Problem 1 (Ross 7.1,2)
100 iterations:
Avg customer time in system: 0.2153. Avg idle time: 2.3418. Avg overtime: 0.1892

1000 iterations:
Avg customer time in system: 0.2106. Avg idle time: 2.3369. Avg overtime: 0.1503

Problem 2 (Ross 7.3)
Avg time on break: 53.9884

Problem 3 (Ross 7.5,6)
Expected number of lost customers: 120.586

Problem 4 (Ross 7.11)
Probability always positive: 0.905

Problem 5 (Ross 7.15)

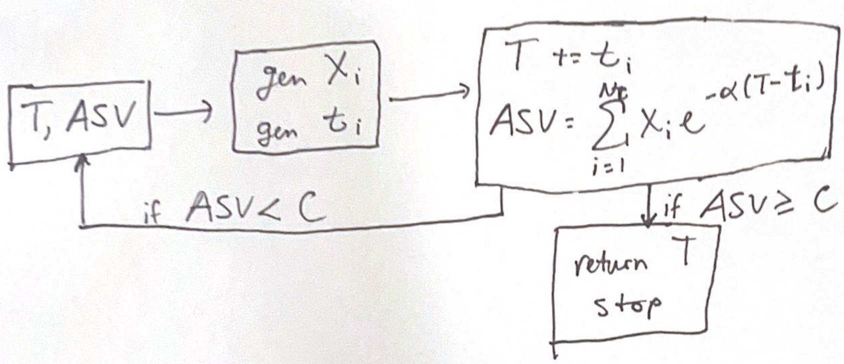
7.15. a

Events: time of shock

Variables: t_s: time of shock "s"

X_s: initial size of shock "s"

net shock damage: ASV:



b. Average time to system fail: 0.5092208923480627

Problem 6 (Ross 7.17)
Expected stock gain at beginning of day: {0: 0.0, 1: 0.0, 2: 0.51, 3: 1.2, 4: 2.06, 5: 3.17, 6: 4.77, 7: 6.71, 8: 8.78, 9: 11.05, 10: 13.51, 11: 16.06, 12: 18.73, 13: 21.48, 14: 24.24, 15: 27.24, 16: 30.4, 17: 33.66, 18: 36.99, 19: 40.49}

Problem 7 (Ross 7.18)

7.18

variables:

C = # of customers who don't buy anything, T : time horizon

N_t = toy inventory at time t

Update Method

- initialize $t=0$, $C=0$, $N=4$

- while $t < T$

1) generate t_c : customer arrival time $\sim$ poisson(λ)
can use inverse transform

2) t += t_c

3) if $t > T$
- break out of while loop

4) generate d_c : customer's number of toys wants to buy
follow rules of P_i

5) if $d_c > N$:
- C += 1 : update # who have left w/out buying

else :

- N -= d_c : remove d_c units from inventory

6) if N == 0 :
$N = 10$: restock inventory immediately if empty

Problem 9
Using homogeneous arrival process
1K simulations Mean of customers in system at time = 50: 0.07
1K simulations Variance of customers in system at time = 50: 0.06509999999999999
1K simulations Mean of customers in system at time = 100: 1.043
1K simulations Variance of customers in system at time = 100: 0.04115099999999999