one for each year over 6 years.
 ↪
shelter_build_rate = [0,0,0,0,0,0] # six entries, one for each year over 6␣
 ↪years.
num_in_system_initial = 0
max_in_system = 100
num_annual_build_points = 6
build_frequency_weeks = 9
```

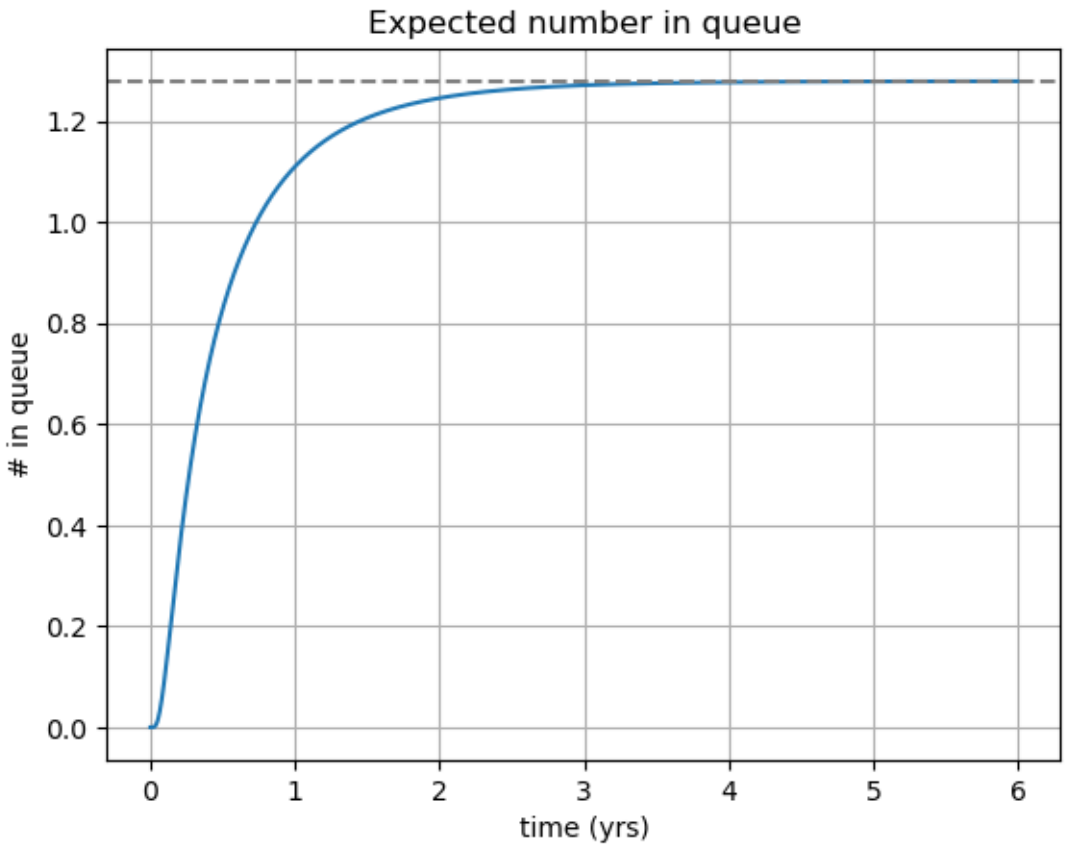
[9]:
```
# create instance of queue
q = qm.queue(annual_arrival_rate,
             mean_service_time,
             servers_initial,
             shelter_initial,
             server_build_rate,
             shelter_build_rate,
             num_in_system_initial,
             max_in_system,
             num_annual_build_points,
             build_frequency_weeks)

# model dynamics of the queue (this updates several attributes of q)
q.model_dynamics(Y=6,d=1) # Y : time horizon for analysis in integer years, d:␣
 ↪width of time step in days.
```

The plots below show the expected number in the system and number in the queue, over a six year time window. The dashed lines indicate the known values as previously calculated. These plots

illustrate that our numerical integration is converging to the correct solution which gives us some trust that the model is behaving properly.

[10]:
```
# plot queue dynamics - num in system
fig, ax = plt.subplots()
ax.plot(np.arange(0,6,1/365), q.num_sys)
plt.axhline(y = n_sys, color = 'gray', linestyle = 'dashed')

ax.set(xlabel='time (yrs)', ylabel='# in system',
       title='Expected number in system')
ax.grid()

plt.show()

# plot queue dynamics - number in queue
fig, ax = plt.subplots()
ax.plot(np.arange(0,6,1/365), q.num_queue)
plt.axhline(y = n_queue, color = 'gray', linestyle = 'dashed')

ax.set(xlabel='time (yrs)', ylabel='# in queue',
       title='Expected number in queue')
ax.grid()

plt.show()
```


Expected number in system
# in system
time (yrs)
0
1
2
3
4
5
6




### 0.1.3 Time dependent behaviour of M(t)/M/s(t) queue

We now make an instance of a queue where we have copied parameters directly from the Simio DES model of Singham et al. (2023). It should be noted that we have assumed re-entries to the system come at a steady rate throughout the time horizon modelled. We have set that rate to correspond with the service rate from the initial number of accommodation units.

[11]:
```
# model queue (time dependent homeless services system)
servers_initial = 40
shelter_initial = 15
server_build_rate = [3,6,7,10,8,4] # number of servers to be added at start of␣
 ↪every two months incl month 0 (different value in each of six years)
shelter_build_rate = [2,2,0,-2,-1,-1] # number of shelters to be added at start␣
 ↪of every two months incl month 0 (different value in each of six years)
mean_service_time = (1/52)*(0+300+400)/3 # mean (years) of the␣
 ↪triangular(0,300week,400week) distribution
annual_arrival_rate = [35.0400, 42.0048, 46.2528, 46.2528, 41.6100, 37.4052] #␣
 ↪six entries, one for each year over 6 years.
reentry_rate = 0.17 # the proportion of those leaving accommodation which␣
 ↪re-enter the system some time later
```

```
annual_arrival_rate_reentries = (servers_initial*reentry_rate)/
 ↪mean_service_time # assuming re-entries from the initial number of servers
annual_arrival_rate = [i+annual_arrival_rate_reentries for i in␣
 ↪annual_arrival_rate]
num_in_system_initial = 120
max_in_system = 1000
num_annual_build_points = 6
build_frequency_weeks = 9
```

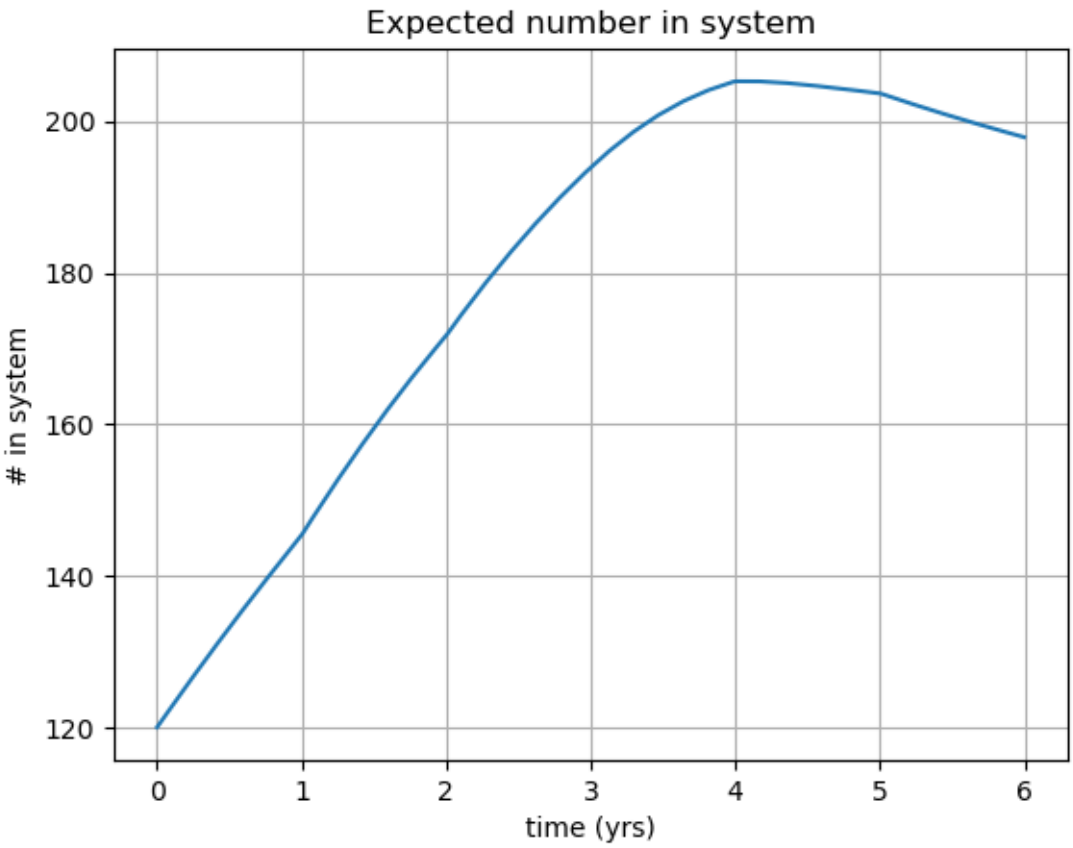
[12]:
```
print('The arrival rates, including re-entries, result in the following annual␣
 ↪and hourly arrival rates')
print('------------------------')

yr = 1
for rate in annual_arrival_rate:
    print('Year ' + str(yr) + ': annual arrival rate of ' + str(rate))
    yr+=1

print('------------------------')

yr = 1
for rate in annual_arrival_rate:
    print('Year ' + str(yr) + ': hourly arrival rate of ' + str(rate/8760))
    yr+=1

print('------------------------')
print('The mean service time in weeks at each accommodation unit is: ' +␣
 ↪str(mean_service_time*52))
```

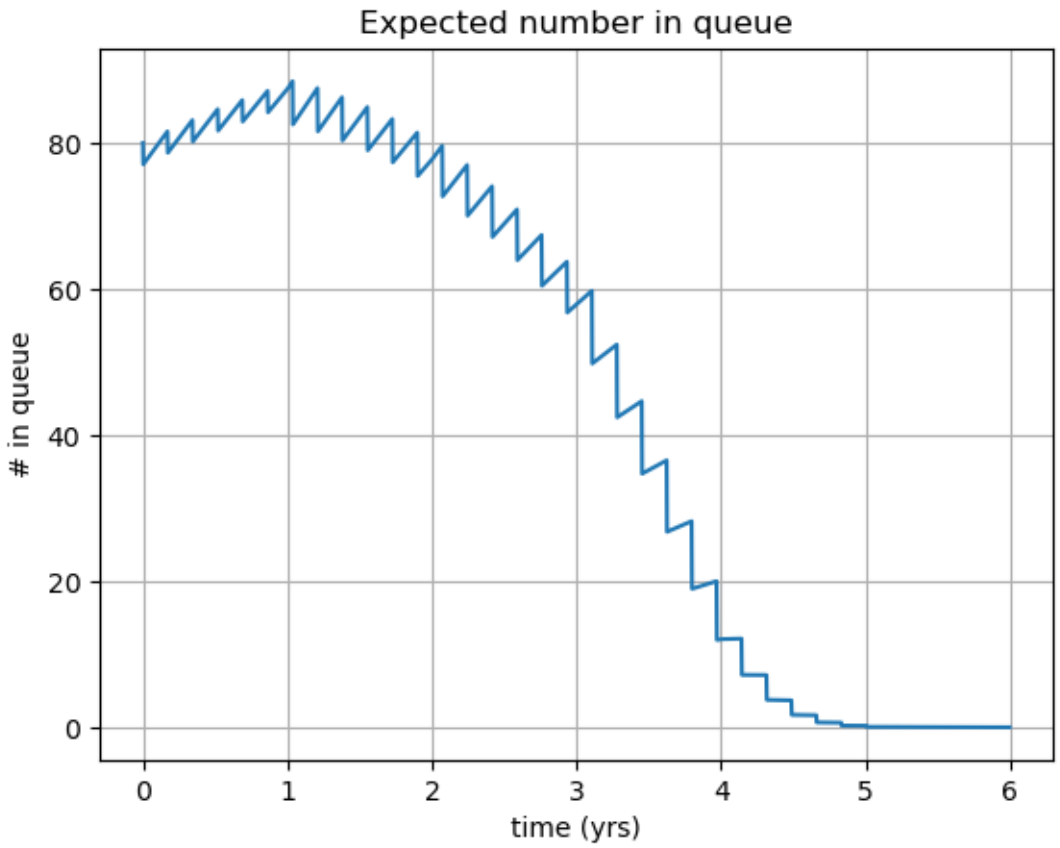
```
The arrival rates, including re-entries, result in the following annual and
hourly arrival rates
------------------------
Year 1: annual arrival rate of 36.55542857142857
Year 2: annual arrival rate of 43.520228571428575
Year 3: annual arrival rate of 47.76822857142857
Year 4: annual arrival rate of 47.76822857142857
Year 5: annual arrival rate of 43.12542857142857
Year 6: annual arrival rate of 38.92062857142857
------------------------
Year 1: hourly arrival rate of 0.004172994129158513
Year 2: hourly arrival rate of 0.004968062622309198
Year 3: hourly arrival rate of 0.005452994129158513
Year 4: hourly arrival rate of 0.005452994129158513
Year 5: hourly arrival rate of 0.004922994129158513
Year 6: hourly arrival rate of 0.0044429941291585125
------------------------
The mean service time in weeks at each accommodation unit is: 233.33333333333334
```

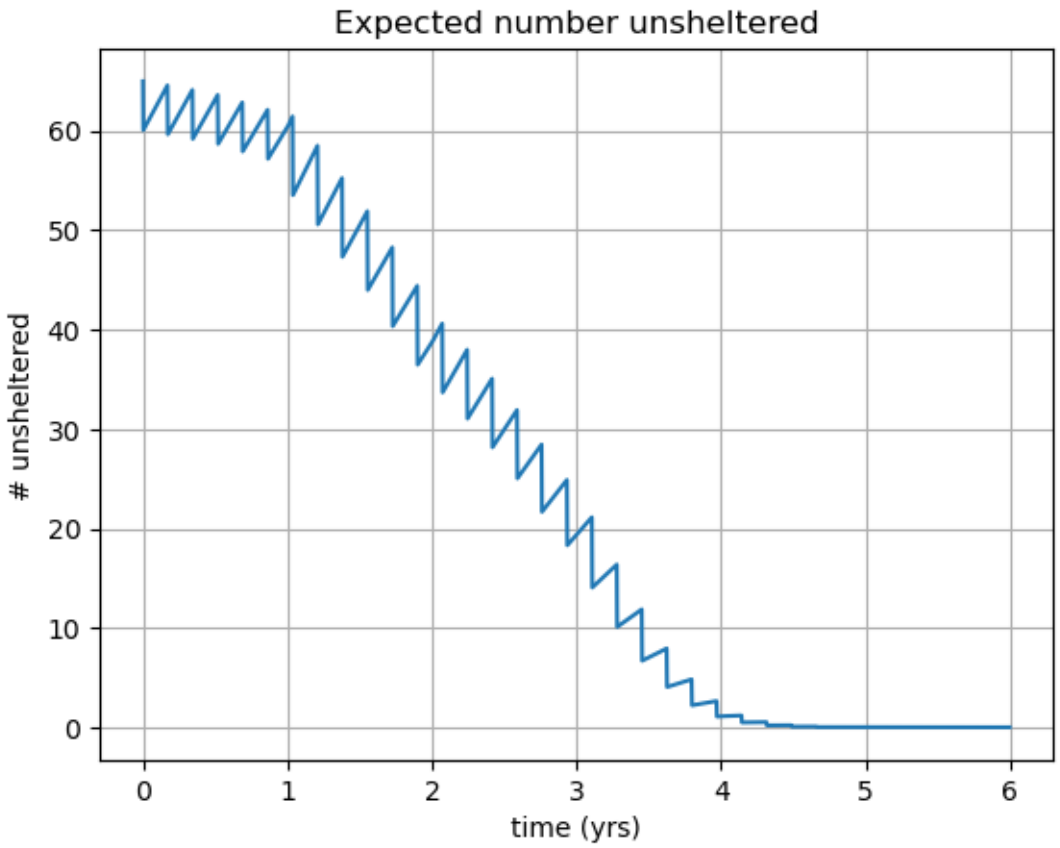
[13]:
```
q = qm.queue(annual_arrival_rate,
             mean_service_time,
             servers_initial,
             shelter_initial,
             server_build_rate,
             shelter_build_rate,
             num_in_system_initial,
             max_in_system,
             num_annual_build_points,
             build_frequency_weeks)
q.model_dynamics(Y=6,d=1) # Y : time horizon for analysis in integer years, d:␣
 ↪width of time step in days.
```

The plots below show the expected number in the system, the expected number in the queue (sheltered and unsheltered) and the expect number of people unsheltered.

[14]:
```
# plot queue dynamics - num in system
fig, ax = plt.subplots()
ax.plot(np.arange(0,6,1/365), q.num_sys)

ax.set(xlabel='time (yrs)', ylabel='# in system',
       title='Expected number in system')
ax.grid()

plt.show()

# plot queue dynamics - number in queue
fig, ax = plt.subplots()
ax.plot(np.arange(0,6,1/365), q.num_queue)

ax.set(xlabel='time (yrs)', ylabel='# in queue',
       title='Expected number in queue')
ax.grid()

plt.show()

# plot queue dynamics - number unsheltered
fig, ax = plt.subplots()
ax.plot(np.arange(0,6,1/365), q.num_unsheltered)

ax.set(xlabel='time (yrs)', ylabel='# unsheltered',
       title='Expected number unsheltered')
ax.grid()

plt.show()
```


Expected number in system
200
180
160
140
120
# in system
0
1
2
3
4
5
6
time (yrs)


Expected number in queue
# in queue
time (yrs)
0
20
40
60
80
0
1
2
3
4
5
6




The expected number of people unsheltered drops every two months as new shetler and accommodation is built. The general trajectory is initially downward as the extra shelter and accommodation built every two months outweighs the growth of the queue in the two month time period. During the fifth year, there is no longer an unsheltered population.

### 0.1.4 Comparison with DES model

To compare these model outputs with the DES model, we make minor modifications to V1 of the simple model in Simio to make it a direct equivalent to the analytical queueing model. The changes are summarised below:

- The service time distribution of the DES model is changed to be exponential, with the same mean as the original triangular distribution.
- The DES pathway leaving permanent accommodation and returning to queue for shelter (originally applying to on average 17% of those leaving permanent accommodation) is removed as it is not possible to model this specific behaviour in the analyical queueing model.
- To account for this removed pathway, the time-dependent Poisson arrival process in the DES model is increased at all times by a constant factor to account for 17% of people leaving 40 units of permanent accommodation (given the service time of those units) re-entering the system from the start. The arrival rates are now identical in the DES model and the analytical

model.

We perform 1000 replications of this DES model and monitor the number of unsheltered people at 9 week intervals. The data is collected every 9 weeks, momentarily before new shelter / accommodation is added to the system and occupied. We collect the equivalent data from the queueing model, i.e. just before new shelter/accommodation is added.

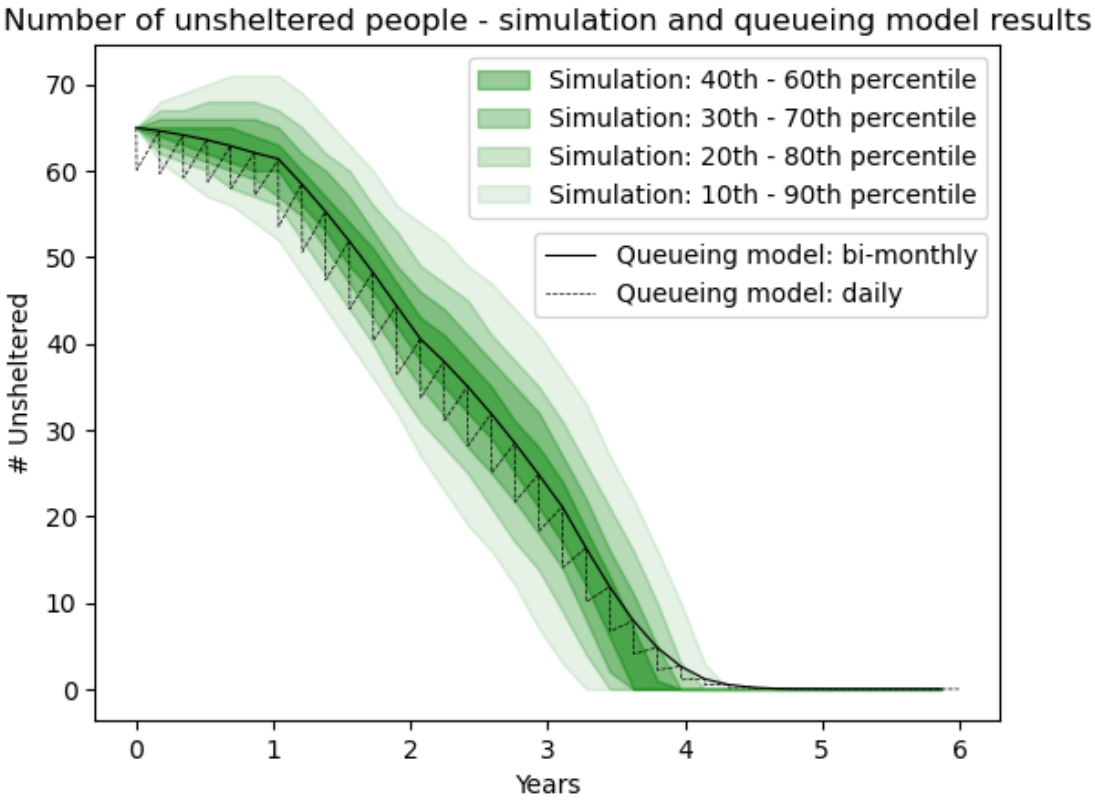
[16]:
```
# read simulation data from file
data_sim = []
unsheltered_initial = num_in_system_initial - servers_initial - shelter_initial␣
 ↪# to add to the first row of each data file

for i in range(1000):
    run = i+1
    file_name = 'data/simio_outputs/
 ↪2023-08-11-SimpleModelAC_IncV2_no_absorbing_state/
 ↪SimOutput_Experiment1_Scenario 4_Rep' + str(run) + '.xlsx'
    sim_out = pd.read_excel(file_name)
    sim_out.columns = ['housing_capacity', 'shelter_capacity',␣
 ↪'housing_occupied', 'shelter_occupied', 'num_unsheltered',␣
 ↪'last_shetler_time']
    data_sim.append(np.array(pd.concat([pd.Series([unsheltered_initial]),␣
 ↪sim_out['num_unsheltered'][1:]])))

data_sim = np.array(data_sim).T
```

[17]:
```
# get the queue data we need (i.e. the data just before new shelter/
 ↪accommodation is filled every 9 weeks)
index_list = [0]
day = 1
two_months_in_days = 63
num_two_months = 0
sim_length = data_sim.shape[0]-1 # length in units of two months

while num_two_months < sim_length:
    if math.floor((day+1)/two_months_in_days) > math.floor((day)/
 ↪two_months_in_days):
        index_list.append(day)
        num_two_months += 1
    day += 1

data_q = [q.num_unsheltered[i] for i in index_list]
```

Below we overlay the spread of the simulation results with the analytical results from the queueing model. From the simulation data we display four different percentile ranges for the number of unsheltered people. From the queueing model we display the daily expected value of the number of homeless people with a thin dotted line and the bi-monthly expected values (i.e. the expected values at the times just before new shelter / accommodation is added as opposed to an average over the

two month periods) with a thicker solid line. The latter bimonthly data from the queueing model is equivalent to the simulation data, and we can see that until time $t \approx 3.5$ years, the expected value of the number of unsheltered people from the queueing model sits between the 40th and 60th percentile of this quantity from the simulation model, indicating good agreement between the two models. Between time $t \approx 3.5$ years and $t \approx 4$ years, the simulation model shows a quicker fall to an empty queue of unsheltered people. It is not yet known why we see this difference. It was suspected that this was due to the fact that in the original setup of the simulation, on any given simulation run, as soon as the number of unsheltered people dropped to zero, no more events werre scheduled and this number remained at zero, whereas the queueing model allowed for the possibility of new unsheltered people after a state of zero unsheltered people. But we have now adjusted the simulation model so that there is no absorbing state at any time in the simulation, and we still see this difference between time $t \approx 3.5$ years and $t \approx 4$ years.

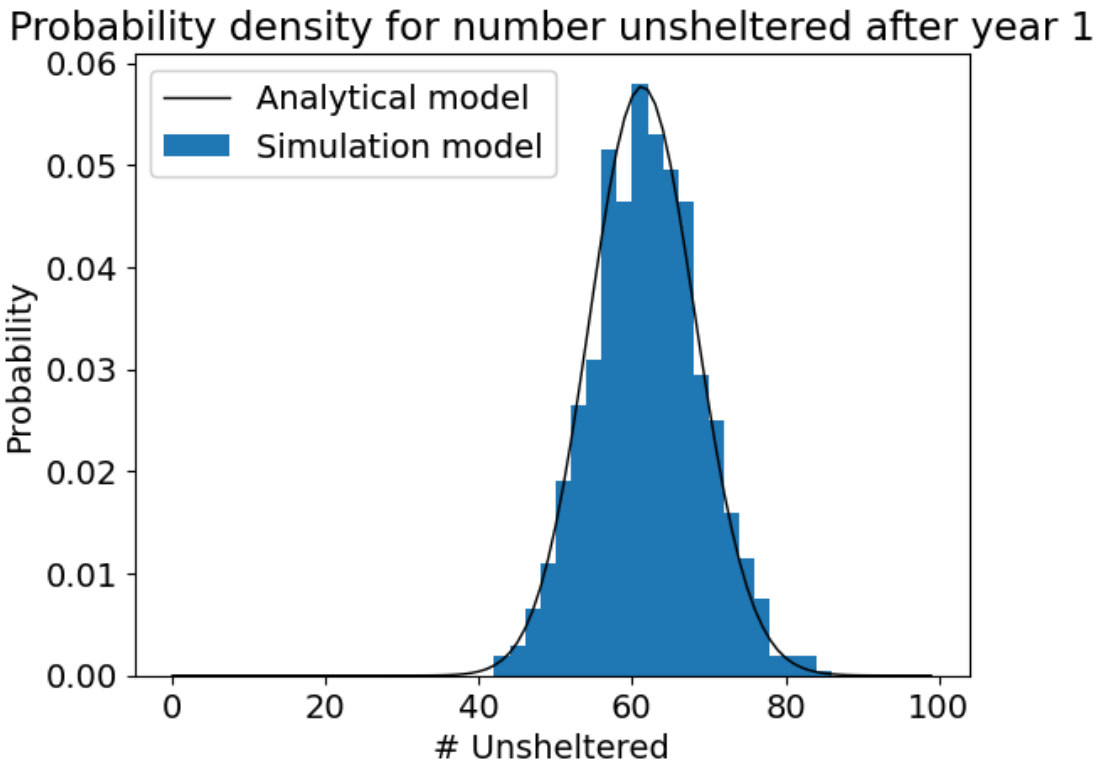
[18]:
```
# create fan chart
fig, ax = qm.create_fanchart(data_sim, data_q, q)
```



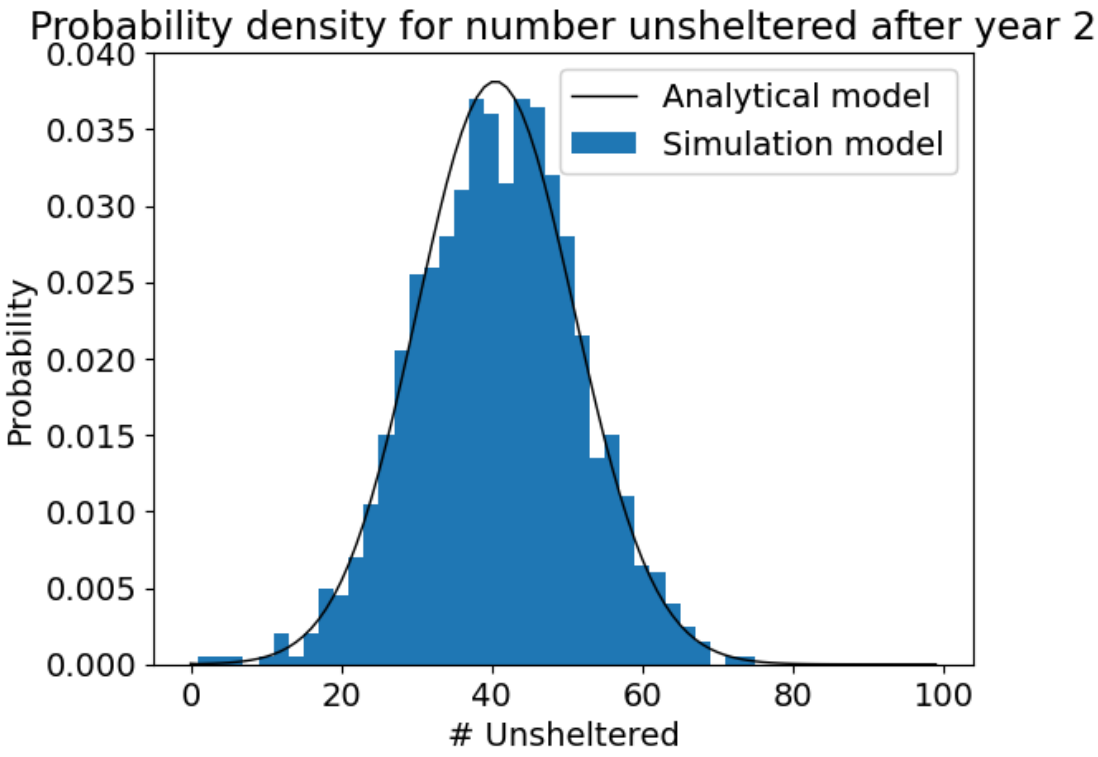

### 0.1.5 Further comparison with DES model

In the three charts below we take snapshots after years 1, 2 and 3 respectively. At each snapshot, we look at the PDF given by the analytical model for the number of unsheltered people (black solid line) alongisde the empirical PDF from the simulation model (in the form of a histogram).

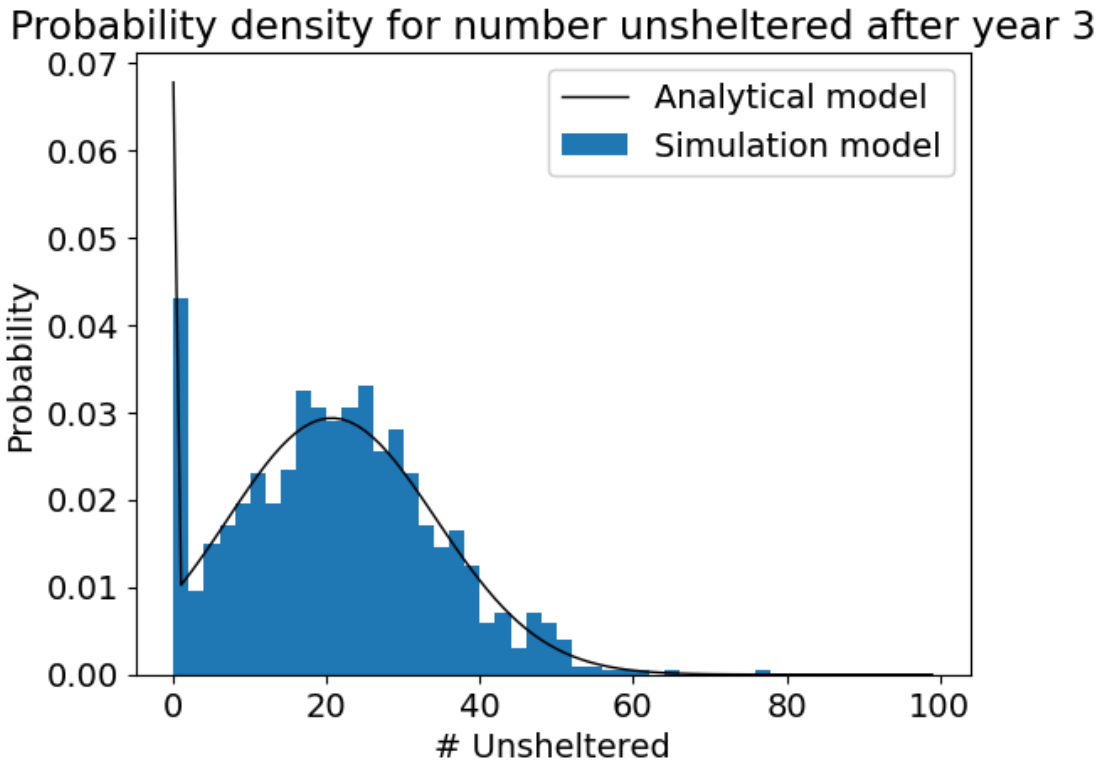
[19]:
```
font = {'size'   : 14}
plt.rc('font', **font)
fig, ax = qm.create_chart_prob_dists(data_sim,q,t=1,binwidth=2)
```




[20]:
```
fig, ax = qm.create_chart_prob_dists(data_sim,q,t=2,binwidth=2)
```

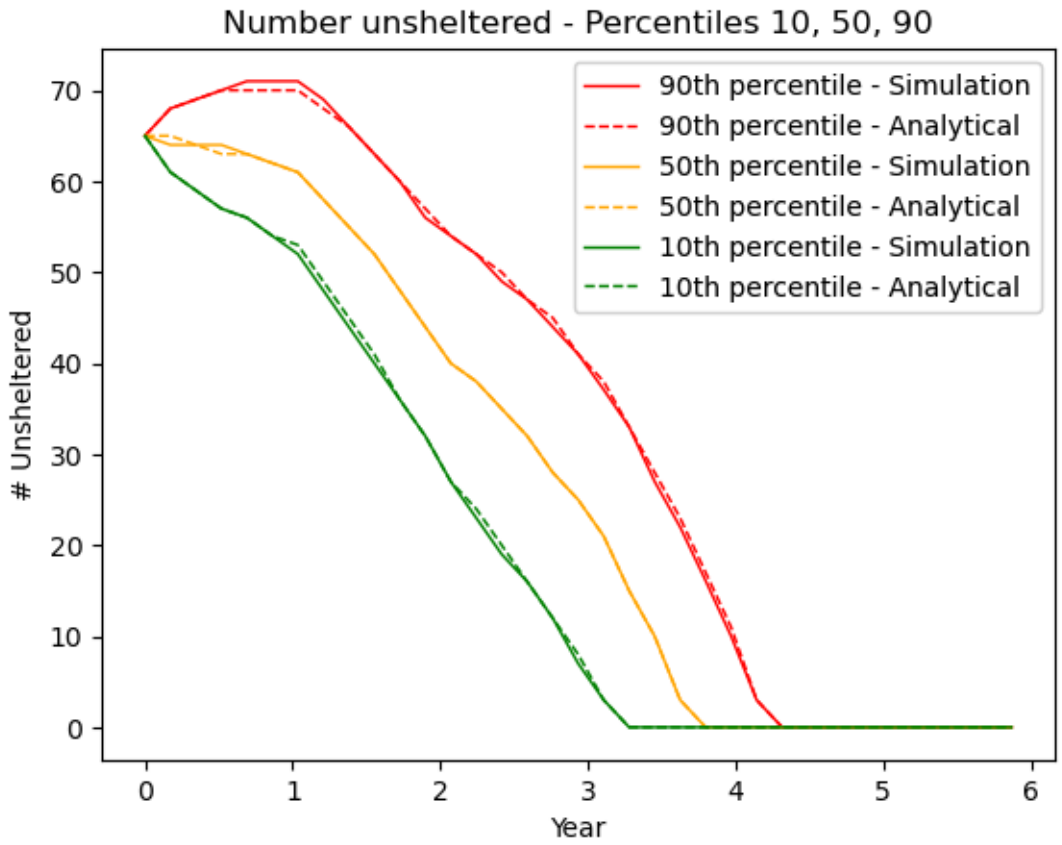



[21]:
```
fig, ax = qm.create_chart_prob_dists(data_sim,q,t=3,binwidth=2)
```




A visual comparison between the outputs from the analytical model and those from simulation model, at each of the three points in time, suggests that the shapes of the two PDFs are very similar. The similarities are also visualised in a plot of the three percentiles (10th, 50th, 90th) over time for both the analytical model (in dashed lines in the chart below) and the simulation model (in solid lines below).

[22]:
```
font = {'size'   : 10}
plt.rc('font', **font)
fig, ax = qm.
  ↪create_chart_comparing_percentiles(data_sim,q,percentiles=[10,50,90],␣
  ↪index_list=index_list)
```




A statistical tool for comparing the above distributions is the Kolmogorov Smirnoff (K-S) test which compares a sample from an unknown distribution (in our case the empirical distribution from the simulation model) with a known analytic distribution (in our case the analytic queueing model). Below we do three separate K-S tests with the simulation data at three different points in time (after 1, 2 and 3 yrs of simulated time)

[23]:
```
# yr 1
sample_data = data_sim[6] # simulation data after 6 9 week periods (1 yr)
print('Year 1 comparison:')
print(stats.kstest(sample_data, qm.analytic_cdf, args = (q,100,1)))

# yr 2
sample_data = data_sim[12] # simulation data after 12 9 week periods (2yrs)
print('------------------')
print('Year 2 comparison:')
print(stats.kstest(sample_data, qm.analytic_cdf, args = (q,100,2)))

# yr 3
sample_data = data_sim[18] # simulation data after 18 9 week periods (3yrs)
print('------------------')
```

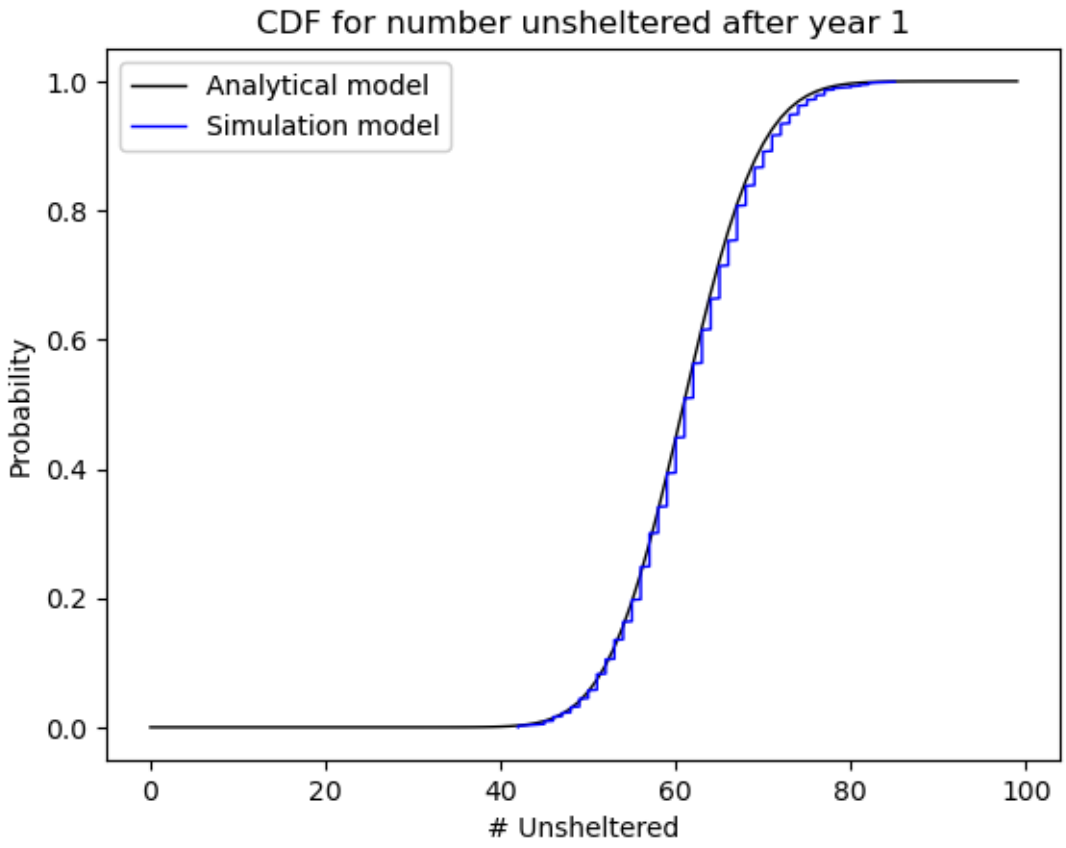
```
print('Year 3 comparison:')
print(stats.kstest(sample_data, qm.analytic_cdf, args = (q,100,3)))
```

```
Year 1 comparison:
KstestResult(statistic=0.05866267343976039, pvalue=0.001964812103114551,
statistic_location=65, statistic_sign=-1)
------------------
Year 2 comparison:
KstestResult(statistic=0.042308781787174454, pvalue=0.05416145337081235,
statistic_location=44, statistic_sign=-1)
------------------
Year 3 comparison:
KstestResult(statistic=0.06773550985796457, pvalue=0.00019635162630691317,
statistic_location=0, statistic_sign=-1)
```

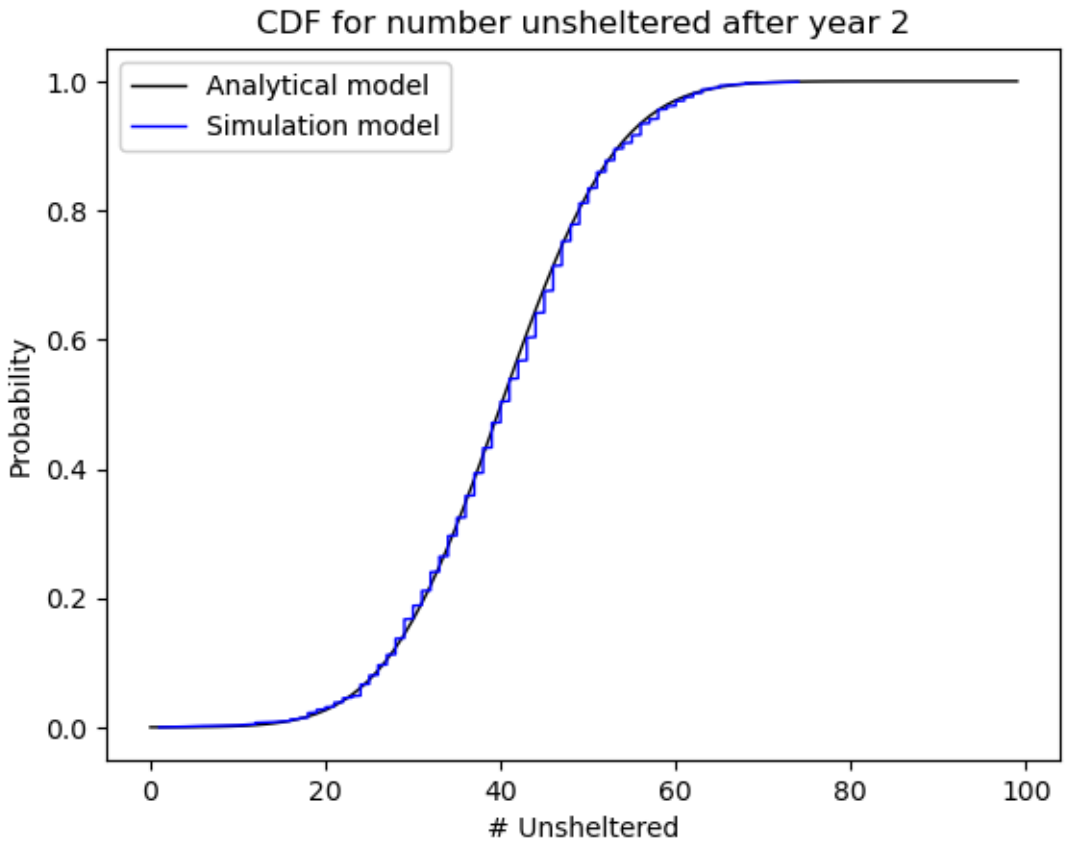
The outputs of the above K-S tests indicate that after one year and after three years, given the p-values are less than 0.05, we reject the null hypothesis that the simulated data is distributed as the analytical queue CDF, suggesting that there is a difference between the two distributions. After two years, the p-value is just over 0.05 and we therefore we cannot reject the null hypothesis, which suggests there is no difference between the distributions at this point in time.

The small differences (and similarities) between the CDFs can be visualised in the plots below.

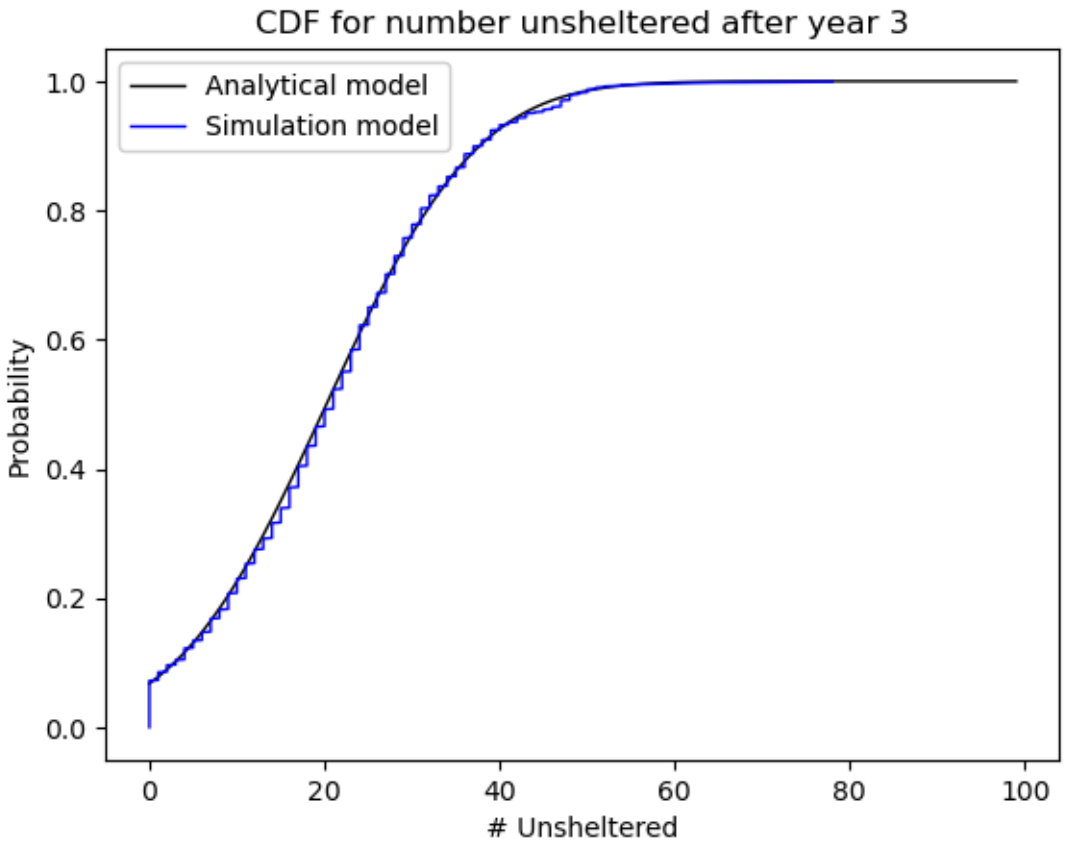
[24]:
```
fig, ax = qm.compare_cdf(data_sim, q, 100, 1)
```




[25]:
```
fig, ax = qm.compare_cdf(data_sim, q, 100, 2)
```




[26]:
```
fig, ax = qm.compare_cdf(data_sim, q, 100, 3)
```


CDF for number unsheltered after year 3
Analytical model
Simulation model
Probability
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
# Unsheltered