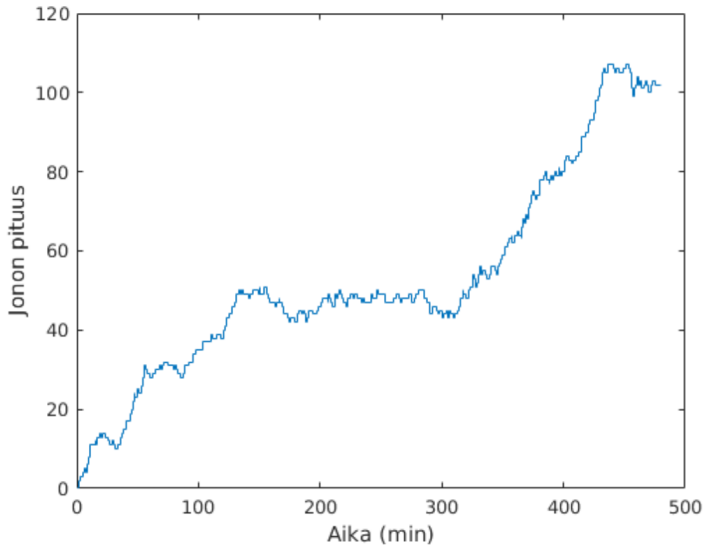
SCI-C0200 - Fysiikan ja matematiikan menetelmien studio

Hampurilaisbaarin jonon simulointi

Osama Abuzaid 524832
Joonatan Bergholm 507260

6. toukokuuta 2016



Kuva 1: Tilanne kun vain kaksi kassaa on käytössä.

Poisson-jakauma

Saamme eksponentiaalisesti jakautuneen satunnaismuuttujan t yhtälöstä $F(t) = R$, jossa F on eksponentiaalijakauman kertymäfunktio ja R on tasajakautunut satunnaismuuttuja.

$$\begin{aligned} F(t) &= R \\ 1 - e^{-\lambda t} &= R \\ -\lambda t &= \ln(1 - R) \\ t &= -\frac{1}{\lambda} \ln(1 - R), \end{aligned}$$

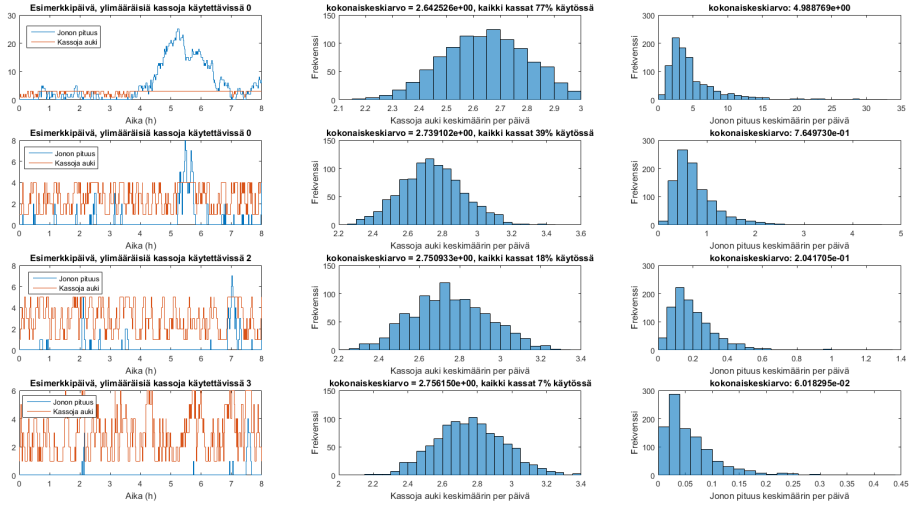
missä λ = autoja/aikayksikkö keskimäärin ja μ asiakkaan asiointikesto keskimäärin. Tälloin poisson-jakauma saadaan, kun otetaan satunnaislukuja t kunnes niiden summa ylittää jonkin rajan.

Meidän simulaatiossa $\lambda = 40\frac{1}{\text{h}}$ ja $\mu = 4$ min.

Tehtävä a

Kuvassa 1 näkyy jonon pituuden kehitys, kun vain kaksi kassaa on auki. Tämä olisi hampurilaisyrittäjän kannalta erittäin epäedullinen tilanne. Monte Carlo -simulaatiolla saadaan keskimäärin jonon pituudeksi päivän lopussa noin 83.

Tehtävä b



Kuva 2: Monte-Carlo simulaation tulokset. Vasemmanpuoleisimmat kuvaajat kertovat yhden päivän jononkehityksen tietyllä kassojen määrällä. Ylimääräisillä kassoilla tarkoitetaan niitä kassoja, jotka eivät ole välttämättömiä pitämään jonoa keskimäärin vakiona, ts. jos kassoja on n kappaletta, on ylimääräisten kassojen lukumäärä $n - \lceil \mu\lambda \rceil$. Keskellä olevissa histogrammeissa on aukiolevien kassojen keskimääräinen määrä per päivä, otos 1000 päivää. Vastaavasti oikeanpuolimmaisssa histogrammeissa on jonon keskimääräinen pituus per päivä, otos 1000 päivää.

Kassahenkilökuntaa on oltava käytettävissä vähintään sen verran, että jono ei pääse kasvamaan mielivaltaisesti. Tämä tapahtuu silloin, kun odotusarvoisesti asiakkaita tulee yhtä nopeasti sisään kuin niitä menee myös ulos. Mallin mukaan asiakkaita poistuu keskimäärin $\frac{1}{\mu_j}$ ja autoja tulee λ autoa aikayksikössä. Jotta

jonon pitus ei pääsisi pitkällä aikavälillä kasvamaan, on oltava

$$\begin{aligned} \max\left(\frac{1}{\mu_j}\right) &= \lambda \\ \frac{s}{\mu} &= \lambda \\ s &= \mu\lambda \approx 2,7. \end{aligned} \tag{1}$$

Näin ollen kassahenkilökuntaa on oltava saatavilla vähintään $\lceil \mu\lambda \rceil = 3$ kappaletta, jotta jonon pituus pysyy kutakuinkin vakiona. Kuitenkin pahimpien ruuhkien varalta kannattaa olla jokunen lisäkassa käytettävissä, jotta jonojen pituudet voidaan laskea halutulle tasolle.

Kokeillaan strategiaa, jossa kassoja pidetään auki yhtä monta kuin asioivia asiakkaita on kyseisellä hetkellä, jos suinkin mahdollista. Tällöin asiakkaiden jonotusaika on triviaalisti optimaalinen. Kassojen määrää optimoidaan Monte-Carlo simulaatiolla: testataan 1000 päivän otoksella keskimääräisiä jonojen pituuksia sekä auki olevien kassojen määriä eri kassamäärillä ja valitaan näistä se, joka on kustannustehokkain ja jossa jonoja ei pääse kertymään kohtuuttoman paljon. Simulaation tulokset näkyvät kuvassa 2.

Havaitaan, että ylimääräisten kassojen lisääminen ei vaikuta merkittävästi aukiolevien kassojen keskimääräiseen lukumäärään, mutta niiden vaikutus jonojen pituuteen ja erityisesti jakaumaan on merkittävä. Kun ylimääräisiä kassoje ei ole lainkaan käytettävissä, yltää jonojen keskimääräisen pituuden histogrammin häntä jopa 35 asti. Esimerkkipäivän aikaväliä 4-7h tutkailemalla nähdään hyvin miksi: jos asiakkaita tulee kerralla paljon, niin jonon pituutta ei saada lyhennettyä tehokkaasti edes täydellä kapasiteetilla, vaikkei se enää merkittävästi kasvakaan. Tämä näkyy myös siinä, että kaikki kassat on käytettävissä 77% ajasta. Tämä oli odotettavissakin, sillä vaadittava minimikassamäärä on määritetty siten, ettei jonon pituus keskimäärin muutu miksikään.

Jonon keskimääräisen pituuden häntä lyhenee huomattavasti jo ensimmäisen ylimääräisen kassan lisäyksellä. Vaikka jonon pituus jossain vaiheessa kasvaisikin pitkäksi, niin ylimääräinen kassa pitää huolen siitä, että se myös lyhenee. Tässä skenaariossa kassoja pidetään keskimäärin auki vain n. 0,08 enemmän kuin edellisessä, mutta jonon keskimääräinen pituus lyhenee n. 85% ja häntäkin yltää vain viiteen 35:n sijasta, joten yhden kassan lisäämisen hyöty on hyvin merkittävä tässä tilanteessa.

Kahden ylimääräisen kassan tapauksessa jonoja ei enää käytännössä ole: jonon keskimääräisen pituuden häntä yltää vain 1,4:n asti. Kassojenkaan keskimääräinen lukumäärä ei muutu juuri miksikään edelliseen verrattuna. Kuitenkin voi kysyä, onko näin montaa kassaa järkevää ylläpitää, kun kaikki kassat ovat käytössä vain 18% ajasta - jokainen lisäkassa nimittäin tarkoittaa lisäkustannuksia. Käytännössä siis tätä enempää kassoja ei kannata ylläpitää.

Jos itse olisin hampurilaisbaarin pitäjä, menisin yhden ylimääräisen kassan strategialla. Kassojen käyttöaste on tällöin järkevä eivätkä jonot pääse kasvamaan

kohtuuttoman pitkiksi. Kassoja olisi siis tällöin 4 kappaletta.

A Yleinen lähdekoodi

```
function r = exprand( N, lambda )
%exprand Generates exponentially divided random numbers
%   exprand( N, lambda ) returns N random numbers. Lambda is inverse of
%   estimated value.

    r = -1/lambda*log(1 - rand(N, 1));

end

function prnd = poisson( arg )
%poisson returns random number from poisson distribution.
%   poisson( N, arg ) returns random number from
%       poisson distribution with intensity arg.

    prnd = exprand(1, arg);
end

function res = realisation( lambda, t1, t2 )
%realisation returns the realisation of poisson distribution
%   between t1 and t2.

    t = [t1];
    while t(end) < t2
        t(end + 1) = t(end) + poisson(lambda);
    end
    res = length(t) - 2; % Koska poisson-funktiota ollaan kutsuttu length(t) - 1
end
```

B Tehtävä A

```
function [re] = project()
    s = 2;
    j = [0];
    t = 0;
    lambda = 40/60;
    mu = 4;

    while t < 8*60
        arriving = realisation(lambda, t, t + 1);
        leaving = realisation(1/mu_j(mu, s, j(end)), t, t + 1);
        j(end + 1) = max(j(end) + arriving - leaving, 0);
        t = t + 1;
    end
```

```
    stairs(0:t, j)
    xlabel('Aika (h)')
    ylabel('Jonon pituus')
    re = j(end);
end
```

C Tehtävä B

```
function [s, j, maxsprop] = dayinabar (lambda, mu, smax)
% DAYINABAR Simulates one day in a bar.
% Parameters:
%     lambda  = cars coming per unit time
%     mu      = customers leaving per unit time
%     smax    = maximum number of cashes in use at the same time
% Returns:
%     s       = Vector telling the number of cashes open at given time
%     j       = Vector telling the lenghth of the queue at given time
%     maxprop = Proportion of time when all the cashes are in use

    j = 0; % Initialize queue vector
    s = 1; % Initialize open cahses vector
    t = 0; % Initialize time

    % This loop runs the simulation
    while t < 8*60
        % Determine number of arriving people
        arriving = realisation(lambda, t, t + 1);
        % Determine number of leaving people.
        leaving = realisation(1/mu_j(mu, s(end), j(end)), t, t + 1);
        % Add the total number of customers to the end.
        j(end + 1) = max(j(end) + arriving - leaving, 0);
        % Determine number of open cashes and add it to the end.
        s(end+1) = floor(min(max(j(end), 1), smax));
        t = t + 1;
    end

    % Determine the length of the queue by subtracting the number of
    % customers with number of open cashes at a given time.
    j = j-s;
    % No negative queue is possible
    j(j<0) = 0;

    % Determines the proportion of time when all the cashes are in use.
    maxs = s;
    maxs(maxs < smax) = [];
```

```
    maxsprop = length(maxs)/length(s);
end

function [ avgs, avgj, maxsprop ] = montecarlo( lambda, mu, smax, N )
% MONTECARLO Simulates N days in a bar.
% Parameters:
%     lambda = cars coming per unit time
%     mu     = customers leaving per unit time
%     smax   = maximum number of cashes in use at the same time
%     N      = number of days to be simulated
% Returns:
%     avgs   = Vector telling the average number of cashes open at each day
%     avgj   = Vector telling the average lenghth of the queue at each day
%     maxprop = Proportion of time when all the cashes are in use

    % Initialize the vectors
    avgs = [];
    avgj = [];
    maxsprop = [];

    % Run the Monte-Carlo simulation:
    for n = 1:N
        [s, j, maxspr] = dayinabar(lambda, mu, smax);
        avgs(end + 1) = mean(s);
        avgj(end + 1) = mean(j);
        maxsprop(end+1) = maxspr;
    end
    % Solve the proportion of time when all the cashes are in use.
    maxsprop = mean(maxsprop);
end

function [] = project()
    % Runs 4 simulations with different amount of cashes and plots stuff
    % with some statistical data.

    lambda = 40/60; % Number of cars coming per minute
    mu = 4;         % Number of customers leaving per minute
    smax = ceil(mu*lambda); % Minimum for maximum number of cashes,
                            % based on calculations.

    % Run the simulations and plot data.
    figure
    for i = 0:3
        [avgs, avgj, maxsprop] = montecarlo(lambda, mu, smax+i, 1000);
        runningavgs = mean(avgs);
        runningavgj = mean(avgj);
```

```
        subplot(4,3, 3*i+2)
        histogram(avgs)
        str = strcat(sprintf('kokonaiskeskiarvo = %d, kaikki kassat %i', ...
              runningavgs, round(100*maxsprop)), '% k ytss');
        title(str)
        xlabel('Kassoja auki keskimn per p iv')
        ylabel('Frekvenssi')

        subplot(4,3, 3*i+3)
        histogram(avgj)
        str = sprintf('kokonaiskeskiarvo: %d', runningavgj);
        title(str)
        xlabel('Jonon pituus keskimn per p iv')
        ylabel('Frekvenssi')

        subplot(4,3,3*i+1)
        [s, j] = dayinabar(lambda, mu, smax+i);
        stairs(linspace(0, 8, length(j)), j)
        hold on
        stairs(linspace(0, 8, length(s)), s)
  title(sprintf('Esimerkkipaiva, ylimaaraisia kassoja kaytettavissa %i',i))
        xlabel('Aika (h)')
        legend('Jonon pituus', 'Kassoja auki')
    end

end
```