Universidade Federal de Uberlândia

GBC065 – Modelagem e Simulação
TRABALHO 3

Alunos: -João Vitor Rodrigues de Alcântara Ferreira
-Thiago Lucas Carvalho

Exercícios:

Exercise 3.1.2
(a) Relative to the steady-state statistics in Example 3.1.3 and the statistical equations in Section 1.2, list all of the consistency checks that should be applicable.

$\bar{r}$	$\bar{w}$	$\bar{d}$	$\bar{s}$	$\bar{l}$	$\bar{q}$	$\bar{x}$
2.00	3.83	2.33	1.50	1.92	1.17	0.75

w é a média de jobs em espera
d é a média de jobs em delay
s é a média de jobs em serviço
l é o tempo médio no nó
x é o tempo médio de ocupação no server em %
q é o tempo médio na fila

Elas se relacionam da seguinte maneira: w = d + s

Para verificarmos o tempo médio no nó temos: l = q + x

(b) Verify that all of these consistency checks are valid.

Para verificar se a média de jobs em fila é válido temos:
3.83 = 2.33 + 1.50
3.83 = 3.83
Essa consistência é válida

Para verificar se o tempo médio no nodo é válido temos:
1.92 = 1.17 + 0.75
1.92 = 1.92
Essa consistência é válida

Exercise 3.1.4

(a) Conduct a transition-to-steady-state study like that in Example 3.1.3 except for a service time model that is Uniform(1.3, 2.3). Be specific about the number of jobs that seem to be required to produce steady-state statistics.

Seed 2121212

```
Indice 61840: 10.82
Indice 61880: 10.82
Indice 61920: 10.82
Indice 61960: 10.83
Indice 62000: 10.82
Indice 62040: 10.82
Indice 62080: 10.82
Indice 62120: 10.82
Indice 62160: 10.82
Indice 62200: 10.81
Indice 62240: 10.82
Indice 62280: 10.81
Indice 62320: 10.81
Indice 62360: 10.80
Indice 62400: 10.80
Indice 62440: 10.80
Indice 62480: 10.79
Indice 62520: 10.79
Indice 62560: 10.79
Indice 62600: 10.79
Indice 62640: 10.79
Indice 62680: 10.79
Indice 62720: 10.78
Indice 62760: 10.78
Indice 62800: 10.78
Indice 62840: 10.78
Indice 62880: 10.78
Indice 62920: 10.77
Indice 62960: 10.77
```

```
Indice 1245200: 10.29
Indice 1245240: 10.29
Indice 1245280: 10.29
Indice 1245320: 10.29
Indice 1245360: 10.29
Indice 1245400: 10.29
Indice 1245440: 10.29
Indice 1245480: 10.29
Indice 1245520: 10.29
Indice 1245560: 10.29
Indice 1245600: 10.29
Indice 1245640: 10.29
Indice 1245680: 10.29
Indice 1245720: 10.29
Indice 1245760: 10.29
Indice 1245800: 10.29
Indice 1245840: 10.29
Indice 1245880: 10.29
Indice 1245920: 10.29
Indice 1245960: 10.29
Indice 1246000: 10.29
Indice 1246040: 10.29
Indice 1246080: 10.29
Indice 1246120: 10.29
Indice 1246160: 10.29
Indice 1246200: 10.29
Indice 1246240: 10.29
Indice 1246280: 10.29
Indice 1246320: 10.29
```

```
Indice 2743880: 10.19
Indice 2743920: 10.19
Indice 2743960: 10.19
Indice 2744000: 10.19
Indice 2744040: 10.19
Indice 2744080: 10.19
Indice 2744120: 10.19
Indice 2744160: 10.19
Indice 2744200: 10.19
Indice 2744240: 10.19
Indice 2744280: 10.19
Indice 2744320: 10.19
Indice 2744360: 10.19
Indice 2744400: 10.19
Indice 2744440: 10.19
Indice 2744480: 10.19
Indice 2744520: 10.19
Indice 2744560: 10.19
Indice 2744600: 10.19
Indice 2744640: 10.19
Indice 2744680: 10.19
Indice 2744720: 10.19
Indice 2744760: 10.19
Indice 2744800: 10.19
Indice 2744840: 10.19
Indice 2744880: 10.19
Indice 2744920: 10.19
Indice 2744960: 10.19
Indice 2745000: 10.19
Indice 2745040: 10.19
```

```
 Para produzir um steady-state precisamos de: 2745060 jobs
average interarrival time =   2.00
average wait ............ =  10.18
average delay ........... =   8.38
average service time .... =   1.80
average # in the node ... =   5.09
average # in the queue .. =   4.19
utilization ............. =   0.90
```

Seed 54321

C:\Users\Thiago\Desktop'

		Indice 2153040: 10.13
Indice 46800: 10.75	Indice 1574560: 10.02	Indice 2153080: 10.13
Indice 46840: 10.75	Indice 1574600: 10.02	Indice 2153120: 10.13
Indice 46880: 10.75	Indice 1574640: 10.02	Indice 2153160: 10.13
Indice 46920: 10.74	Indice 1574680: 10.02	Indice 2153200: 10.13
Indice 46960: 10.74	Indice 1574720: 10.02	Indice 2153240: 10.13
Indice 47000: 10.74	Indice 1574760: 10.02	Indice 2153280: 10.13
Indice 47040: 10.74	Indice 1574800: 10.02	Indice 2153320: 10.13
Indice 47080: 10.74	Indice 1574840: 10.02	Indice 2153360: 10.13
Indice 47120: 10.73	Indice 1574880: 10.02	Indice 2153400: 10.13
Indice 47160: 10.74	Indice 1574920: 10.02	Indice 2153440: 10.13
Indice 47200: 10.73	Indice 1574960: 10.02	Indice 2153480: 10.13
Indice 47240: 10.73	Indice 1575000: 10.02	Indice 2153520: 10.13
Indice 47280: 10.72	Indice 1575040: 10.02	Indice 2153560: 10.13
Indice 47320: 10.72	Indice 1575080: 10.02	Indice 2153600: 10.13
Indice 47360: 10.72	Indice 1575120: 10.02	Indice 2153640: 10.13
Indice 47400: 10.72	Indice 1575160: 10.02	Indice 2153680: 10.13
Indice 47440: 10.73	Indice 1575200: 10.02	Indice 2153720: 10.13
Indice 47480: 10.73	Indice 1575240: 10.02	Indice 2153760: 10.13
Indice 47520: 10.72	Indice 1575280: 10.02	Indice 2153800: 10.13
Indice 47560: 10.72	Indice 1575320: 10.02	Indice 2153840: 10.13
Indice 47600: 10.71	Indice 1575360: 10.02	Indice 2153880: 10.13
Indice 47640: 10.71	Indice 1575400: 10.02	Indice 2153920: 10.13
Indice 47680: 10.72	Indice 1575440: 10.02	Indice 2153960: 10.13
Indice 47720: 10.72	Indice 1575480: 10.02	Indice 2154000: 10.13
Indice 47760: 10.71	Indice 1575520: 10.02	Indice 2154040: 10.13
Indice 47800: 10.71	Indice 1575560: 10.02	Indice 2154080: 10.13
Indice 47840: 10.70	Indice 1575600: 10.02	Indice 2154120: 10.13
Indice 47880: 10.70	Indice 1575640: 10.02	Indice 2154160: 10.13
Indice 47920: 10.70	Indice 1575680: 10.02	Indice 2154200: 10.13

```
 Para produzir um steady-state precisamos de: 2154200 jobs
average interarrival time =   2.00
average wait ............ =  10.13
average delay ........... =   8.33
average service time .... =   1.80
average # in the node ... =   5.06
average # in the queue .. =   4.16
utilization ............. =   0.90
```

Seed 12345

```
Indice 50920: 9.91
Indice 50960: 9.92
Indice 51000: 9.92
Indice 51040: 9.91
Indice 51080: 9.91
Indice 51120: 9.91
Indice 51160: 9.91
Indice 51200: 9.92
Indice 51240: 9.93
Indice 51280: 9.94
Indice 51320: 9.94
Indice 51360: 9.94
Indice 51400: 9.94
Indice 51440: 9.94
Indice 51480: 9.94
Indice 51520: 9.94
Indice 51560: 9.93
Indice 51600: 9.93
Indice 51640: 9.93
Indice 51680: 9.92
Indice 51720: 9.92
Indice 51760: 9.92
Indice 51800: 9.92
Indice 51840: 9.92
Indice 51880: 9.91
Indice 51920: 9.91
Indice 51960: 9.91
Indice 52000: 9.90
Indice 52040: 9.90
```

```
Indice 1129800: 10.12
Indice 1129840: 10.12
Indice 1129880: 10.12
Indice 1129920: 10.12
Indice 1129960: 10.12
Indice 1130000: 10.12
Indice 1130040: 10.12
Indice 1130080: 10.12
Indice 1130120: 10.12
Indice 1130160: 10.12
Indice 1130200: 10.12
Indice 1130240: 10.12
Indice 1130280: 10.12
Indice 1130320: 10.12
Indice 1130360: 10.12
Indice 1130400: 10.12
Indice 1130440: 10.12
Indice 1130480: 10.12
Indice 1130520: 10.12
Indice 1130560: 10.12
Indice 1130600: 10.12
Indice 1130640: 10.12
Indice 1130680: 10.12
Indice 1130720: 10.12
Indice 1130760: 10.12
Indice 1130800: 10.12
Indice 1130840: 10.12
Indice 1130880: 10.12
Indice 1130920: 10.11
```

```
Indice 3371000: 10.11
Indice 3371040: 10.11
Indice 3371080: 10.11
Indice 3371120: 10.11
Indice 3371160: 10.11
Indice 3371200: 10.11
Indice 3371240: 10.11
Indice 3371280: 10.11
Indice 3371320: 10.11
Indice 3371360: 10.11
Indice 3371400: 10.11
Indice 3371440: 10.11
Indice 3371480: 10.11
Indice 3371520: 10.11
Indice 3371560: 10.11
Indice 3371600: 10.11
Indice 3371640: 10.11
Indice 3371680: 10.11
Indice 3371720: 10.11
Indice 3371760: 10.11
Indice 3371800: 10.11
Indice 3371840: 10.11
Indice 3371880: 10.11
Indice 3371920: 10.11
Indice 3371960: 10.11
Indice 3372000: 10.11
Indice 3372040: 10.11
Indice 3372080: 10.11
Indice 3372120: 10.11
Indice 3372160: 10.11
```

```
 Para produzir um steady-state precisamos de: 3372180 jobs
average interarrival time =   2.00
average wait ............ =  10.11
average delay ........... =   8.31
average service time .... =   1.80
average # in the node ... =   5.06
average # in the queue .. =   4.16
utilization ............. =   0.90
```

(b) Comment.

O steady-state varia constantemente até que os valores vão convergindo até chegar em um estado quase constante.

Exercise 3.1.5

(a) Verify that the mean service time in Example 3.1.4 is 1.5.

C:\Users\User\Desktop\TesteMS3\bin\Debug\TesteMS3.exe

```
for 500000 jobs
   average interarrival time =   2.00
   average wait ............ =   5.77
   average delay ........... =   4.27
   average service time .... =   1.50
   average # in the node ... =   2.89
   average # in the queue .. =   2.14
   utilization ............. =   0.75
```

O exemplo 3.1.4 mostra o seguinte service time :

$\bar{r}$	$\bar{w}$	$\bar{d}$	$\bar{s}$	$\bar{l}$	$\bar{q}$	$\bar{x}$
2.00	5.77	4.27	1.50	2.89	2.14	0.75

Apesar deste resultado ser igual, caso o número de jobs fosse menor (ex: 10000 padrão), o service time seria 1.49.

(b) Verify that the steady-state statistics in Example 3.1.4 seem to be correct.

```
for 500000 jobs
   average interarrival time =   2.00
   average wait ............ =   5.77
   average delay ........... =   4.27
   average service time .... =   1.50
   average # in the node ... =   2.89
   average # in the queue .. =   2.14
   utilization ............. =   0.75
```

Os resultados do exemplo 3.1.4 são obtidos mais precisamente de acordo com que o número de Jobs definido é aumentado, chegando a um valor exato ao do livro com 500.000 mil jobs processados.

(c) Note that the arrival rate, service rate, and utilization are the same as those in Example 3.1.3, yet all the other statistics are larger than those in Example 3.1.3. Explain (or conjecture) why this is so. Be specific.

As taxas no exemplo 3.1.4 serem maiores do que as do ex. 3.1.3 são explicadas pela sensibilidade das medidas de desempenho à distribuição do tempo de serviço, destacando também a importância de usar-se um modelo de tempo de serviço o mais preciso possível.

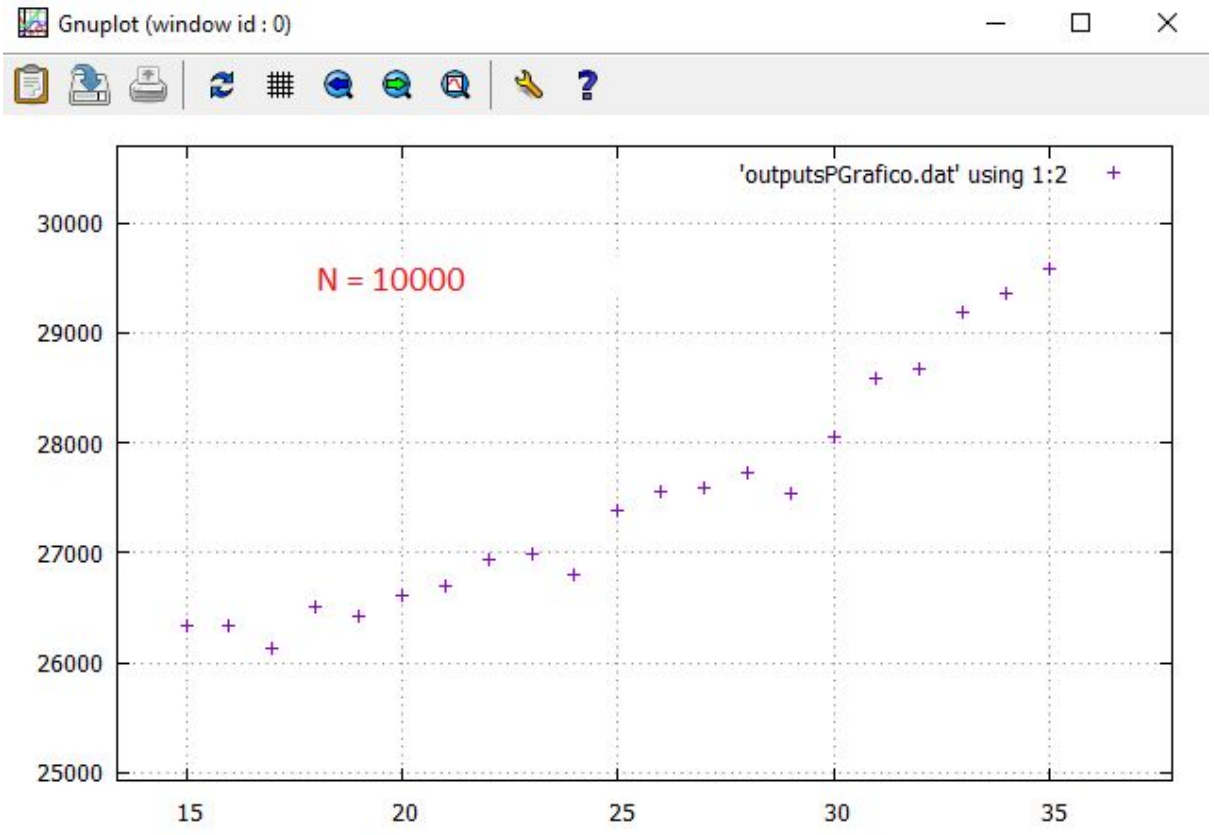
Exercise 3.1.6

(a) Modify program sis2 to compute data like that in Example 3.1.7. Use the functions PutSeed and GetSeed from the library rng in such a way that one initial

seed is supplied by the system clock, printed as part of the program's output and used automatically to generate the same demand sequence for all values of s.

Código em anexo

(b) For s = 15, 16, . . . , 35 create a figure (or table) similar to the one in Example 3.1.7.

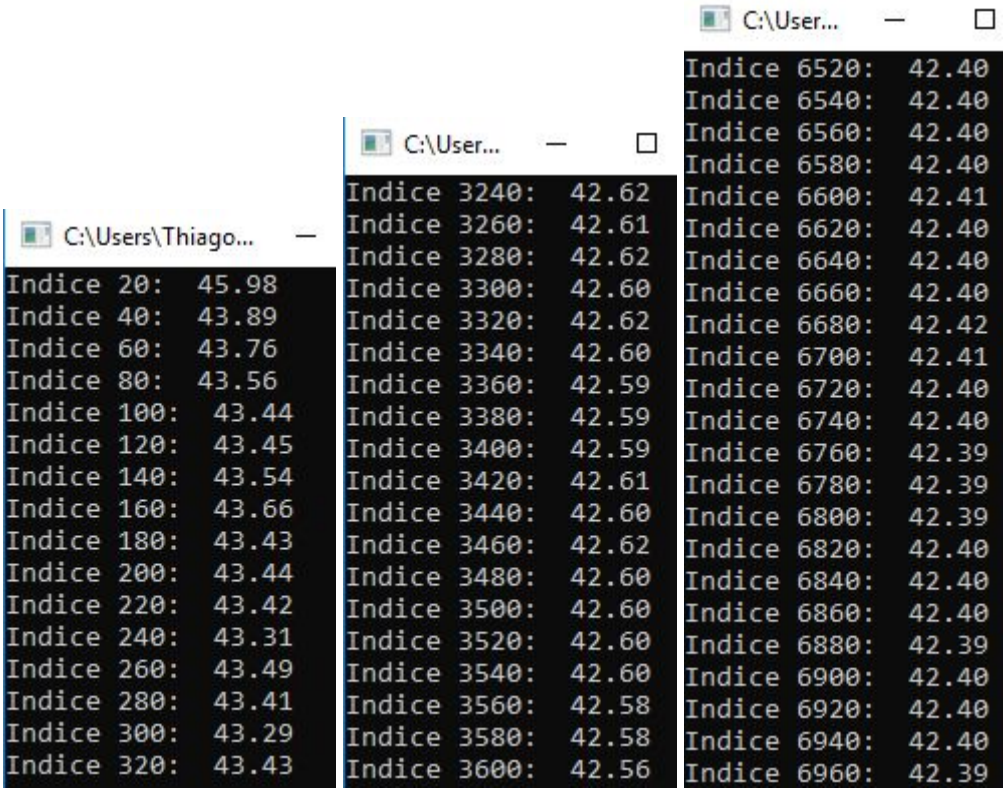


(c) Comment.

Pelo exemplo observado no livro e pelos testes feitos, pode-se observar que o uso de seed's definidas inicialmente para seed's geradas por alguma política randômica é que a variabilidade das medidas e resultados mudam, sendo a randômica geradora de resultados mais naturais e satisfatórios.

Exercise 3.1.7
(a) Relative to Example 3.1.5, if instead the random variate sequence of demands are generated as d i = Equilikely(5, 25) + Equilikely(5, 25) i = 1, 2, 3, . . . then, when compared with those in Example 3.1.6, demonstrate that some of the steady-state statistics will be the same and others will not.

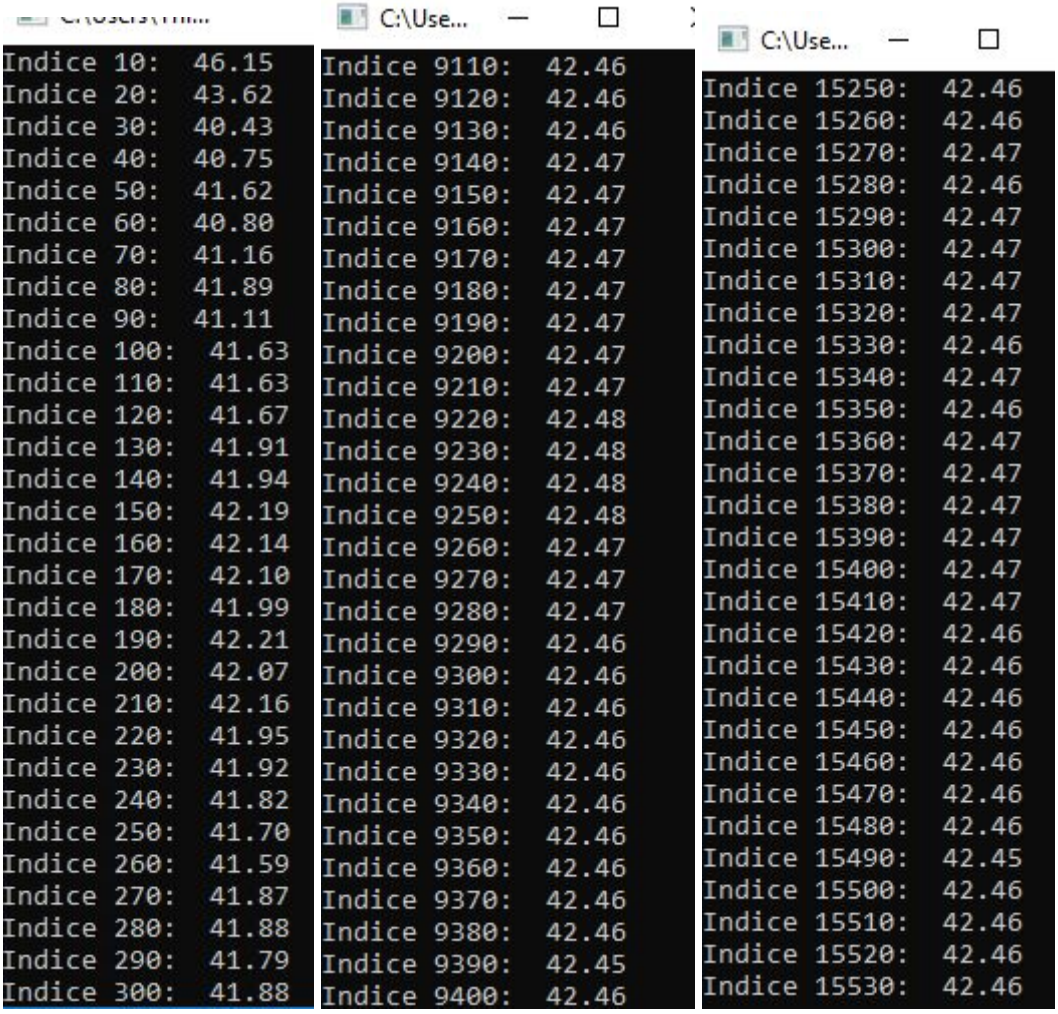
Seed 2121212




```
for 6980 time intervals with an average demand of  30.00
and policy parameters (s, S) = (20, 80)

  average order ............. =  30.00
  setup frequency ........... =   0.39
  average holding level ..... =  42.58
  average shortage level .... =   0.18
  Nivel medio de inventario . =  42.39
```

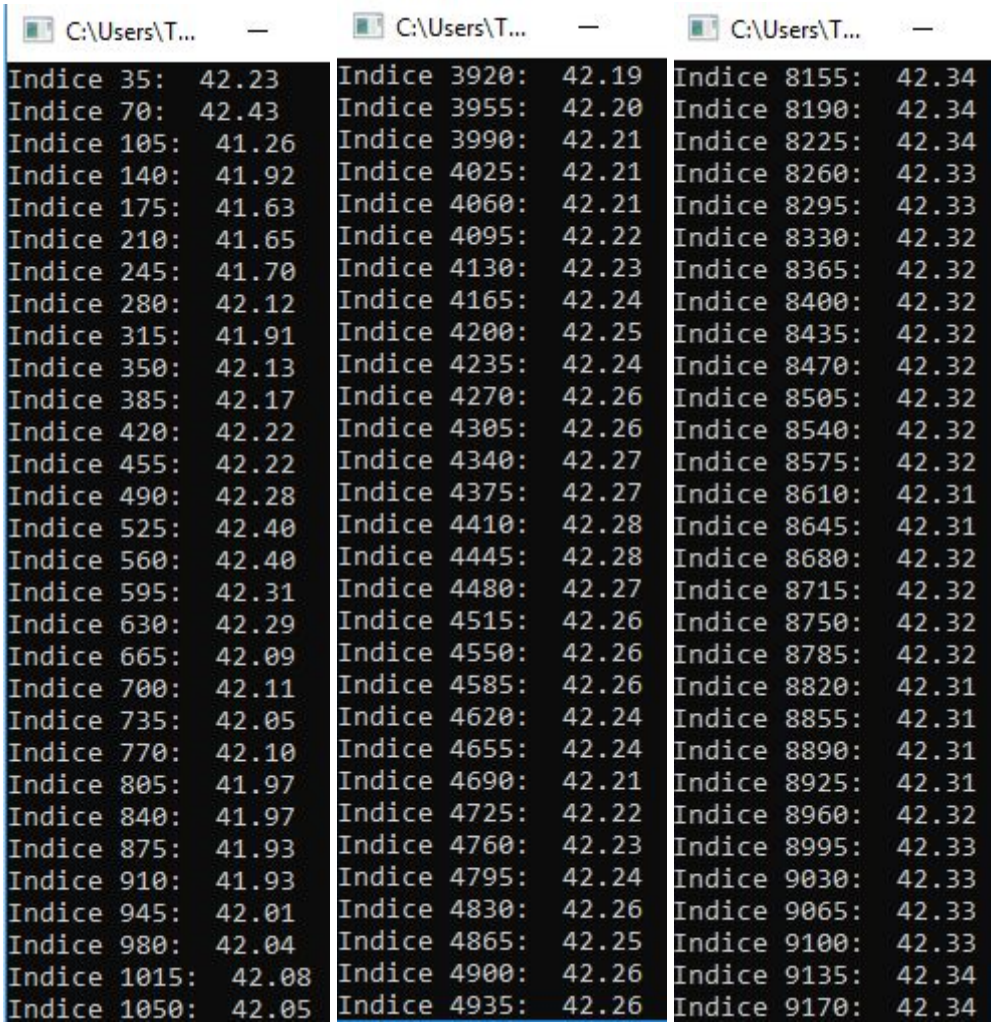
Seed 54321

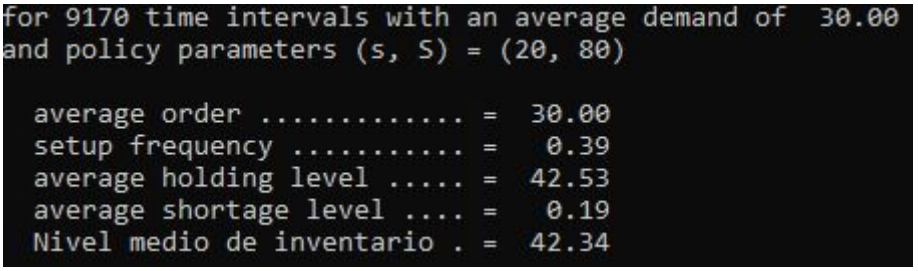



```
for 15530 time intervals with an average demand of  30.00
and policy parameters (s, S) = (20, 80)

  average order ............. =  30.00
  setup frequency ........... =   0.39
  average holding level ..... =  42.64
  average shortage level .... =   0.18
  Nivel medio de inventario . =  42.46
```

Seed 12345





(b) Explain why this is so.

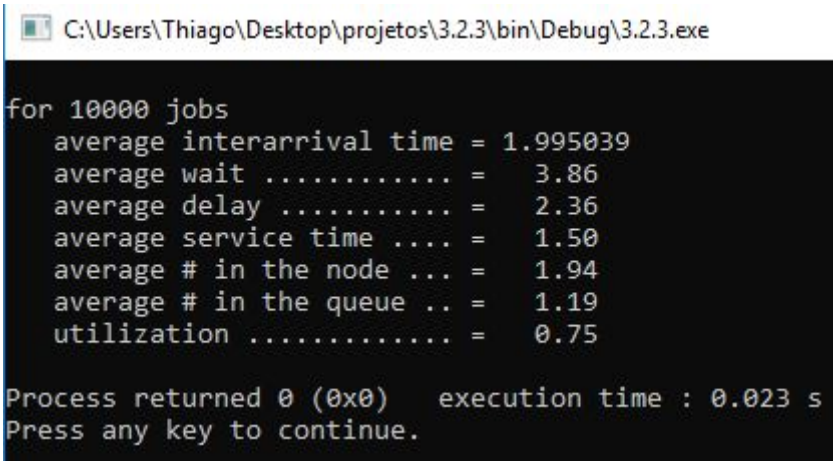
O que acontece é que os valores de Equilikely tem um intervalo menor do que o do exemplo, e quando executamos o programa ele retorna valores menores para o holding level e shortage level.

Exercise 3.2.3 Modify program ssq2 as suggested in Example 3.2.7 to create two programs that differ only in the function GetService. For one of these programs, use the function as implemented in Example 3.2.7; for the other program, use

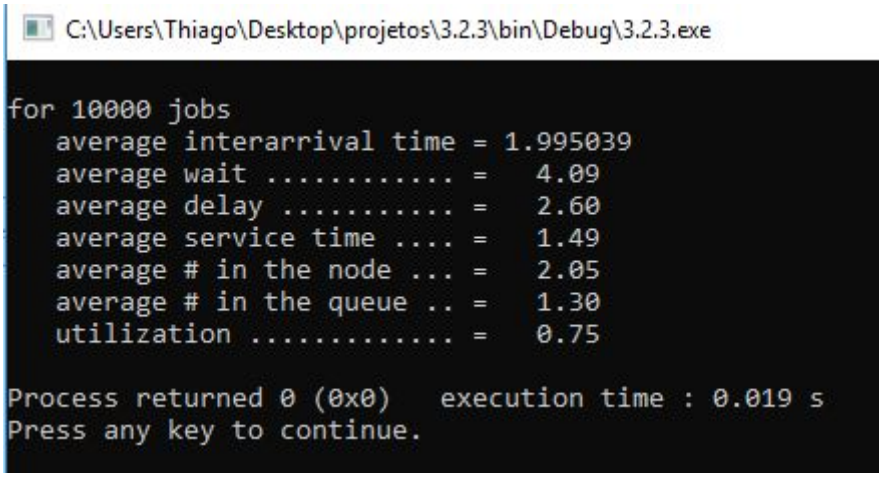
double GetService(void) { SelectStream(2); /* this line is new */ return (Uniform(0.0, 1.5) + Uniform(0.0, 1.5)); }

(a) For both programs verify that exactly the same average interarrival time is produced (print the average with d.dddddd precision). Note that the average service time is approximately the same in both cases, as is the utilization, yet the service nodes statistics $\bar{w}$, d, $\bar{l}$, and $\bar{q}$ are different.

Código 1:



Código 2:



(b) Why?

A média de tempo de serviço fica parecido nos dois códigos pois a função é a função uniform Uniform(0.0,1.5) + Uniform(0.0,1.5) é quase a mesma coisa que Uniform(1.0,2.0)

Exercise 3.3.10 Modifique o programa sis2 para incluir lag na entrega do fornecedor e construir gráficos similares ao Example 3.3.4. Para reproduzir o gráfico sem lag faça com que o Delta (variável que implementa o lag) seja igual a zero e compare os resultados.