Solution 8

Robin Niebergall robin.niebergall@student.uni-tuebingen.de 4255194

Aron Rath aron.rath@student.uni-tuebingen.de 4251981

Problem 8.1

8.1.1

State 0: System ist leer.

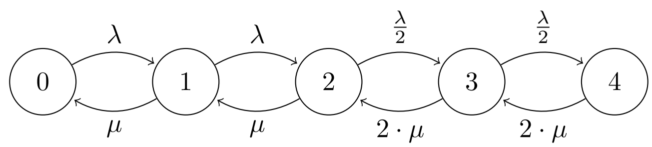
State 1: Eine Service-Einheit ist aktiv.

State 2: Beide Service-Einheiten sind aktiv.

State 3: Beide Service-Einheiten sind aktiv, ein Waiting-Slot ist belegt.

State 4: Beide Service-Einheiten sind aktiv, beide Waiting-Slots sind belegt.

8.1.2



8.1.3

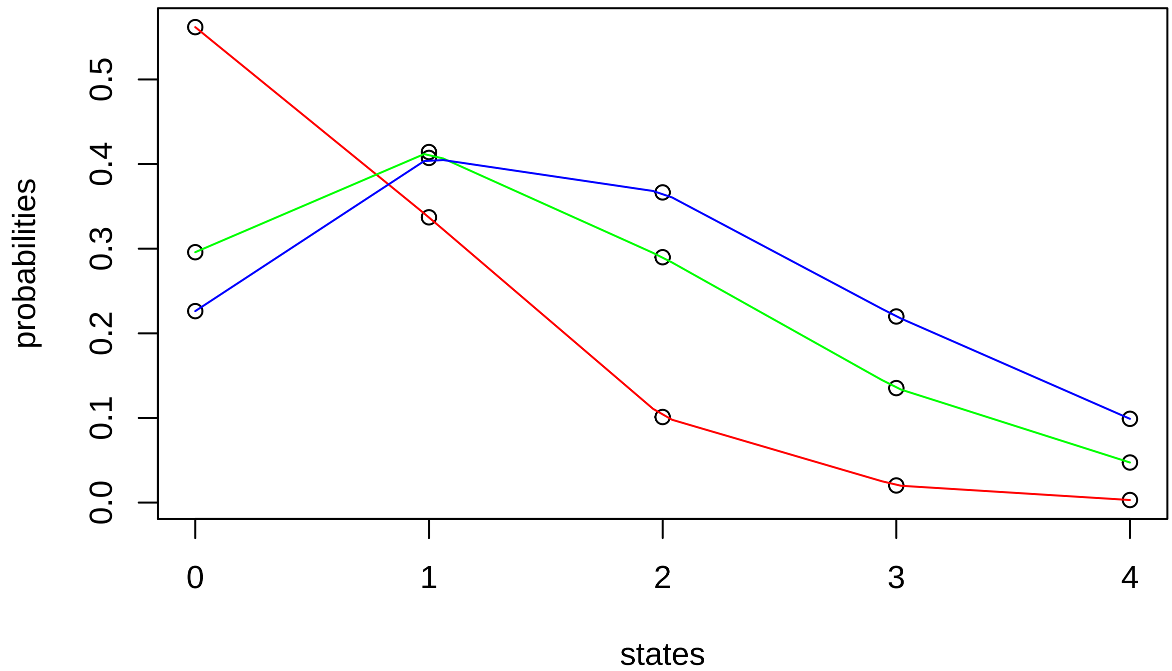
$$
\begin{aligned}
x(0) &= \left(1 + \sum_{0<i\leq n} \frac{\prod_{0<k\leq i} \lambda_{k-1}}{\prod_{0<k\leq i} \mu_k}\right)^{-1} \\
x(0) &= \left(1 + \frac{\lambda}{\mu} + \frac{\lambda^2}{\mu^2} + \frac{\lambda^3}{2^2 \cdot \mu^3} + \frac{\lambda^4}{2^4 \cdot \mu^4}\right) \\
x(1) &= x(0) \cdot \frac{\lambda}{\mu} \\
x(2) &= x(0) \cdot \frac{\lambda^2}{\mu^2} \\
x(3) &= x(0) \cdot \frac{\lambda^3}{2^2 \cdot \mu^3} \\
x(4) &= x(0) \cdot \frac{\lambda^4}{2^4 \cdot \mu^4}
\end{aligned}
$$

8.1.4

```r
# 8.1.4
n <- 2
rho <- c(0.3, 0.7, 0.9)

stateProbabilities <- function(i, a, n) {
  sum <- 0.0
  for (j in 0:n) {
```

```
    sum <- sum + ((a ^ j) / factorial(j))
  }
  x_i <- ((a ^ i) / factorial(i)) / sum
  return(x_i)
}

x0_a1 = stateProbabilities(i = 0, a = (rho[1] * n), n)
x1_a1 = stateProbabilities(i = 1, a = (rho[1] * n), n)
x2_a1 = stateProbabilities(i = 2, a = (rho[1] * n), n)
x3_a1 = stateProbabilities(i = 3, a = (rho[1] * n), n)
x4_a1 = stateProbabilities(i = 4, a = (rho[1] * n), n)

x0_a2 = stateProbabilities(i = 0, a = (rho[2] * n), n)
x1_a2 = stateProbabilities(i = 1, a = (rho[2] * n), n)
x2_a2 = stateProbabilities(i = 2, a = (rho[2] * n), n)
x3_a2 = stateProbabilities(i = 3, a = (rho[2] * n), n)
x4_a2 = stateProbabilities(i = 4, a = (rho[2] * n), n)

x0_a3 = stateProbabilities(i = 0, a = (rho[3] * n), n)
x1_a3 = stateProbabilities(i = 1, a = (rho[3] * n), n)
x2_a3 = stateProbabilities(i = 2, a = (rho[3] * n), n)
x3_a3 = stateProbabilities(i = 3, a = (rho[3] * n), n)
x4_a3 = stateProbabilities(i = 4, a = (rho[3] * n), n)

data <- c(
  x0_a1,
  x0_a2,
  x0_a3,
  x1_a1,
  x1_a2,
  x1_a3,
  x2_a1,
  x2_a2,
  x2_a3,
  x3_a1,
  x3_a2,
  x3_a3,
  x4_a1,
  x4_a2,
  x4_a3
)

probabilities <- matrix(
  data = data,
  nrow = 5,
  ncol = 3,
  byrow = TRUE
)

states <- matrix(0:4, 5, 3)

plot(states, probabilities)
```

```
lines(approx(states[, 1], probabilities[, 1]), col = "red")
lines(approx(states[, 2], probabilities[, 2]), col = "green")
lines(approx(states[, 3], probabilities[, 3]), col = "blue")
```



8.1.5

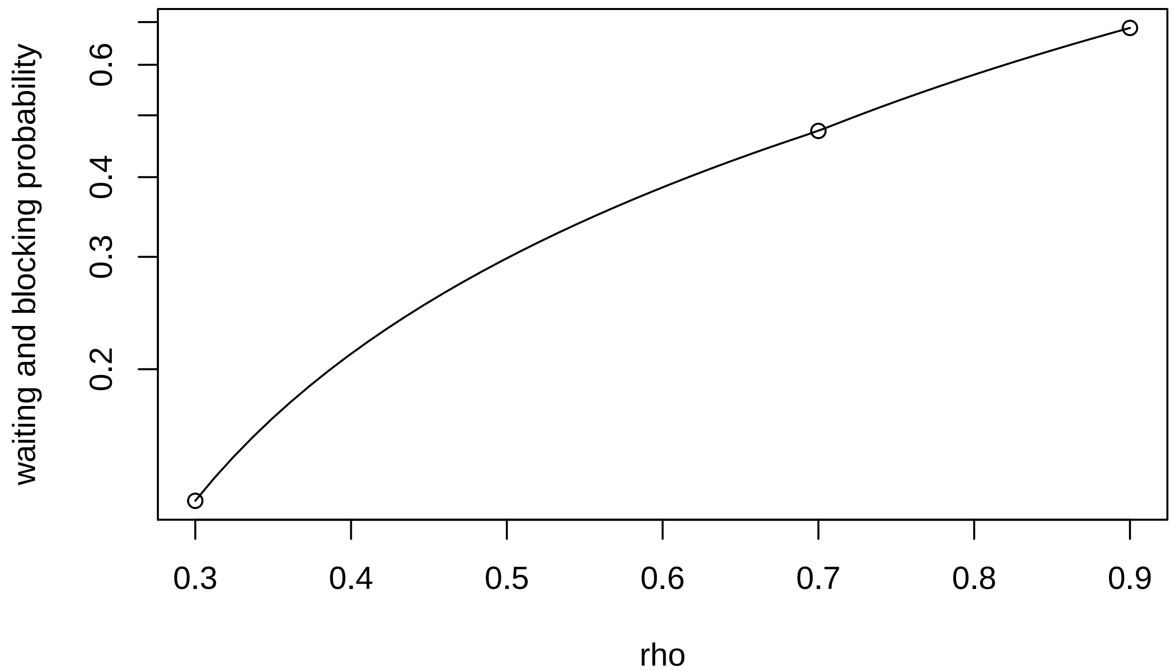
8.1.6

```
# 8.1.6
for (i in seq_along(rho)) {
  message("rho = ", rho[i])
  message("waiting and blocking probability for new customers: ",
          sum(probabilities[3:5, i]))
}
```

```
## rho = 0.3

## waiting and blocking probability for new customers: 0.12438202247191

## rho = 0.7

## waiting and blocking probability for new customers: 0.472603550295858

## rho = 0.9

## waiting and blocking probability for new customers: 0.685384615384615
```

8.1.7

```r
wbprobabilities = c(sum(probabilities[3:5, 1]),
                    sum(probabilities[3:5, 2]),
                    sum(probabilities[3:5, 3]))
plot(rho, wbprobabilities, ylab = "waiting and blocking probability", log = "y")
lines(approx(rho, wbprobabilities))
```



8.1.8

```r
# 8.1.8
for (i in seq_along(rho)) {
  p_u = ((rho[i] * n) * (1 - probabilities[5, i])) / n
  message("rho = ", rho[i])
  message("p_u = ", p_u)
}
```

```
## rho = 0.3

## p_u = 0.299089887640449

## rho = 0.7

## p_u = 0.666850098619329

## rho = 0.9

## p_u = 0.81093665158371
```