Lesson #12: More on Birth Death Simulations and M/G, G/M systems.

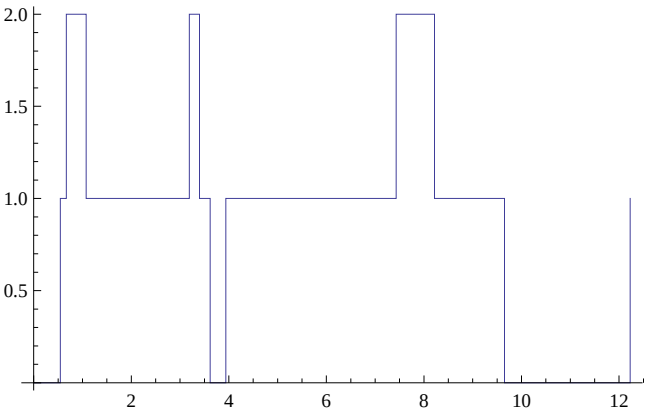
From Previous Lesson :

In[25]:=
```
bdSim[birthFunction_, deathFunction_, timeHorizon_] :=
 Module[{state = 0, time = 0.0, index = 1},
  rel = Table[{kuku, bobo},
    {Round[4 * 2 * (birthFunction[1] + deathFunction[1]) * timeHorizon]}];
  (*rel={{time,state}};*)
  rel[[index++]] = {time, state};
  (*Now loop untill time > timeHorizon and append to rel as we go*)
  While[time < timeHorizon,
   birthRate = birthFunction[state];
   deathRate = deathFunction[state];
   time += RandomReal[ExponentialDistribution[birthRate + deathRate]];
   rel[[index++]] = {time, state};
   (*AppendTo[rel,{time,state}];*)
   state += 2 RandomInteger[BernoulliDistribution[birthRate/(birthRate + deathRate)]] - 1;
   rel[[index++]] = {time, state};
   (*AppendTo[rel,{time,state}];*)
  ];
  Take[rel, index - 1]
 ]
```

■ M/M/1

In[26]:=
```
λ = 0.4;
μ = 1.0;
mm1Birth[s_] := λ;
mm1Death[s_] := If[s > 0, μ, 0];
rel = bdSim[mm1Birth, mm1Death, 10];
ListPlot[rel, Joined → True]
```

Out[31]=



Different Timing Attempts

```
Table[{t, First[Timing[bdSim[mm1Birth, mm1Death, t];]]}, {t, 500, 5000, 500}]
```

```
{{500, 0.312}, {1000, 1.438}, {1500, 3.562}, {2000, 6.282}, {2500, 10.39},
 {3000, 15.641}, {3500, 21.109}, {4000, 24.782}, {4500, 33.921}, {5000, 36.688}}
```

```
Table[{t, First[Timing[bdSim[mm1Birth, mm1Death, t];]]}, {t, 500, 5000, 500}]
```

```
{{500, 0.188}, {1000, 0.781}, {1500, 2.187}, {2000, 4.813}, {2500, 8.656},
 {3000, 12.672}, {3500, 18.813}, {4000, 26.}, {4500, 32.531}, {5000, 38.172}}
```

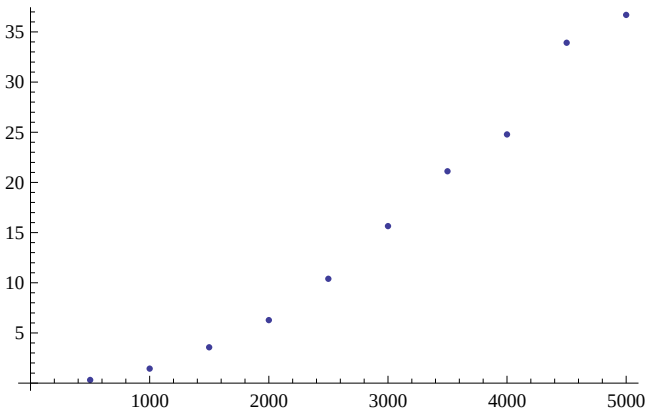
```
Table[{t, First[Timing[bdSim[mm1Birth, mm1Death, t];]]}, {t, 500, 5000, 500}]
```

```
{{500, 0.125}, {1000, 0.219}, {1500, 0.328}, {2000, 0.422}, {2500, 0.547},
 {3000, 0.672}, {3500, 0.765}, {4000, 0.907}, {4500, 0.968}, {5000, 1.11}}
```

```
Table[{t, First[Timing[bdSim[mm1Birth, mm1Death, t];]]}, {t, 500, 5000, 500}]
```

```
{{500, 0.141}, {1000, 0.547}, {1500, 1.25}, {2000, 1.859}, {2500, 3.063},
 {3000, 4.531}, {3500, 6.719}, {4000, 8.875}, {4500, 11.234}, {5000, 13.344}}
```

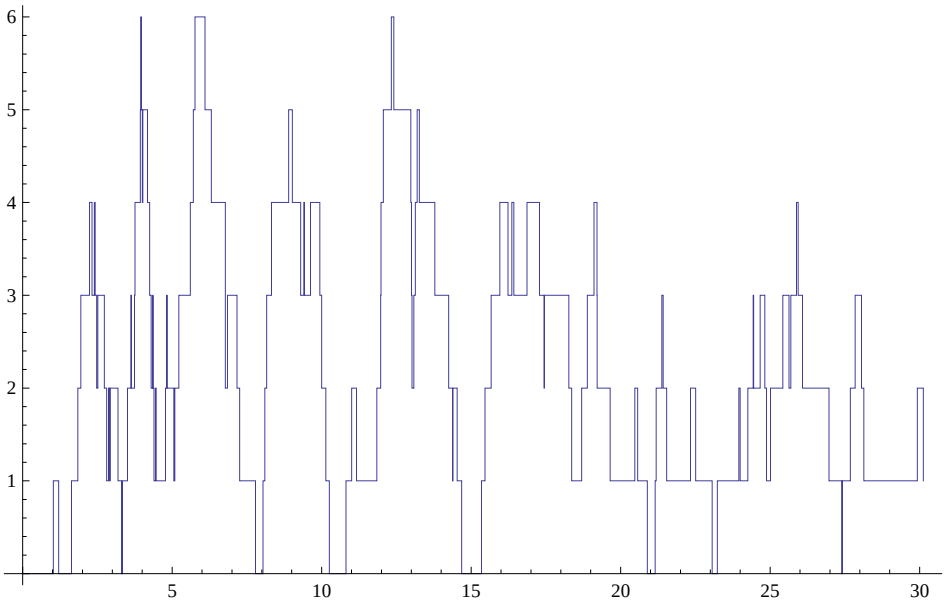
```
ListPlot[%]
```



M/M/K/K (Erlang Loss System)

In[32]:=
```
λ = 2.0;
μ = 1.0;
KK = 8;
mm1Birth[s_] := If[s < KK, λ, 0];
mm1Death[s_] := μ s;
rel = bdSim[mm1Birth, mm1Death, 30];
ListPlot[rel, Joined → True]
```

Out[38]=



Joint Class Exercise

1) Collect Stationary Distribution from Simulation

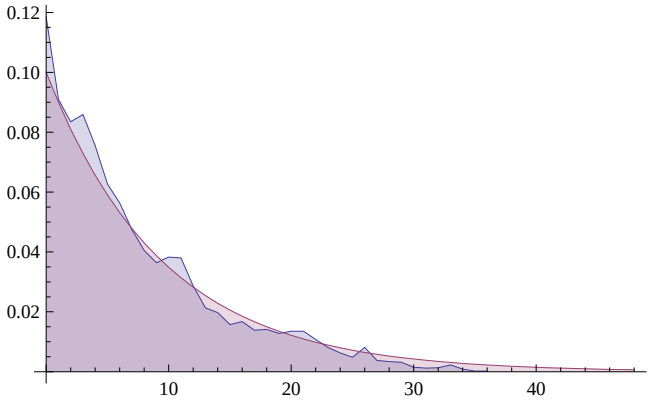
In[39]:=
```
bdSimWithStat[birthFunction_, deathFunction_, timeHorizon_] :=
 Module[{state = 0, time = 0.0},
  rel = {{time, state}};
  histogramTable = Table[0.0, {100}];
  (*Now loop untill time > timeHorizon and append to rel as we go*)
  While[time < timeHorizon,
   birthRate = birthFunction[state];
   deathRate = deathFunction[state];
   stateDuration = RandomReal[ExponentialDistribution[birthRate + deathRate]];
   histogramTable[[state + 1]] += stateDuration;
   time += stateDuration;
   AppendTo[rel, {time, state}];
   state += 2 RandomInteger[BernoulliDistribution[birthRate/(birthRate + deathRate)]] - 1;
   AppendTo[rel, {time, state}];
  ];
  {rel, histogramTable/timeHorizon}
 ]
```

M/M/1 Histogram

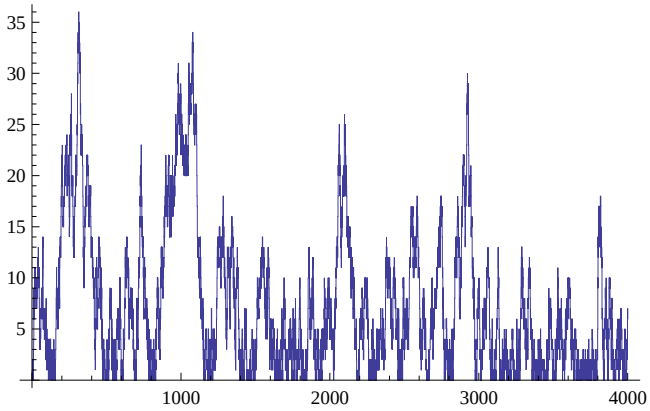
In[40]:=
```
λ = 0.9;
μ = 1.0;
mm1Birth[s_] := λ;
mm1Death[s_] := If[s > 0, μ, 0];
out = bdSimWithStat[mm1Birth, mm1Death, 4000];
hist = Select[Transpose[{Table[i, {i, 0, Length[out[[2]]] - 1}], out[[2]]}], #[[2]] > 0 &];
theoreticalHist = Table[{i, (1 - λ/μ) (λ/μ)^i}, {i, 0, Round[1.3 * Length[hist]]}];
```

In[47]:=
```
ListPlot[{hist, theoreticalHist}, Filling → Axis,
 Joined → True, PlotRange → All, PlotStyle → PointSize[0.02]]
ListPlot[out[[1]], Joined → True]
```

Out[47]=



Out[48]=

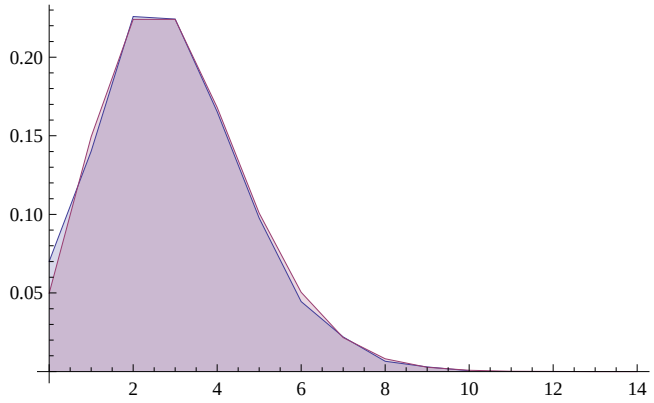


■ M/M/∞ Histogram

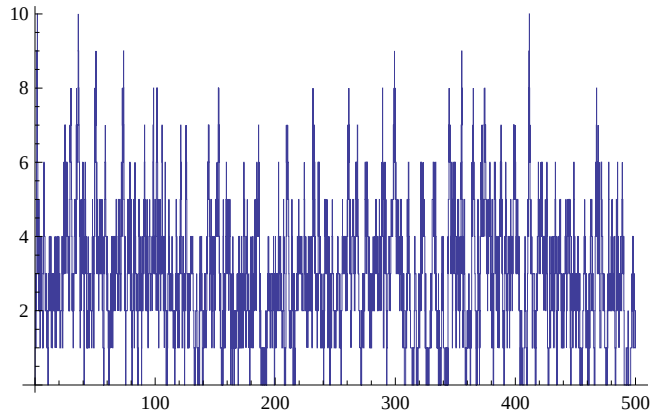
In[49]:=
```
λ = 3.0;
μ = 1.0;
mm1Birth[s_] := λ;
mm1Death[s_] := μ s;
out = bdSimWithStat[mm1Birth, mm1Death, 500];
hist = Select[Transpose[{Table[i, {i, 0, Length[out[[2]]] - 1}], out[[2]]}], #[[2]] > 0 &];
theoreticalHist = Table[{i, E^(-λ/μ) (λ/μ)^i / i!}, {i, 0, Round[1.3 * Length[hist]]}];
```

In[56]:=
```
ListPlot[{hist, theoreticalHist}, Filling → Axis,
 Joined → True, PlotRange → All, PlotStyle → PointSize[0.02]]
ListPlot[out[[1]], Joined → True]
```

Out[56]=



Out[57]=

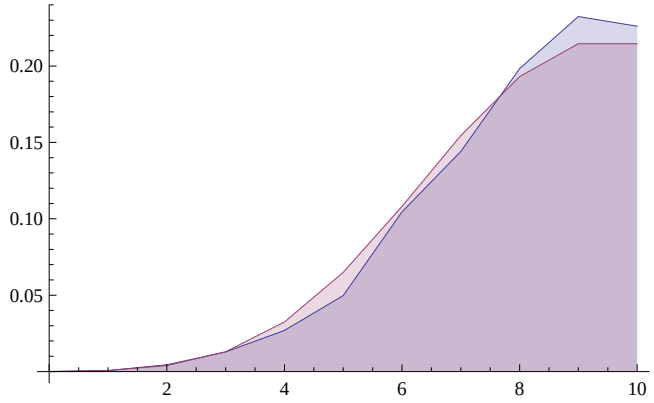


■ M/M/K/K Histogram

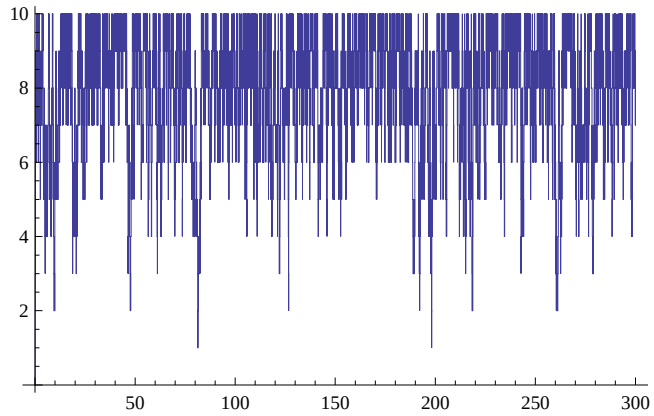
In[58]:=
```
λ = 10.0;
μ = 1.0;
K = 10;
mm1Birth[s_] := If[s < K, λ, 0];
mm1Death[s_] := μ s;
out = bdSimWithStat[mm1Birth, mm1Death, 300];
hist = Select[Transpose[{Table[i, {i, 0, Length[out[[2]]] - 1}], out[[2]]}], #[[2]] > 0 &];
theoreticalHist = Table[{i, 1 / Sum[(λ/μ)^j / j!, {j, 0, K}] (λ/μ)^i / i!}, {i, 0, K}];
```

In[66]:=
```
ListPlot[{hist, theoreticalHist}, Filling → Axis, Joined → True,
 PlotRange → All, PlotStyle → PointSize[0.02], AxesOrigin → {0, 0}]
ListPlot[out[[1]], Joined → True]
```

Out[66]=



Out[67]=



2) Sojourn Times

In[68]:=
```
bdSimWithStatAndSojourns[birthFunction_, deathFunction_, timeHorizon_] :=
 Module[{state = 0, time = 0.0},
  rel = {{time, state}};
  soujournList = {};
  (*queue is a list of arrival times of customers*)
  queue = {};
  histogramTable = Table[0.0, {100}];
  (*Now loop untill time > timeHorizon and append to rel as we go*)
  While[time < timeHorizon,
   birthRate = birthFunction[state];
   deathRate = deathFunction[state];
   stateDuration = RandomReal[ExponentialDistribution[birthRate + deathRate]];
   histogramTable[[state + 1]] += stateDuration;
   time += stateDuration;
   AppendTo[rel, {time, state}];
   change = 2 RandomInteger[BernoulliDistribution[birthRate/(birthRate + deathRate)]] - 1;
   state += change;
   If[change == 1,
    AppendTo[queue, time]
    ,
    (*Else change==-1*)
    AppendTo[soujournList, time - queue[[1]]];
    queue = Drop[queue, 1]
   ];
   AppendTo[rel, {time, state}];
  ];
  {rel, histogramTable/timeHorizon, soujournList}
 ]
```

In[69]:=
```
Table[
 λ = 0.3;
 μ = 1.0;
 mm1Birth[s_] := λ;
 mm1Death[s_] := If[s > 0, μ, 0];
 out = bdSimWithStatAndSojourns[mm1Birth, mm1Death, 3000];
 out[[3]] // Mean,
 {10}]
```

Out[69]= {1.46791, 1.57294, 1.31035, 1.33803, 1.41529, 1.33298, 1.5891, 1.44209, 1.45477, 1.39011}

The M/G/1 Queue - The Pollaczek-Khinchin Formulas

$$E[W] = \frac{\rho}{1-\rho}\,\frac{1}{2}\left(c_s^2+1\right)\,\frac{1}{\mu}$$

Stationary distribution of W and/or N is given in form of LSTs,

$$W^*(s) = \frac{(1-\rho)\,s}{\lambda\,G^*(s)+s-\lambda}$$

Similar formula for stationary distribution of Number in System

The G/M/1 Queue

Here we need to solve the equation

$$\sigma = A^*(\mu - \mu\sigma)$$

The Queue length that an arriving customer sees is $\text{Geometric}_0(\sigma)$

```
Clear[λ, μ, K]
```

```
A1[s_] := λ/(λ + s)
```

```
A2[s_] := (2 λ/(2 λ + s))^2
```

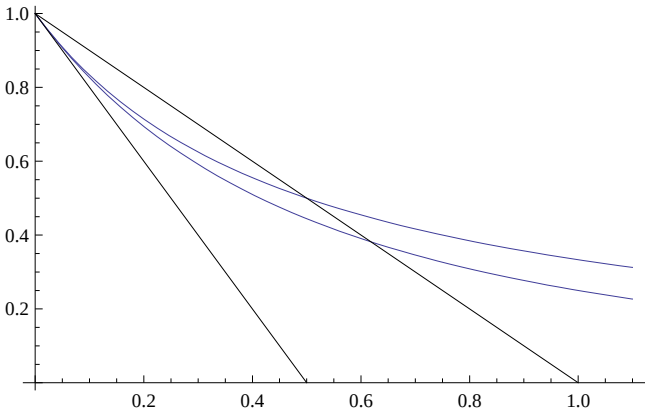
```
-D[A1[s], s] /. s → 0
```

$$\frac{1}{\lambda}$$

```
-D[A2[s], s] /. s → 0
```

$$\frac{1}{\lambda}$$

```
Plot[{A1[s], A2[s]} /. λ → 1/2, {s, 0, 1.1},
 AxesOrigin → {0, 0}, Epilog → {Line[{{0, 1}, {1, 0}}],
   Line[{{0, 1}, {1/2, 0}}]}]
```




```
A[s_] := (kk λ/(kk λ + s))^kk
```