Wireless Communication Systems HW1

通訊所一年級 110064533 陳劭珩

October 18 2021

1. Question one

Implement the Erlang-B formula, and calculate the total offered traffic load for the following system parameters:

- **Channel number: 1∼20, 200∼220**
- **Blocking rate: 1%, 3%, 5%, 10%**

1.1 Answer

Blocking Prob.

$$B(\rho, m) = \frac{\rho^m}{m! \sum_{k=0}^{m} \frac{\rho^k}{k!}}$$

- m is the total number of channels in the trunk
- $\rho = \lambda\mu$ is the total offered traffic
  - λ is the call arrival rate (Poisson call arrivals)
  - μ is the mean call duration (exponentially distributed)
- $G_c = \rho/m$ is the offered traffic per channel

第一題要算給定條件下的 total offered traffic load，從上述公式可以看到，如果要把 ρ 完全整理到式子的左邊或右邊並帶入條件求解，應該是沒辦法做到。所以先試試看用 brute force 把 ρ 試出來，當式子等號兩邊的值非常接近，但又不超過 blocking rate 時的 ρ 就是我們要找的結果，由於上課投影片給的 erlang-B table 的數值是到小數點後第 4 位，所以把每次試的 step size 大小先設為 0.0001。

結果馬上遇到一個大問題，如果直接照公式一個一個帶，不用到 200! 光是 171! 就因為數字太大算不出來。

MATLAB 的最大整數極限應該是

$$double(factorial(170)) = 7.2574e^{306}$$

再更大的階乘就只會顯示 Inf 完全算不了。為此一開始先嘗試寫大數運算以解決最大整數受限的問題，但試算一下發現數字真的超大，礙於程式能力不足用大數運算暴力解只是徒增困擾，改嘗試從簡化數學式下手。

發現可以把 $m!$ 提到分子變成 $\rho^m/m!$，分母變成等比級數 $\sum_{k=0}^{m} \rho^k/k!$，式子整理成

$$B(\rho, m) = \frac{\frac{\rho^m}{m!}}{\sum_{k=0}^{m} \frac{\rho^k}{k!}} = \frac{\frac{\rho^m}{m!}}{1 + \frac{\rho}{1!} + \frac{\rho^2}{2!} + ... + \frac{\rho^m}{m!}} = \frac{numerator}{denominator}$$

這樣算起來最大的 $denominator$ 差不多是 $7.2029e^{101}$ 不會超出 MATLAB 極限。

1.2 Code

試算一下發現，200! 有 375 位，220! 有 422 位，如果要暴力硬解感覺有點複雜。

factorial.cpp

```cpp
#include <bits/stdc++.h>
// using namespace std;

//User function template for C++
class Solution {
public:
    std::vector<int> factorial(int N) {
        std::vector<int> result;
        result.push_back(1);
        for (int x = 2; x <= N; x++) {
            int carry = 0;
            for (int i = 0; i < (int)result.size(); i++) {
                int prod = result[i] * x + carry;
                result[i] = prod % 10;
                carry = prod / 10;
            }
            while (carry != 0) {
                result.push_back(carry % 10);
                carry /= 10;
            }
        }
        std::reverse(result.begin(), result.end());
        return result;
    }
};
```

```
// { Driver Code Starts.
int main() {
    int t;
    std::cin >> t;
    std::cout << "number of instances: " << t << "\n";
    while (t--) {
        int N;
        std::cin >> N;
        Solution ob;
        std::vector<int> result = ob.factorial(N);
        std::cout << "factorial of " << N << ": ";
        for (int i = 0; i < (int)result.size(); ++i){
            std::cout<< result[i];
        }
        std::cout << "\nnumber of digits: " << result.size() << "\n";
    }
    std::cout << std::endl;
    return 0;
}  // } Driver Code Ends
```

$./factorial.exe$

$D : /C++ > ./factorial.exe$

$numberofinstances : 2$

$factorialof200 : 788657867364790503552363213932185062295135977687173263294742533244359449963403342920304284011984623904177212138919638830257642790242637105061926624952829931113462857270763317237396988943922445621451664240254033291864131227428294853277524242407573903240321257405579568660226031904170324062351700858796178922227896237038973747200000000000000000000000000000000000000000000000000000000000000000000000$

$numberofdigits : 375$

$factorialof220 : 2283860335914641457397265865115333727042973071546228701773634716126027692603024845877776549791921102945706558196074779575009550523224197049956176972302056587667226166060976323404977554732543013557133146825747553799450849523377065894531021055272516334278466875614904921365807833845853428557155180084957884822642989867003294551385992993862178352349027264696691854493614080000000000000000000000000000000000000000000000000000000000000000000000000000$

$numberofdigits : 422$

$HW1\_110064533\_1.m$

```
%
% Wireless Communication Systems HW1, 通訊所一年級 110064533 陳劭珩
%
% calculate the total offered traffic load ρ
%
clear;
clc;
%
rho_table = zeros(220, 4); % the result of total offered traffic load ρ
step_size = 0.0001;
%
% Blocking Rate: 1%, 3%, 5%, 10%
%
blocking_rate = [0.01 0.03 0.05 0.1];

%
% brute force
%
for rate_index = 1 : 4
    %
    % Channel Nuber: m = 1~20
    %
    for channel_number = 1 : 20
        m = channel_number;  % the current channel number m
        rho = 0;             % the total offered traffic load ρ
        blocking_prob = 0;   % the blocking prob B(ρ, m)
        while blocking_prob < blocking_rate(1, rate_index)
            rho = rho + step_size*m;
            numerator = 1;   % ρ^m / m!
            denominator = 1; % sigma_k=0~m {ρ^k / k!}
            for k = 1 : m
                numerator = numerator * (rho / k);
                denominator = denominator + numerator;
            end
            blocking_prob = numerator / denominator;
        end
        % 當逼近的blocking_prob超過題目規定的blocking rate時會跳出while loop
        % 想要的是最接近但不超過的ρ 也就是超過條件的前一個ρ
        rho_table(m, rate_index) = rho - step_size*m; % 並將結果存至ρ table
    end
```

```
    %
    % Channel Nuber: m = 200~220
    %
    for channel_number = 200 : 220
        m = channel_number;  % the current channel number m
        rho = 0;             % the total offered traffic load ρ
        blocking_prob = 0;   % the blocking prob B(ρ, m)
        while blocking_prob < blocking_rate(1, rate_index)
            rho = rho + step_size*m;
            numerator = 1;   % ρ^m / m!
            denominator = 1; % sigma_k=0~m {ρ^k / k!}
            for k = 1 : m
                numerator = numerator * (rho / k);
                denominator = denominator + numerator;
            end
            blocking_prob = numerator / denominator;
        end
        % 當逼近的blocking_prob超過題目規定的blocking rate時會跳出while loop
        % 想要的是最接近但不超過的ρ 所以減去1單位channel_number的step_size
        rho_table(m, rate_index) = rho - step_size*m; % 並將結果存至ρ table
    end
end
%
% save the result as a csv file, named 'rho_table.txt'
csvwrite('rho_table.txt', rho_table);
```

1.3 Erlang-B table

Table 1 Channel Number m = 1∼20

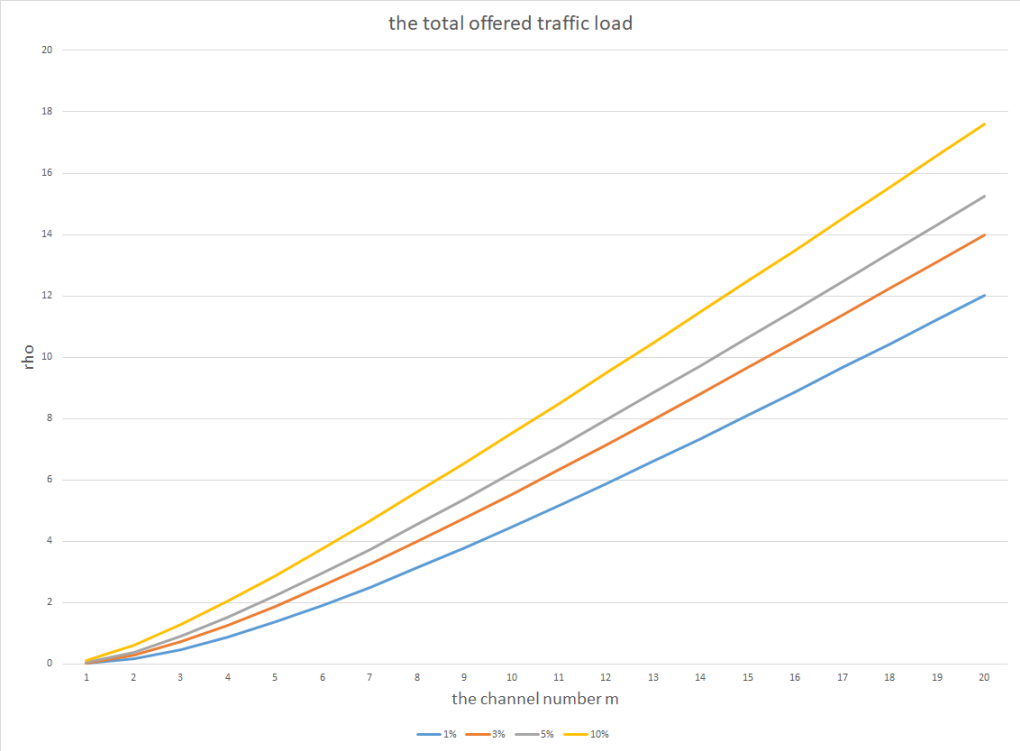
(m, B)	1%	3%	5%	10%
1	0.0101	0.0309	0.0526	0.1111
2	0.1524	0.2814	0.3812	0.5954
3	0.4554	0.7149	0.8991	1.2705
4	0.8692	1.2588	1.5244	2.0452
5	1.3605	1.8750	2.2180	2.8810
6	1.9086	2.5428	2.9598	3.7584
7	2.5004	3.2494	3.7373	4.6655
8	3.1272	3.9864	4.5424	5.5968
9	3.7818	4.7475	5.3694	6.5457
10	4.4610	5.5290	6.2150	7.5100
11	5.1590	6.3272	7.0763	8.4865
12	5.8752	7.1400	7.9500	9.4740
13	6.6066	7.9664	8.8348	10.469
14	7.3514	8.8032	9.7286	11.473
15	8.1075	9.6495	10.632	12.483
16	8.8736	10.504	11.542	13.499
17	9.6509	11.368	12.461	14.521
18	10.436	12.238	13.385	15.547
19	11.229	13.114	14.315	16.577
20	12.030	13.996	15.248	17.612

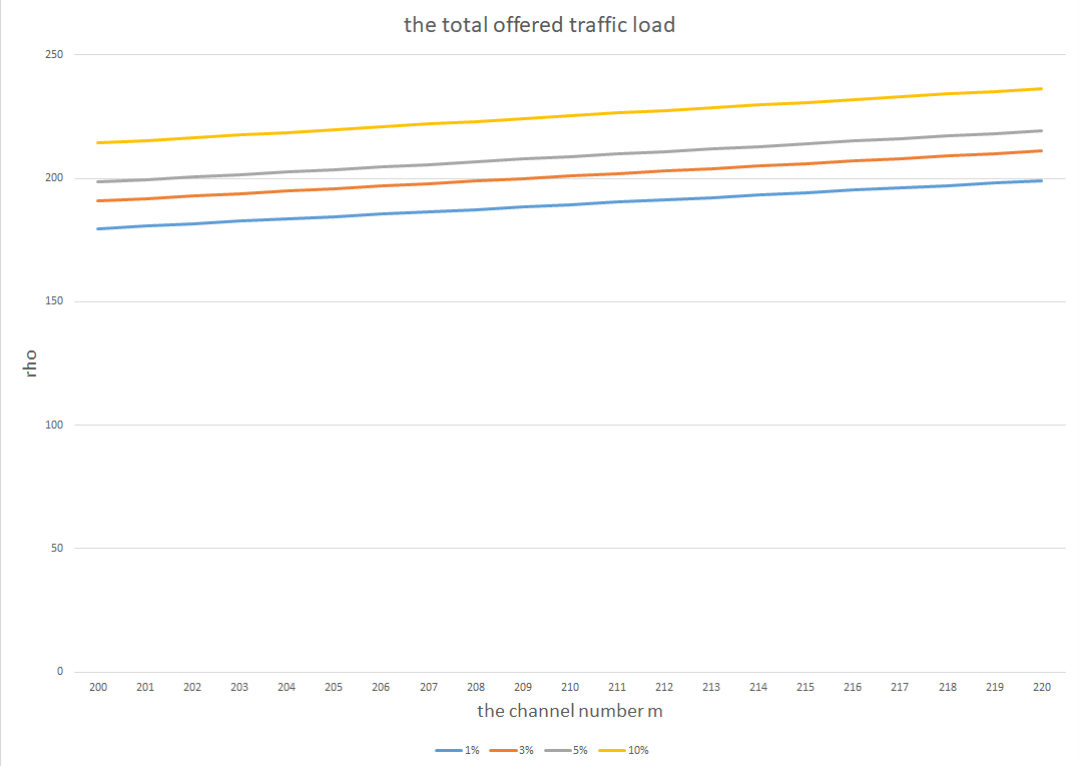
Table 1: Erlang-B table for channel number m = 1∼20

Table 2 Channel Number m = 200∼220

(m, B)	1%	3%	5%	10%
200	179.72	190.88	198.50	214.32
201	180.70	191.87	199.53	215.41
202	181.66	192.89	200.57	216.52
203	182.64	193.91	201.62	217.64
204	183.60	194.9	202.65	218.73
205	184.56	195.92	203.69	219.84
206	185.54	196.94	204.72	220.94
207	186.51	197.93	205.76	222.05
208	187.47	198.95	206.81	223.16
209	188.43	199.95	207.85	224.26
210	189.42	200.97	208.89	225.37
211	190.39	201.97	209.92	226.47
212	191.35	202.99	210.96	227.58
213	192.32	203.99	212.0	228.68
214	193.28	205.01	213.04	229.79
215	194.25	206.01	214.08	230.89
216	195.22	207.01	215.11	232.01
217	196.19	208.04	216.15	233.10
218	197.18	209.04	217.19	234.22
219	198.15	210.04	218.23	235.32
220	199.12	211.07	219.27	236.41

Table 2: Erlang-B table for channel number m = 200∼220


the total offered traffic load
rho
the channel number m
1%
3%
5%
10%


the total offered traffic load
rho
the channel number m
1%
3%
5%
10%

2. Question two

(a) Could it be possible that the total offered traffic load is larger than the number of available channels? Why?

(b) How to determine the traffic that has been served?

2.1 Answer

(a) 直觀上由第一題的結果看起來" 可以"，因為 Table 2 最後一直行的計算結果，也就是 $m = 200 \sim 220, B(\rho, m) = 10\%$ 的時候，看起來算得的 ρ 都比 m 還大，但是因為"total" offered traffic load 有包含 blocked 跟 served 的部分:

$$\rho_{total} = \rho_{served} + \rho_{blocked}$$

所以實際上超過的部分應該會被屏蔽掉，沒有提供服務，但還是能算有接收到。

(b) 必須先扣掉 $\rho_{blocked}$ 的部分，剩下才是實際上有提供服務的。

$$\rho_{served} = \rho_{total} - \rho_{blocked} = \rho_{total} * (1 - B(\rho, m))$$

3. Question three

Assume that there are 600 channels equally shared by 1) one, 2) two, or 3) three operators by using the frequency reuse factor N = 5.

- **Find the maximum offered traffic load per cell for the three cases with the blocking rate equal to 1%, 3%, 5%, or 10%**
- **Which case (one, two, or three operators) is more efficient?**

3.1 Answer

由關係式

$$number\_of\_channels\_per\_cell = \frac{number\_of\_total\_channels}{(number\_of\_operators) * (frequency\_reuse\_factor)}$$

可以推得 3 種情況下的等效 channel number 並將結果列於 Table3

Table 3 Channel number per cell

	1 operator	2 operators	3 operators
channel number per cell	120	60	40

Table 3: Channel number per cell for the three cases

3.2 Code

從題目條件換算得到等效的 channel number 如 Table3 所示，並稍加修改第一題的 code 即可帶入求解。

$HW1\_110064533\_3.m$

```
%
% Wireless Communication Systems HW1, 通訊所一年級 110064533 陳劭珩
%
% 3.Assume that there are 600 channels equally shared by 1)one, 2)two,
%   or 3)three operators by using the frequency reuse factor N = 5.
%   - Find the maximum offered traffic load per cell for the 3 cases with
%     the blocking rate equal to 1%, 3%, 5%, or 10%
%   - Which case (one, two, or three operators) is more efficient?
%
clear;
clc;
%
total_channels = 600;          % the number of total channels
operators = [1 2 3];           % the number of operators
N = 5;                         % the frequency reuse factor N
channel_per_cell = [0 0 0];    % the number of channels per cell of 3 cases
rho_table3 = zeros(3, 4);      % the result of maximum offered traffic load
step_size = 0.0001;
%
% initialize the number of channels per cell of the 3 cases
for i = 1 : 3
    channel_per_cell(i) = total_channels / N / operators(i);
    fprintf("%d operator shared %d channels\n", i, channel_per_cell(i));
end
%
% Blocking Rate: 1%, 3%, 5%, 10%
%
blocking_rate = [0.01 0.03 0.05 0.1];
%
```

```
% brute force
%
for rate_index = 1 : 4
    for m = 1 : 3 % the channel number index m
        rho = 0;              % the maximum offered traffic load ρ
        blocking_prob = 0;    % the blocking prob B(ρ, m)
        while blocking_prob < blocking_rate(1, rate_index)
            rho = rho + step_size*channel_per_cell(1, m);
            numerator = 1;    % ρ^m / m!
            denominator = 1; % sigma_k=0~m {ρ^k / k!}
            for k = 1 : channel_per_cell(1, m)
                numerator = numerator * (rho / k);
                denominator = denominator + numerator;
            end
            blocking_prob = numerator / denominator;
        end
        % 當逼近的blocking_prob超過題目規定的blocking rate時會跳出while loop
        % 要的是最接近但不超過的ρ 也就是超過條件的前一個 並將結果存至ρ table3
        rho_table3(m, rate_index) = rho - step_size*channel_per_cell(1, m);
    end
end
%
% save the results as a csv file, named 'rho_table3.txt'
csvwrite('rho_table3.txt', rho_table3);
```

3.3 Erlang-B table

Table 4 Maximum traffic load per operator

(amount, B)	1%	3%	5%	10%
1 operator	102.960	110.640	115.764	126.072
2 operators	46.9440	51.5640	54.5640	60.3960
3 operators	29.0040	32.4080	34.5920	38.7840

Table 4: Maximum offered traffic load per operator for the three cases

Table 5 Maximum traffic load per cell

(amount, B)	1%	3%	5%	10%
1 operator	102.960	110.640	115.764	126.072
2 operators	93.8880	103.128	109.128	120.792
3 operators	87.0120	97.2240	103.776	116.352

Table 5: Maximum offered traffic load per cell for the three cases

Table 6 Offered traffic load per channel G_c

(amount, B)	1%	3%	5%	10%
1 operator	0.8580	0.9220	0.9647	1.0506
2 operators	0.7824	0.8594	0.9094	1.0066
3 operators	0.7251	0.8102	0.8648	0.9696

Table 6: Offered traffic load per channel $G_c = \rho/m$ for the three cases

由 Table4∼6 整理的結果可以看到，在固定 total channel number 及固定 reuse factor 的情況下，如果 operator 的數量越多，就會導致每個 operator、每個 cell 實際分到的 channel 越少。

當每個 server 都處於忙碌時，不同的 operator 不能使用其他的 server 而造成效率變差，因此覺得在僅有 1 個 operator 的情況下，可分配使用的 channel 較多、ρ 也較大，所以是效能最好、最有效率的選擇。