In [ ]:

```
import numpy as np
import bisect
import math


class Customer:
    def __init__(self, arrival_time, service_time):
        self.service_time = service_time
        self.blocked = False

        self.event = "arrival"
        self.event_time = arrival_time


    def arrive(self, servers, event_list):
        if servers < 1:
            self.blocked = True
            return servers
        else:
            servers -= 1
            servers = max(servers, 0)
            self.event = "departure"
            self.event_time += self.service_time
            bisect.insort(event_list, self, key=lambda x: x.event_time)
            return servers

    def depart(self, servers, m):
        servers += 1
        servers = min(servers, m)
        return servers


def main_loop(arrival_interval, service_time, m, repititions = 10):
    blocked = np.zeros(repititions)
    for i in range(repititions):
        arrival_intervals = arrival_interval()
        service_times = service_time()
        arrival_times = np.cumsum(arrival_intervals)
        event_list = [Customer(arrival_times[i],service_times[i]) for i in range
        event_list.sort(key=lambda x: x.event_time)
        open_servers = m
        while event_list:
            event = event_list.pop(0)
            if event.event == "arrival":
                open_servers = event.arrive(open_servers, event_list)
                blocked[i] += event.blocked
            elif event.event == "departure":
                open_servers = event.depart(open_servers, m)
    return blocked


def confidence_intervals(samples):
    emp_mean = np.mean(samples)
    emp_std = np.std(samples)
    t = 1.96
    return (emp_mean - t*emp_std/np.sqrt(len(samples)), emp_mean + t*emp_std/np.

#Erlang B formula
```

```
def erlang_b(m, A):
    return (A**m/math.factorial(m))/np.sum([A**i/math.factorial(i) for i in rang
```