02633 Indledende programmering med MATLAB

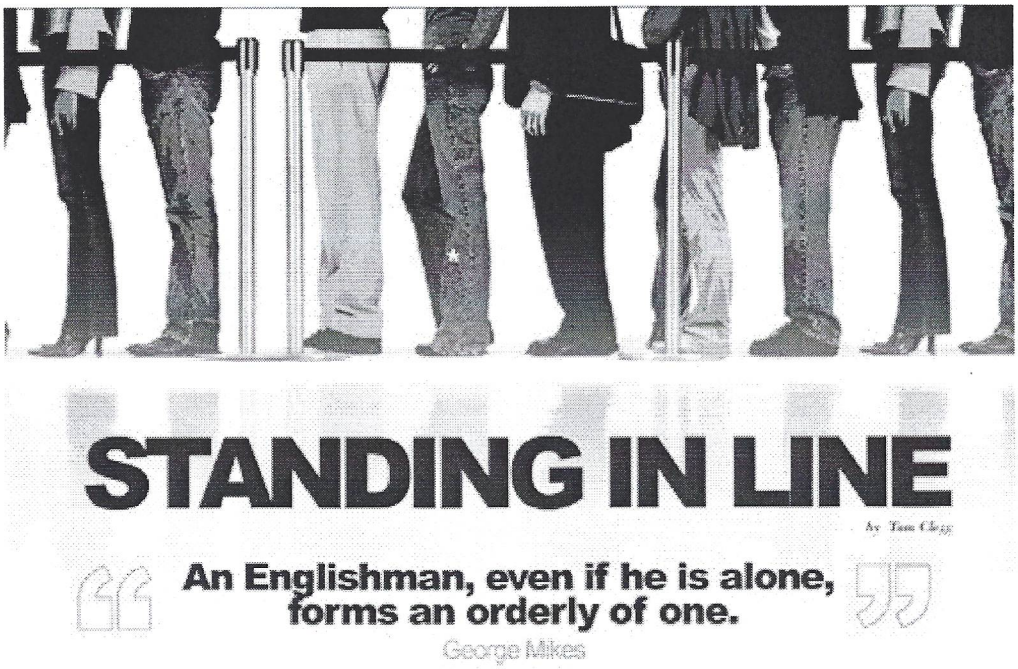
Projekt 1

Simulering af et køsystem

Udført af

Tony Biakceunung Rungling s083422
&
Gustav Collin Rasmussen s053849

Dato: 8. Januar 2010



Programbeskrivelse

Intro

Problemet vi ønsker at løse i projekt 1 er hvorvidt et supermarked (Salling super) kan nøjes med fire ekspeditionskasser eller om flere er nødvendige for at få ekspederet tilstrækkeligt mange kunder. Dette undersøges ved at betragte nogle forskellige forhold under Salling super´s "primetime" ; altså deres travleste periode. Dette tidspunkt strækker sig over 150 minutter om lørdagen. I dette tidsinterval genererer vores super-script målinger for kundernes tid ved indkøbshylderne og ekspeditionstiden, vha. alfa og beta- parametre for henholdsvis kundernes

Indkøbs- og ekspeditionstid som vi tager som parametre i funktionen gammarand.

(den statistiske kontinuerte gammafordeling giver sandsynligheder med en alfa og beta-usikkerhed).

På baggrund af denne sandsynligheds-funktion kan vi simulere forskellige "prime-times" ; altså forskellige lørdage, og herved konkludere om nok kunder i gennemsnit bliver betjent.

Afgørelsen er at finde i nederste afsnit (se Konklusion).

I det følgende vil vi beskrive vores to MATLAB-scripts; nemlig gamparestim- og super- scripterne.

Først ser vi på den første delopgave hvor vi ønsker en tabel opstilt med middelværdier og alfaværdier, og til sidst i tabellen et gennemsnit af disse tal for at se om dette gennemsnit ligger tæt på opgavens angivne middelværdi og alfaværdi.

Gamparestim-scriptet

Dette script hører til den lille delopgave. I Appendix A1 ses en udskrift med programkoden.

Vi starter med en for løkke for at få ti estimater vha. gammarand, og bruger funktionen disp til at udskrive resultaterne vha. en omdannelse fra tal til streng (num2str) til command window (konsollen). Til sidst i koden kaldes disp for at printe gennemsnitsberegningen til konsollen.

Se Appendix A2 for en udskrift af scriptets output.

Super-scriptet

den store delopgave bestod i at gå ind I dette script og foretage en færdiggørelse så scriptet kunne køre simuleringen af supermarkedets kø-udvikling i myldretiden.

Vi starter med at iterere over alle ekspeditioner med en for-løkke. En forklaring til udtrykket [value , index] er som følger: value er den mindste tids-værdi i tids-array´et og index giver hvilken plads i arrayet der først blev ledig. Gives et index-tal mellem 1 og 4 (begge inklusive) så har vi en ledig ekspedient ved denne pågøldende kasse. Dette benyttes i vores if-else-løkke hvor vi enten kan begynde at afvikle køen (under if) eller forlænge den (under else).

Programmet har nu en while-løkke; denne bruges når der stadig er folk i køen og den næste ikke ankommer til køen før den næste ekspedient er ledig.

Til sidst kører super-scriptet det færdige script visresultater, og vi får en figur med grafer over køudviklingen mht. tiden (over de 150 minutter).

Konklusion

Som Appendix B3 viser, i grafen (x,y) = (simuleret tid i minutter,antal kunder i kø)

Så er den resterende kø ved myldretidens afslutning ikke af nogen bemærkelsesværdig størrelse,

Og kommer gennem de fire kørsler af super-scriptet højest op på omkring 10 procent af antallet af kunder i de 150 minutter.

Fire ekspeditionskasser er derfor tilstrækkeligt i vores tilfælde, med de givne tal som stammer fra artiklen

J.P. Lassen, Salling-Super analysen, Ledelse og Erhvervs_konomi/Handelsvidenskabeligt Tidsskrift/Erhvervs_konomisk Tidsskrift, Bind 31 (1967).

Selve programmet er meget brugervenligt, både overfor andre programmører og andre potentielle kunder; det skyldes at vi i stedet for at indsætte konkrete tal, for så vidt muligt søger at indsætte generelle variabelnavne så man kun behøver ændre variablens værdi et sted (I initialiseringen af variablen, som generelt ligger i programmets start)

Et eksempel herpå er variablen kasser i super-scriptet. Her sættes denne til værdien 4 i starten.

Når vi senere benytter variablen, f.eks. under funktionskaldet tider(kasser + 1), så indgår denne variabel ved navn og dens værdi ligger så kun et sted for overskuelighedens skyld. Havde vi ikke valgt denne fremgangsmåde, men skrevet tider(4 + 1) eller endnu værre tider(5) , ville der være

grund til forvirring for andre programmører som så koden med mystiske tal (de kunne spørge sig selv hvor tallet 5 stammer fra).

Ved nogle få og simple ændringer kunne vi anvende vores program til lignende simuleringer, f.eks. køer i lufthavne, supermarkeder med fire kasser og fire køer i stedet for blot en, osv.

Appendices

På de følgende sider ses 5 forskellige appendices, de er indelt som følger:

Appendix A1: gamparestim-scriptet

Appendix A2: output fra gamparestim-scriptet

Appendix B1: super-scriptet

Appendix B2: output fra super-scriptet

Appendix B3: Figur hørende til kørsel nr. 1 af super-scriptet

gamparestim-scriptet:

```
for j = 1:10
i = 1;
L = 0;
K = 0;
x = 0;
for i = 1:10000
    g(i) = gammarand(3.04, 0.341);
    L = g(i)+ L;
    K = log(g(i))+K;
    i = i+1;
end
mean(j) = L/10000;
meanlogg = K/10000;
alfa(j) = fminbnd(@(x)log(gamma(x))-x*(meanlogg-1+log(x/mean
(j))),1,10);
disp(['Estimater ' num2str(j) ': middelværdi = ' num2str(mean(j)) '
alfa = '...
    num2str(alfa(j))])
end
disp('-------------------------------------------------');
disp(['Gennemsnit: middelværdi = ' num2str(sum(mean)/10) ' alfa = '
num2str(sum(alfa)/10) ])
```

Outputtet fra gamparestim-scriptet:

```
Estimater 1: middelværdi = 8.8717 alfa = 3.067
Estimater 2: middelværdi = 8.9619 alfa = 2.9812
Estimater 3: middelværdi = 8.878 alfa = 3.0671
Estimater 4: middelværdi = 8.8815 alfa = 3.0474
Estimater 5: middelværdi = 8.8716 alfa = 3.1083
Estimater 6: middelværdi = 8.9038 alfa = 3.1051
Estimater 7: middelværdi = 8.8947 alfa = 3.0477
Estimater 8: middelværdi = 8.8776 alfa = 2.9718
Estimater 9: middelværdi = 8.9323 alfa = 3.1008
Estimater 10: middelværdi = 8.8579 alfa = 2.993
-------------------------------------------------
Gennemsnit: middelværdi = 8.8931 alfa = 3.0489
>>
```

Super-scriptet:

```
% Dette script SUPER simulerer køsystemet i et supermarked.
%
% Tidsintervallet mellem to kundeankomster,
% tiden for et indkøb (dvs. kundens indsamling af varer) og
% tiden for en kundeekspedition antages alle at være
% gamma(alfa,beta)-fordelte, men med forskellige alfa- og beta-
parametre.

clear all;
% Vi fastlægger først nogle konstanter
%
antalminut = 150;    % simulerer fra kl. 10 til kl. 12.30
kasser = 4;          % antal ekspeditionssteder (som i SUPER-
SALLING,Aarhus)
ankalfa = 1;         % alfa-parameteren for kundeankomst-
tidsintervallet
ankbeta = 4.78;      % beta-parameteren for kundeankomst-
tidsintervallet
koebalfa = 3.04;     % alfa-parameteren for indkøbstiden
koebbeta = 0.341;    % beta-parameteren for indkøbstiden
ekspalfa = 2.94;     % alfa-parameteren for ekspeditionstiden
ekspbeta = 3.60;     % beta-parameteren for ekspeditionstiden
%
% Vi generer nu tidspunktet for den første kunde-ankomst
ank(1) = gammarand(ankalfa,ankbeta);
%
% Kun kunder, der kommer før sluttidspunktet, tælles med
antalfoer = 0;
while ank(antalfoer+1) < antalminut,
    antalfoer = antalfoer+1;
    ank(antalfoer+1) = ank(antalfoer) + gammarand(ankalfa,ankbeta);
end;
%
% Vi får ankomsttider til køen ved at addere købstiderne
for i = 1:antalfoer,
    ank(i) = ank(i) + gammarand(koebalfa,koebbeta);
end;
%
% Ankomsttiderne til køen sorteres i voksende rækkefølge
% ved at kalde MATLAB-funktionen sort
ank = sort(ank);
%
% Kommer kunden efter sluttidspunktet,
% simuleres ekspeditionen af kunden ikke
antaleksp = antalfoer;
while ank(antaleksp) > antalminut,
    antaleksp = antaleksp - 1;
end;
disp(['antal ankomster til supermarked: ',num2str(antalfoer),...
    ', antal ekspeditioner: ',num2str(antaleksp)]);
%
% Fælleskøen består af to arrays, hvor
% koe.antal(koeevents) = nuværende antal kunder i fælleskøen.
% koe.eventtid(koeevents) = tidspunktet, da køantallet
%                           blev det nuværende antal.
% koeevents tælles op, hver gang køen ændres. Den starter med at være
1.
koe = struct('antal', 0, 'eventtid', 0);
koeevents = 1;
%
tider=zeros(kasser+1,1);
```

```
% Vi laver en for-løkke indeholdende alle de ekspederede kunder.
for nr = 1:antaleksp;
    % Da ankomsttiden til køen er sidst i array-et tider, definerer
vi en
    % funktion der finder minimum af tider til at finde den først
ledige
    % ekspedient.
    tider(kasser+1,1) = ank(nr);
    [value,index] = min(tider);

    % Vi afgør om der er en ekspedient ledig:
    if index <= kasser;
        % Da en ekspedient er ledig, lægger vi ekspeditionstiden til
den
        % ekspederende kasse sammen med ankomttiden til kassen.
        tider(index) = gammarand(ekspalfa,ekspbeta) + ank(nr);
    else
        % Da der ikke er nogen ledig ekspedient, sættes kunden i
køen.
        koeevents = koeevents + 1;
        koe.antal(koeevents) = koe.antal(koeevents - 1) + 1;
        koe.eventtid(koeevents) = ank(nr); % kø-tiden noteres.

        % Her findes en ny minimumsværdi til at finde ekspedienten
der
        % bliver hurtigst ledig.
        [value,index] = min(tider(1:kasser));

        % Der er stadig folk i køen og den næste ankommer ikke til
køen før
        % den næste ekspedient er ledig
        while koe.antal(koeevents) > 0 ...
                && (nr == antaleksp || value < ank(nr + 1));

            % Vi noterer køevent, kølængde og køeventtid igen og
lader
            % programmet regne en ny minimumsværdi til at finde den
næste
            % ledige ekspedient.
            tider(index) = gammarand(ekspalfa,ekspbeta) + value;
            koeevents = koeevents + 1;
            koe.antal(koeevents) = koe.antal(koeevents - 1) - 1;
            koe.eventtid(koeevents) = value;
            [value,index] = min(tider(1:kasser));
        end
    end
end;

% Nu køres scriptet visresultater.
visresultater(koe,koeevents,antalminut,'dansk  ');
```

Her ses outputtet til command window fra fire kørsler af super-scriptet:

```
antal ankomster til supermarked: 671, antal
ekspeditioner: 631
antal ankomster til supermarked: 723, antal
ekspeditioner: 673
antal ankomster til supermarked: 732, antal
ekspeditioner: 685
antal ankomster til supermarked: 735, antal
ekspeditioner: 681
>>
```

Her ses graferne tilhørende kørsel nr. 1 af super-scriptet.