Program Współpraca Polska-RPA	**RAPORT CZĄSTKOWY z realizacji projektu w ramach programu międzynarodowego Polska-RPA**			Narodowe Centrum Badań i Rozwoju
Nr raportu	IR-RATfor5G-06			
Data aktualizacji raportu:	2022.01.01		**Wersja**	6
Numer umowy	PL-RPA2/02/RATfor5G+/2019		**Akronim**	RATfor5G+
Okres realizacji projektu	od	2019.01.01	do	2022.06.30
Tytuł projektu	Technologie dostępu radiowego dla standardu 5G i przyszłych generacji sieci bezprzewodowych			
Tytuł raportu	Wyniki symulacji – cz.2			

Niniejszy raport przedstawia poszczególne kroki na drodze do weryfikacji mechanizmów, o które rozszerzony został symulator 5G SLS Vienna z perspektywy zarządzania zasobami w sieciach NOMA w powiązaniu z minimalizacją interferencji. W załączniku **Aneks1** udostępnione zostały elementy kodu zmodyfikowane przez zespół projektowy, które umożliwiają realizację funkcji kontroli przyjmowania zgłoszeń (*admission control*). Pozostałe modyfikacje są dostępne na żądanie pod adresem mailowym adamfli@pbs.edu.pl. Poniższe kroki przedstawiają szczegółowy przebieg kroków podjętych w celu rozpoznania i modyfikacji rozbudowanego kodu symulatora w zakresie niezbędnym do realizacji zadań związanych z algorytmami admission control oraz NOMA radio resource scheduling. Niniejsze opracowanie jest komplementarne raportu IR-RATfor5G-04.

Testy wstępne

Poniższe testy (tj. próby: **Try1,2,3**) zostały zrobione dla połączeń trwających 10x dłużej niż by to wynikało z rozkładu wykładniczego (holding).

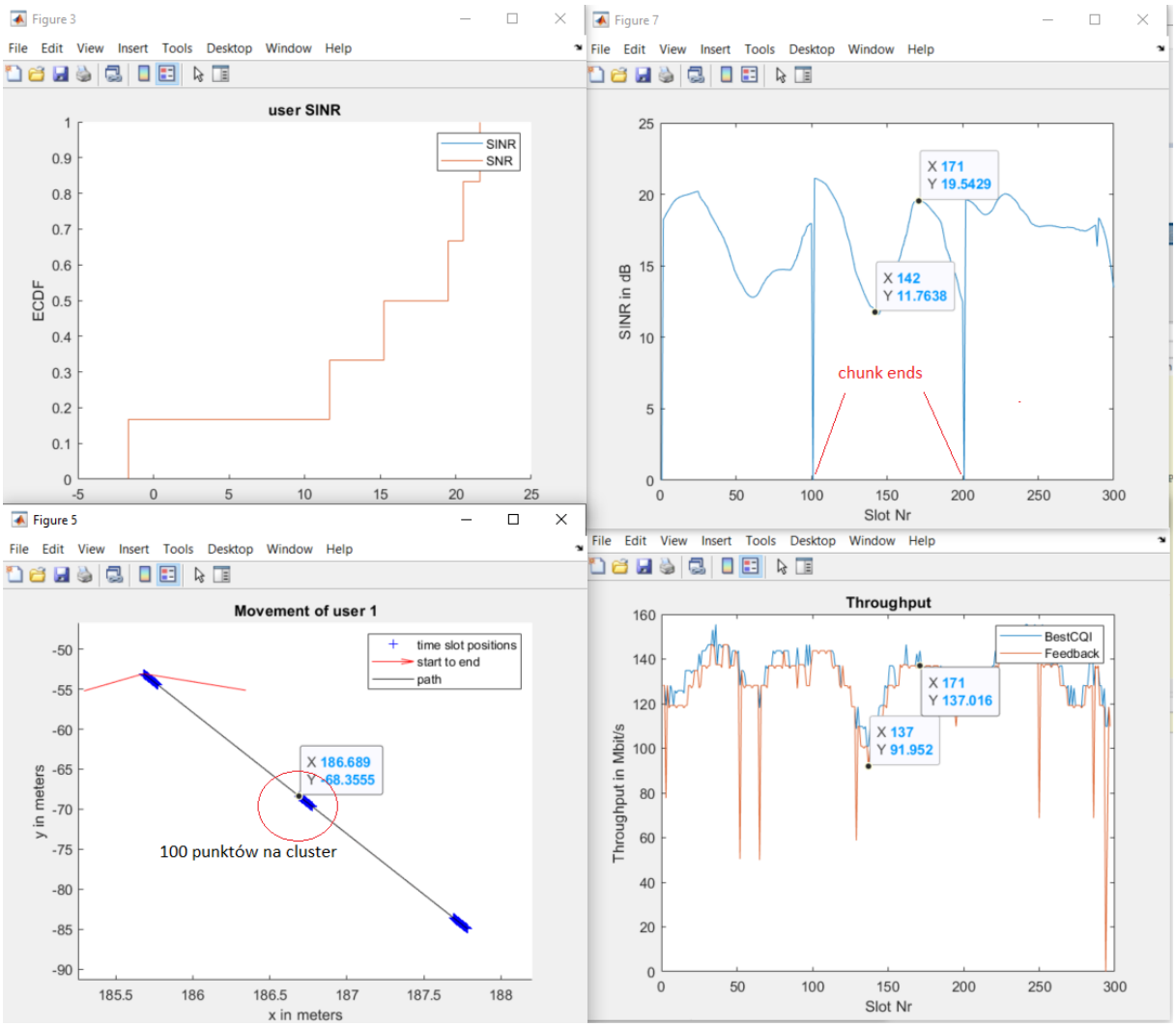
```
if (~isempty(tmp))
                    a = [obj.arrival.T(tmp(1)), obj.arrival.Th(tmp(1)),
obj.arrival.class(tmp(1))]; % always take the first record
                    %if(slot<obj.duration)
                        obj.setSessionTiming(a(1), (a(2)-a(1))*10); % we set the NEW session
based on current reading
```

1. launcherNOMA (Try1)

chunkSimulation	`poissonGenerator=0;`

feedbackBasic	`%params.time.numberOfChunks = 5;`
	`params.time.numberOfChunks = 3;`
	
	`params.time.slotsPerChunk = 100; % each chunk consists of that many slots`
	
	TrafficModel = **FullBuffer**

launcherFeedback	No changes

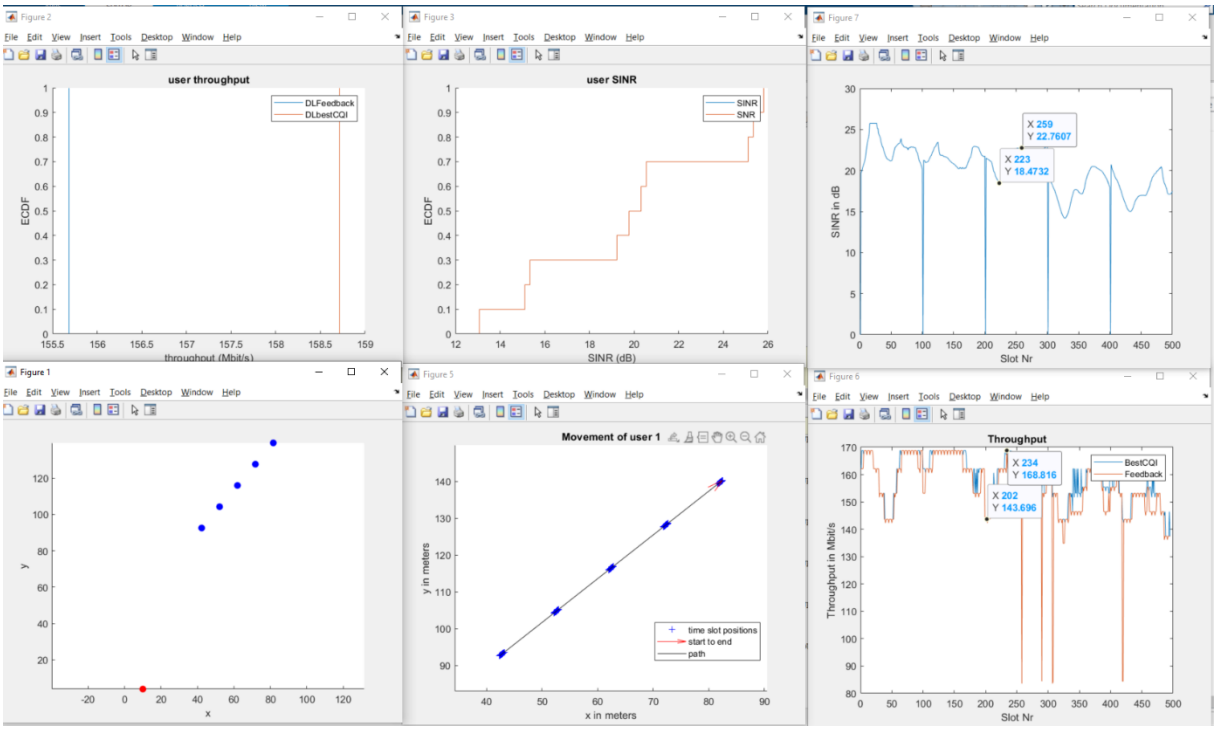


2. LauncherNoma (Try2)

chunkSimulation	`poissonGenerator=0;`

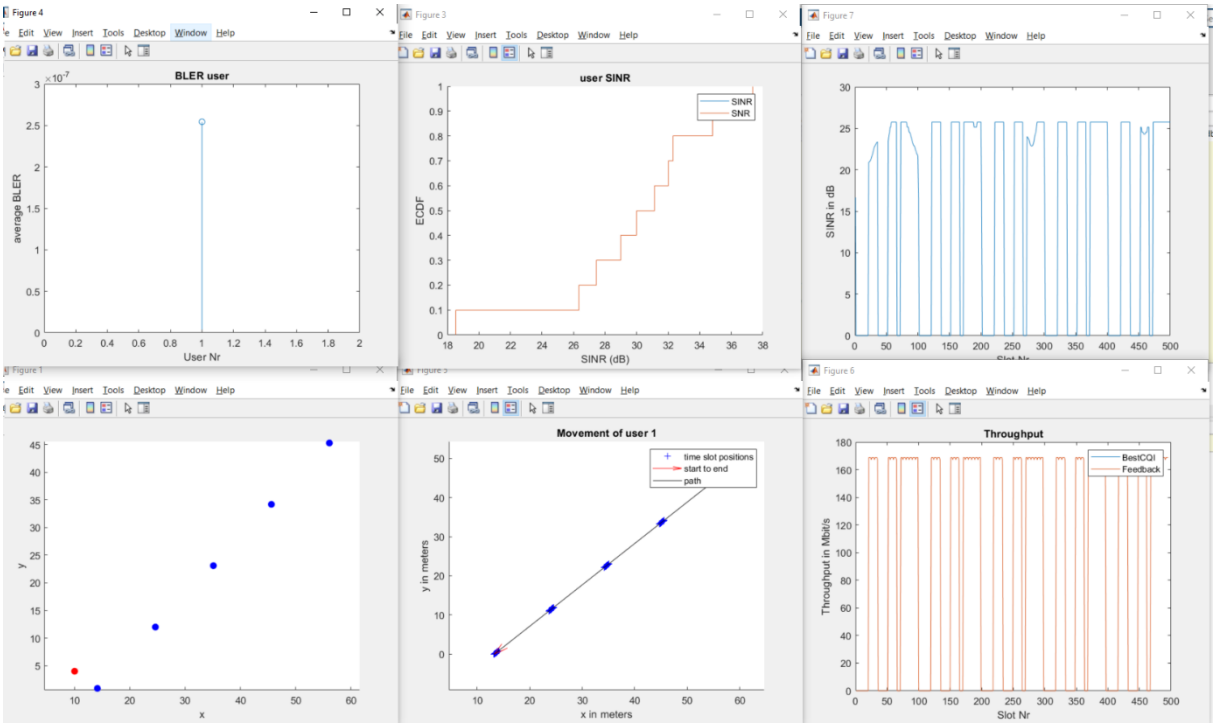
feedbackBasic	`params.time.numberOfChunks = 5;`
	TrafficModel = **FullBuffer**
	
	`reszta bez zmian (pod kątem Try1)`

launcherFeedback	No changes

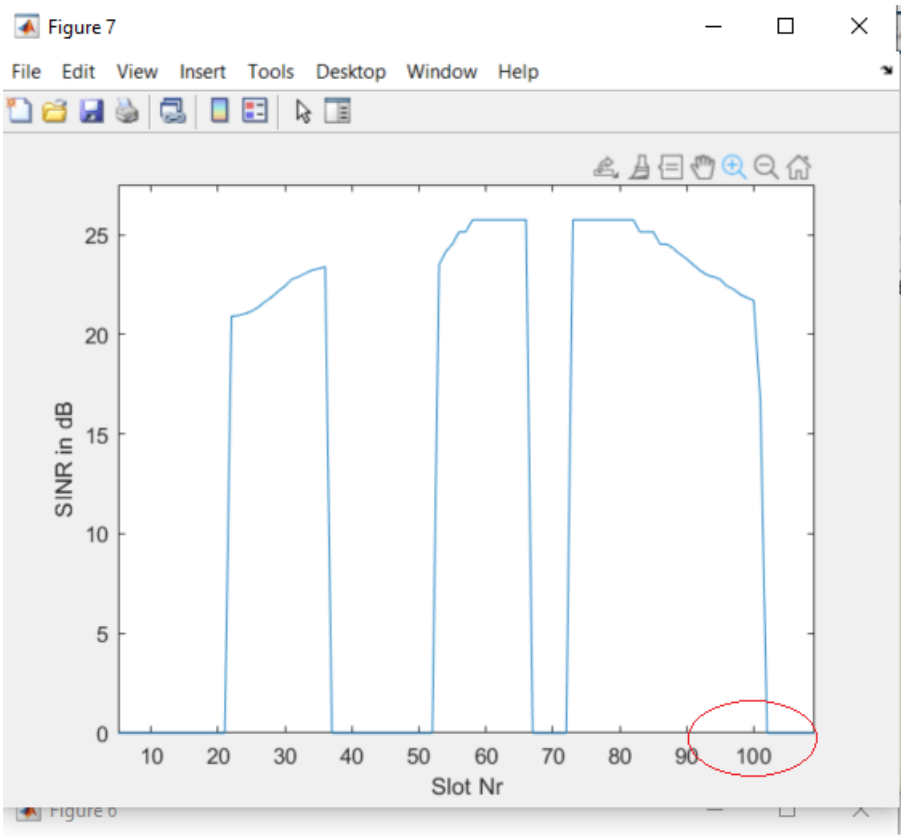


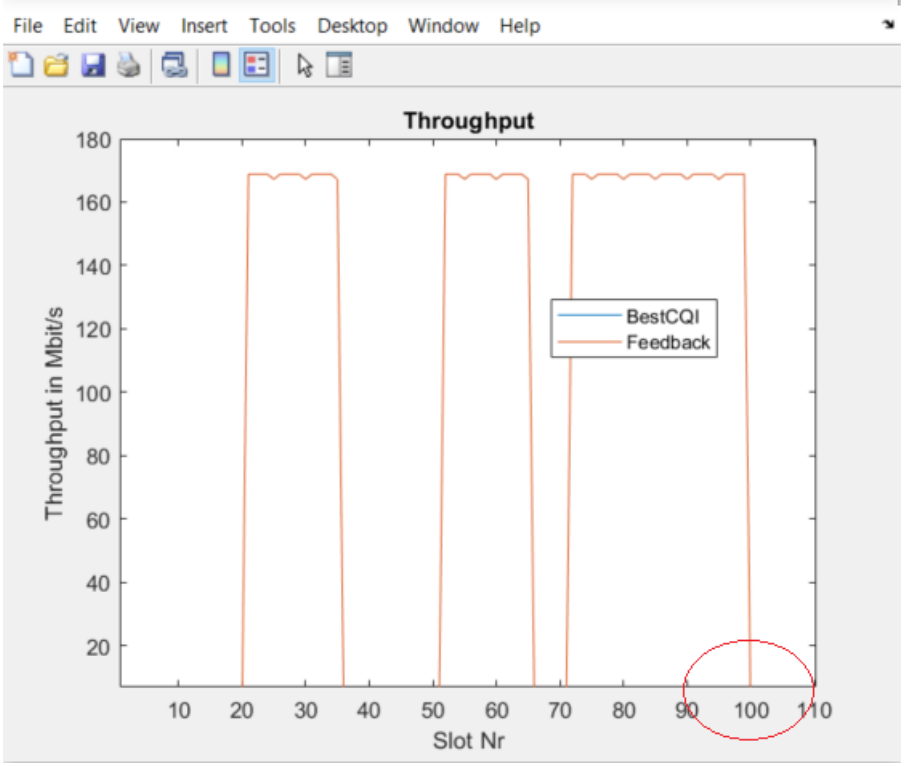
3. LauncherFeedback (**try3**)

Główna zmiana w stosunku do poprzedniego podejścia “Try2”: poissonGenerator =1 (sesje generowane za pomocą rozkładu Poissona + exponential holding time).



Poniżej zoom na pierwsze 100 slotów (=50 ramek * 10ms = 0,5sekundy). Widać że są trzy sesje połączeniowe.





Admission control w następujących momentach

----------> Admission control dla user1. slot=22

----------> Admission control dla user1. slot=53

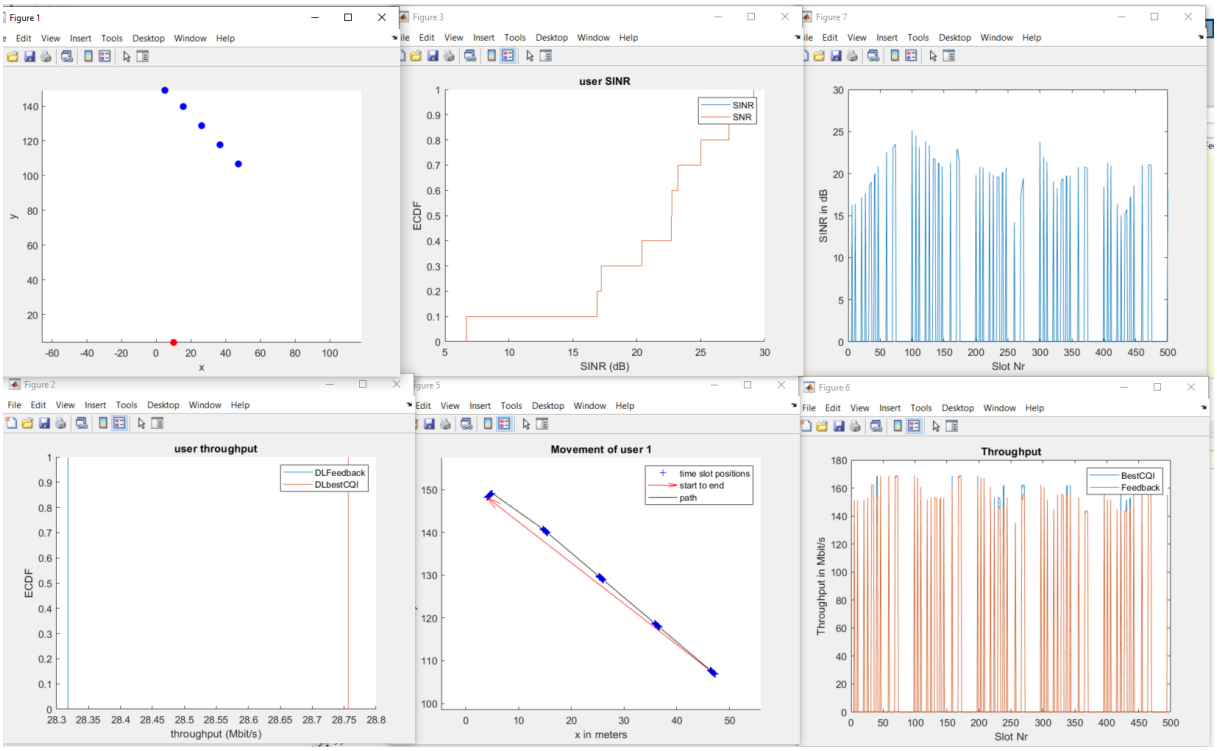
----------> Admission control dla user1. slot=73

4. LauncherFeedback (**Try4**)

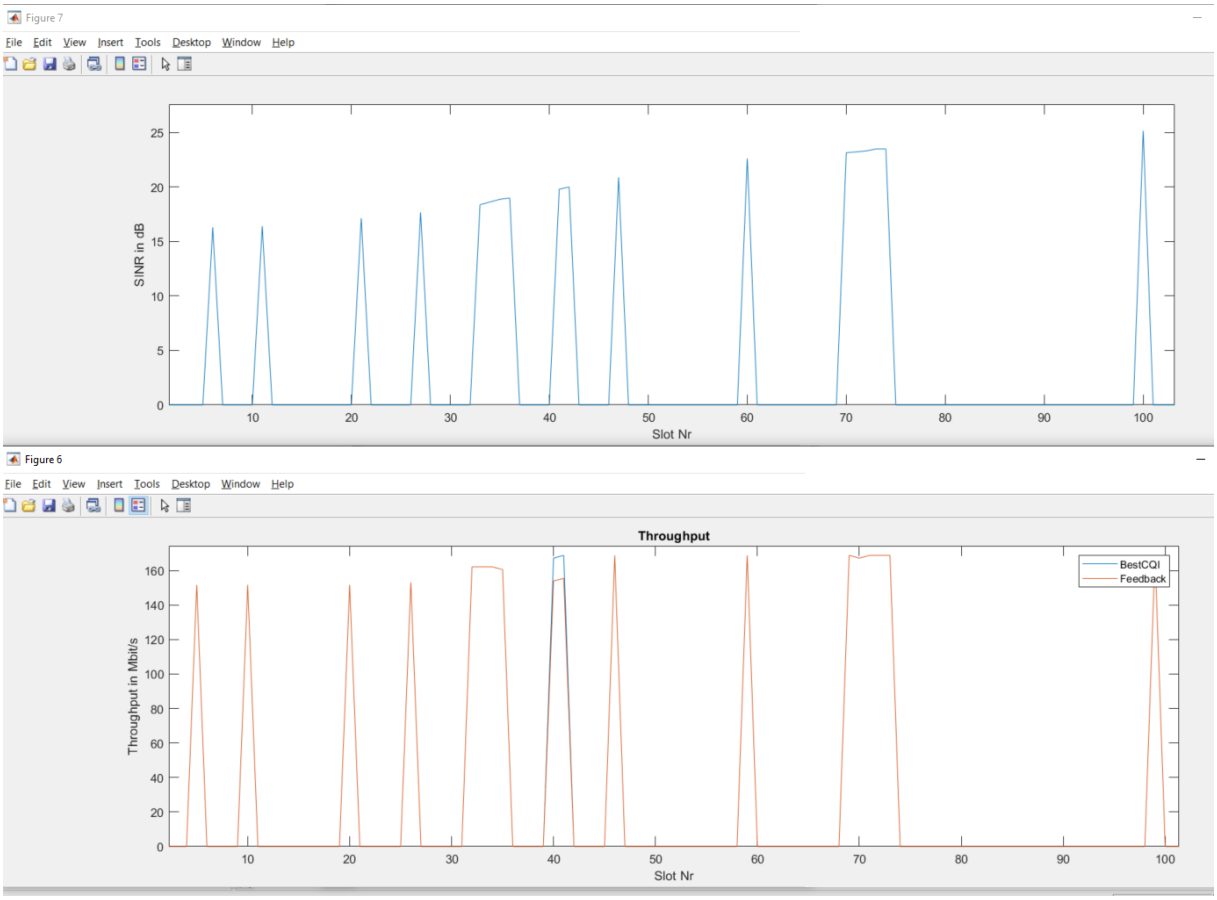
A teraz to samo co powyżej (Try3) ale jak sesje skrócimy o „10”

chunkSimulation	`poissonGenerator=1;`
feedbackBasic	`params.time.numberOfChunks = 5;` TrafficModel = **FullBuffer** reszta bez zmian (pod kątem Try1)
launcherFeedback	No changes

```
if (~isempty(tmp))
                    a = [obj.arrival.T(tmp(1)), obj.arrival.Th(tmp(1)),
obj.arrival.class(tmp(1))]; % always take the first record
                    %if(slot<obj.duration)
                        obj.setSessionTiming(a(1), (a(2)-a(1))*10); % we set the NEW session
based on current reading
```



Zoom na pierwsze 100-slotów:



Admissions:

----------> Admission control dla user1. slot=6 (1 slot!)

----------> Admission control dla user1. slot=11 (1 slot!)

----------> Admission control dla user1. slot=21 (1 slot!)

----------> Admission control dla user1. slot=22 (1 slot!)

----------> Admission control dla user1. slot=27 (1 slot!)

----------> Admission control dla user1. slot=33 (4 sloty)

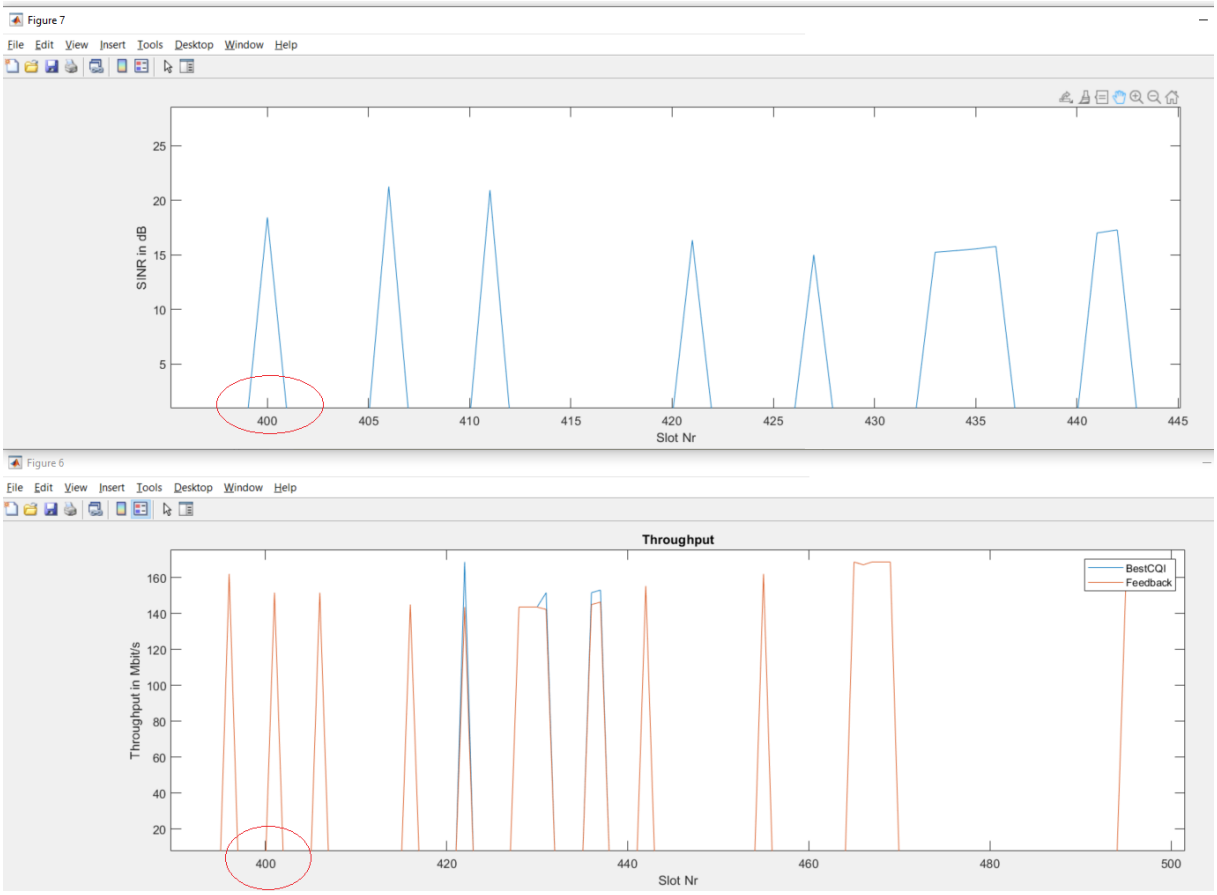
----------> Admission control dla user1. slot=41 (2 sloty)

----------> Admission control dla user1. slot=47

----------> Admission control dla user1. slot=60

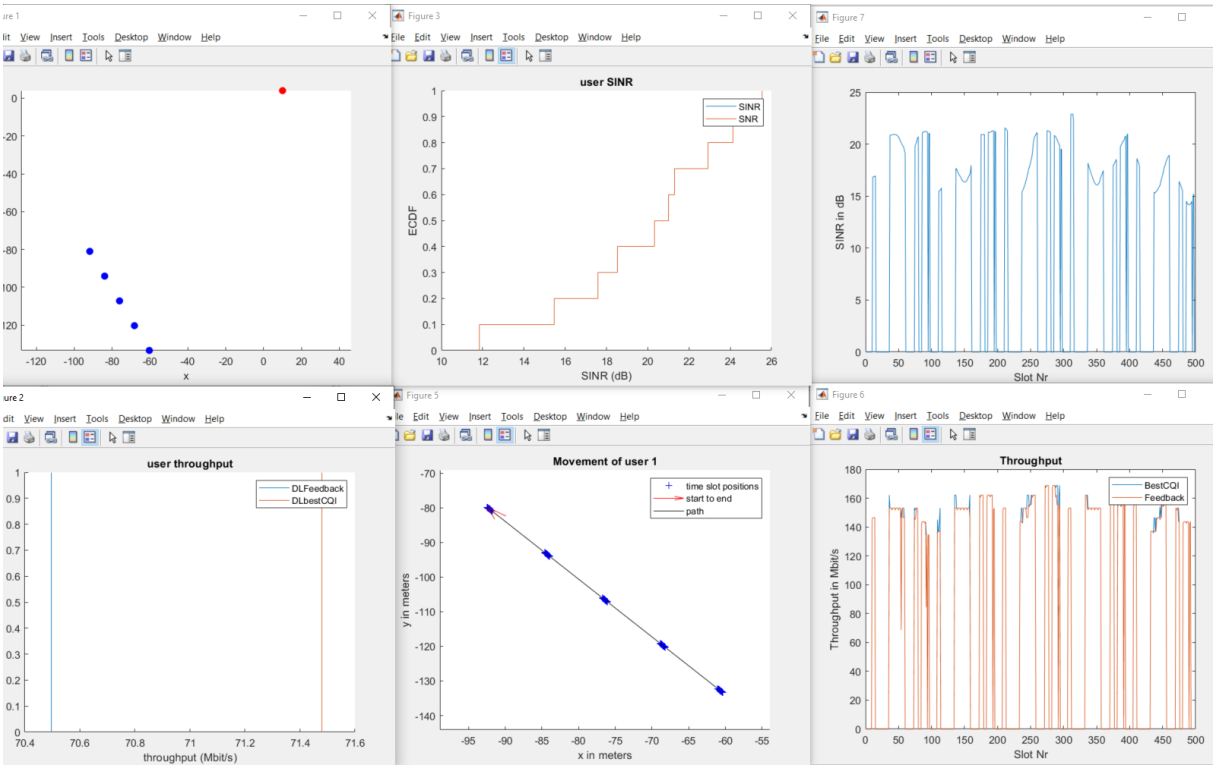
----------> Admission control dla user1. slot=70 (5 slotów)

INFO: jak widać w przypadku powtarzania chunk'ów, sesje userów nie są generowane od nowa. Poniżej zrzut ostatnich 100 slotów (400-500). Jak widać momenty przyjścia userów są takie same jak dla 100-200. Czy tak powinno być?

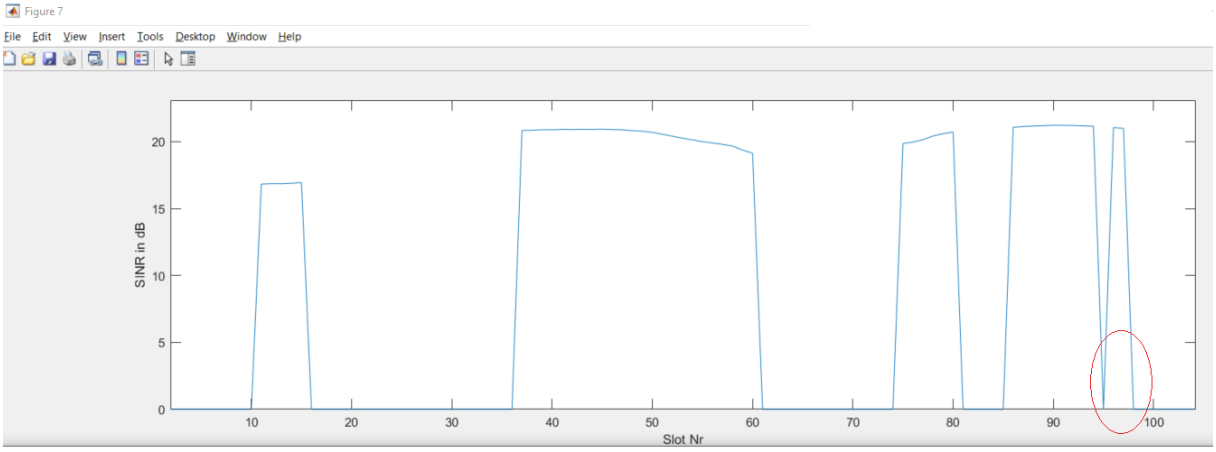


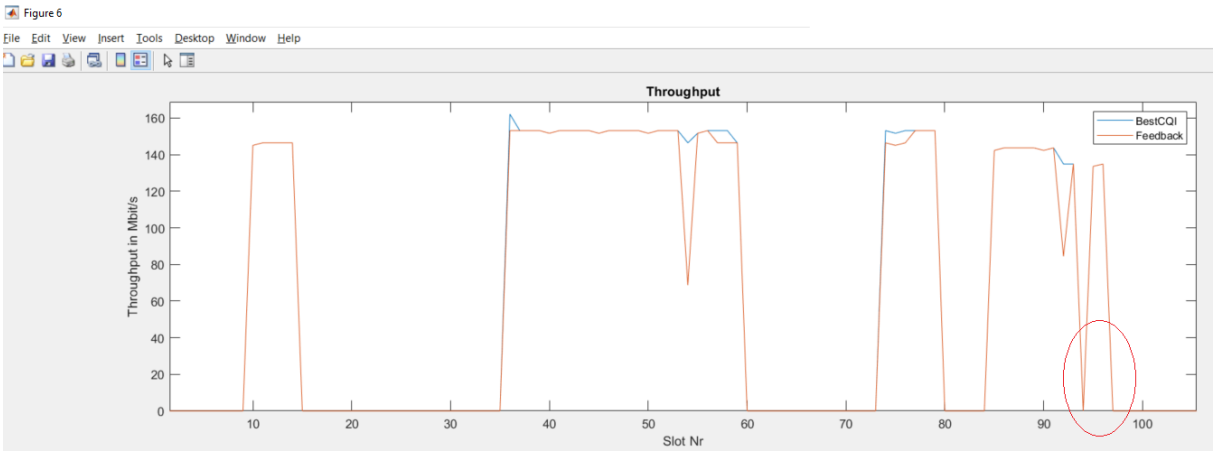
5. LauncherFeedback (**Try5**)

Takie same ustawienia bazowe jak w podejściu “Try4” ale: (a) x10 z powrotem dodałem.



Zoom na pierwsze 100 slotów (1-100):





----------> Admission control dla user1. slot=11 (5 slotów)

----------> Admission control dla user1. slot=37 (23 sloty)

----------> Admission control dla user1. slot=75 (6 slotów)

----------> Admission control dla user1. slot=86 (9 slotów)

UWAGA: widać że w przypadku ostatniej sesji (96-97slot) **nie doszło do aktywowania CAC**, bo ta sesja była zbyt blisko poprzedniej. Ruch się aktywował ale nie sesja widziana jako „CAC"!!!. Na pewno chodzi o zaokrąglenia w warunku odpalającym CAC wewnątrz „ChunkSimulation"

```
nBitsQueue=Inf, sentBits=134880
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=95): 0, User session START=8.815193e+01,DURATION=5.548460e-01
Sesja usera 1, slot=95 NIE-aktywna!
nBitsQueue=Inf, sentBits=0
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=96): 1, User session START=9.252867e+01,DURATION=5.286425e+00
Sesja usera 1, slot=96 AKTYWNA!
nBitsQueue=Inf, sentBits=133608
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=97): 1, User session START=9.252867e+01,DURATION=5.286425e+00
Sesja usera 1, slot=97 AKTYWNA!
nBitsQueue=Inf, sentBits=134880
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=98): 0, User session START=1.002155e+02,DURATION=7.013525e+00
Sesja usera 1, slot=98 NIE-aktywna!
```

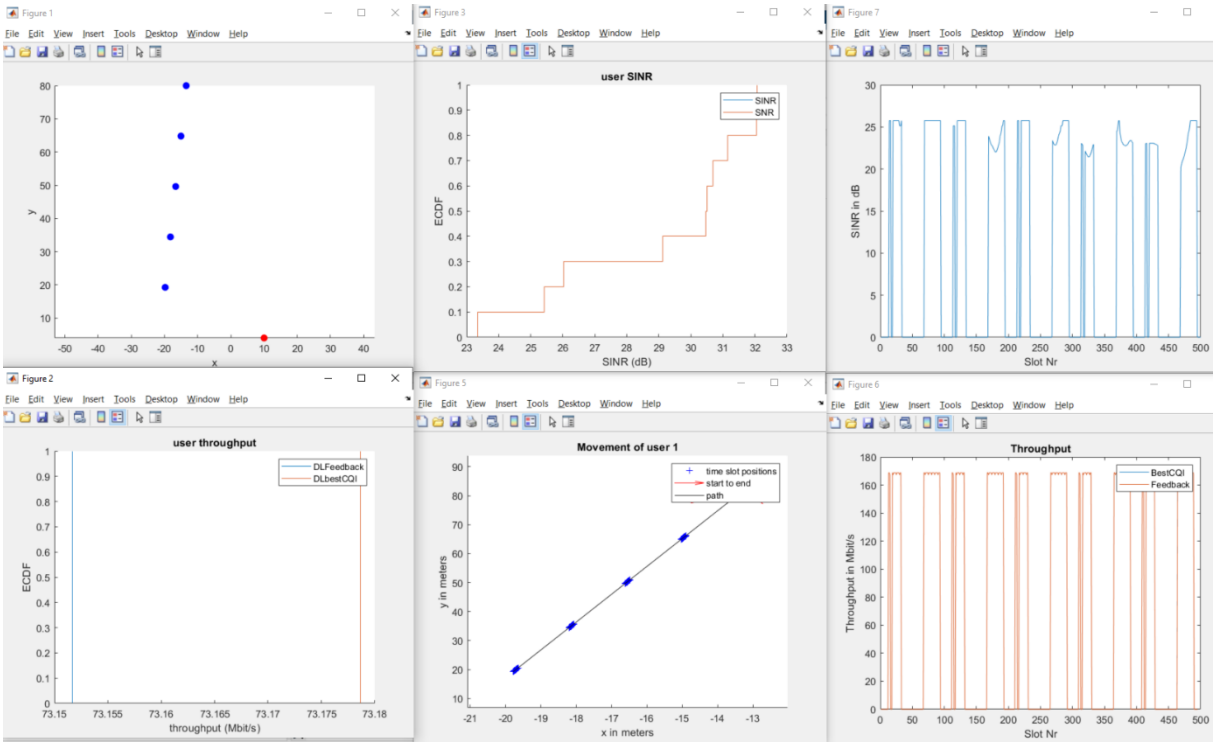
```
% perform admission control
if(ceil(obj.users(ii).start) == iSlot) && (iSlot ~= 1)
    obj.performSessionCAC(obj.users(ii),iSlot); % user(ii) has just now started new session
end
```

UWAGA: zmiana sposobu zaokrąglania z „ceil" na „round" żeby zaokrąglanie było do najbliższej liczby całkowitej.

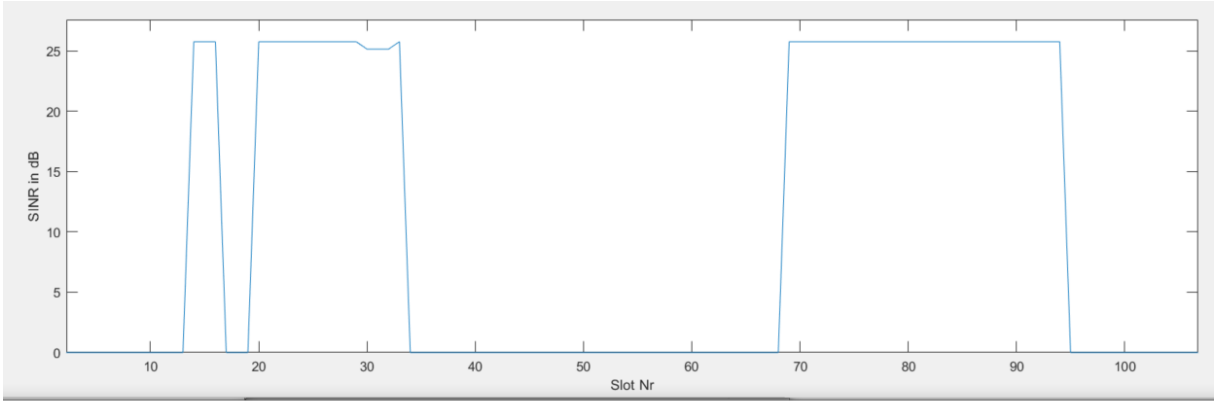
```
% perform admission control
if(round(obj.users(ii).start) == iSlot) && (iSlot ~= 1)
    obj.performSessionCAC(obj.users(ii),iSlot); % user(ii) has just now started new session
end
```

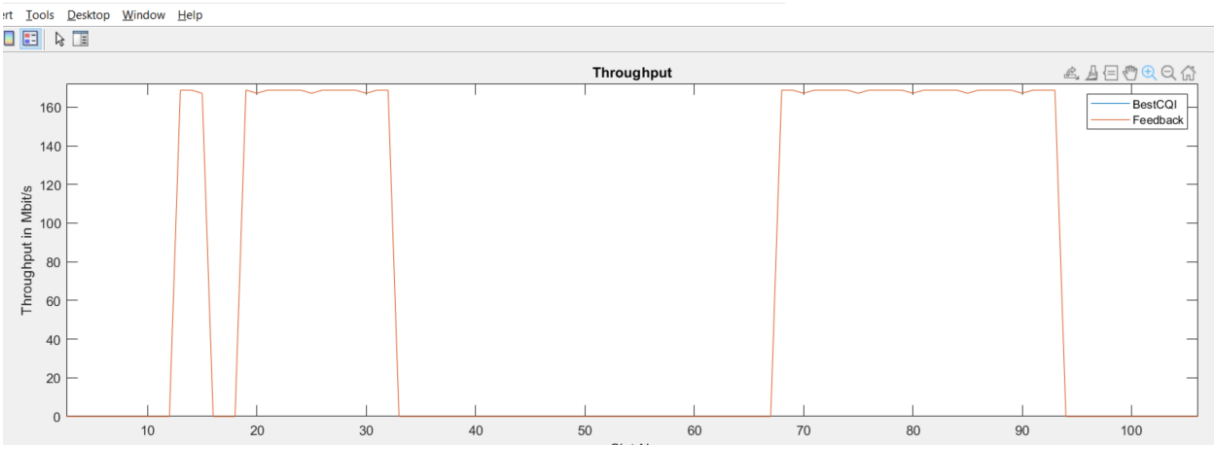
6. LauncherFeedback (**Try6**)

Zmieniona funkcja zaokrąglgająca w wywołaniu „performSessionCAC()”. Reszta BEZ ZMIAN, czyli tak samo jak w Try5.



A teraz zoom dla sprawdzenia „co słychać” w setkach slotów:





Sesja1: ----------> Admission control dla user1. slot=14 (3 sloty)

Sesja2: ----------> Admission control dla user1. slot=19 (13 slotów)

Sesja3: ----------> Admission control dla user1. slot=35 (0 slotów!!!!!)

Sesja4: ----------> Admission control dla user1. slot=68 (18 slotów)

Sesja5: sesja nie zostaje wykryta!

UWAGA: <<**SESJA2**>> nie jest tworzona poprawnie, bo sesja która ma się zacząć 19,3 wg retrieveSession(), zaczyna się dopiero w slocie=20.

```
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=18): 0, User session START=1.931001e+01,DURATION=1.447783e+01
Sesja usera 1, slot=18 NIE-aktywna!
nBitsQueue=Inf, sentBits=0
----------> Admission control dla user1. slot=19
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=19): 0, User session START=1.931001e+01,DURATION=1.447783e+01
Sesja usera 1, slot=19 NIE-aktywna!
nBitsQueue=Inf, sentBits=0
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=20): 1, User session START=1.931001e+01,DURATION=1.447783e+01
Sesja usera 1, slot=20 AKTYWNA!
nBitsQueue=Inf, sentBits=168816
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=21): 1, User session START=1.931001e+01,DURATION=1.447783e+01
Sesja usera 1, slot=21 AKTYWNA!
nBitsQueue=Inf, sentBits=167216
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=22): 1, User session START=1.931001e+01,DURATION=1.447783e+01
Sesja usera 1, slot=22 AKTYWNA!
```

UWAGA: podobnie problem jest przy zakańczaniu sesji, powinna trwać 14 slotów a trwa 13.

```
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=32): 1, User session START=1.931001e+01,DURATION=1.447783e+01
Sesja usera 1, slot=32 AKTYWNA!
nBitsQueue=Inf, sentBits=168816
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=33): 1, User session START=1.931001e+01,DURATION=1.447783e+01
Sesja usera 1, slot=33 AKTYWNA!
nBitsQueue=Inf, sentBits=168816
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=34): 0, User session START=3.518765e+01,DURATION=6.685800e-01
Sesja usera 1, slot=34 NIE-aktywna!
nBitsQueue=Inf, sentBits=0
------------------------------> Admission control dla user1. slot=35
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=35): 0, User session START=3.518765e+01,DURATION=6.685800e-01
```

UWAGA: <<**SESJA3**>> nie zostaje nawet rozpoczęta na skutek zaokrągleń.

```
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=34): 0, User session START=3.518765e+01,DURATION=6.685800e-01
Sesja usera 1, slot=34 NIE-aktywna!
nBitsQueue=Inf, sentBits=0
------------------------------> Admission control dla user1. slot=35
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=35): 0, User session START=3.518765e+01,DURATION=6.685800e-01
Sesja usera 1, slot=35 NIE-aktywna!
nBitsQueue=Inf, sentBits=0
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=36): 0, User session START=6.840646e+01,DURATION=1.763271e+01
Sesja usera 1, slot=36 NIE-aktywna!
nBitsQueue=Inf, sentBits=0
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=37): 0, User session START=6.840646e+01,DURATION=1.763271e+01
```

sesja usera jest za krótka i na skutek zaokrągleń nie zostaje nawet zaczęta.

tu widać że następna sesja jest już wygenerowana

UWAGA: <<SESJA4>> trwa 18 slotów czyli tyle, ile trzeba, ALE w po oczekiwanym zakończeniu sesji, następuje automatyczne rozpoczęcie następnej, która NIE ZOSTAJE ZAUWAŻONA przez „performSessionCAC”! Bo na siebie nachodzą (tj Sesja4 i „niezauważona” Sesja5).

Poniżej widać że Sesja4 się kończy (Slot=86) ale zaraz zaczyna się następna (Slot=87) tylko tyle że nie zostaje zauważona przez „performSessionCAC”.

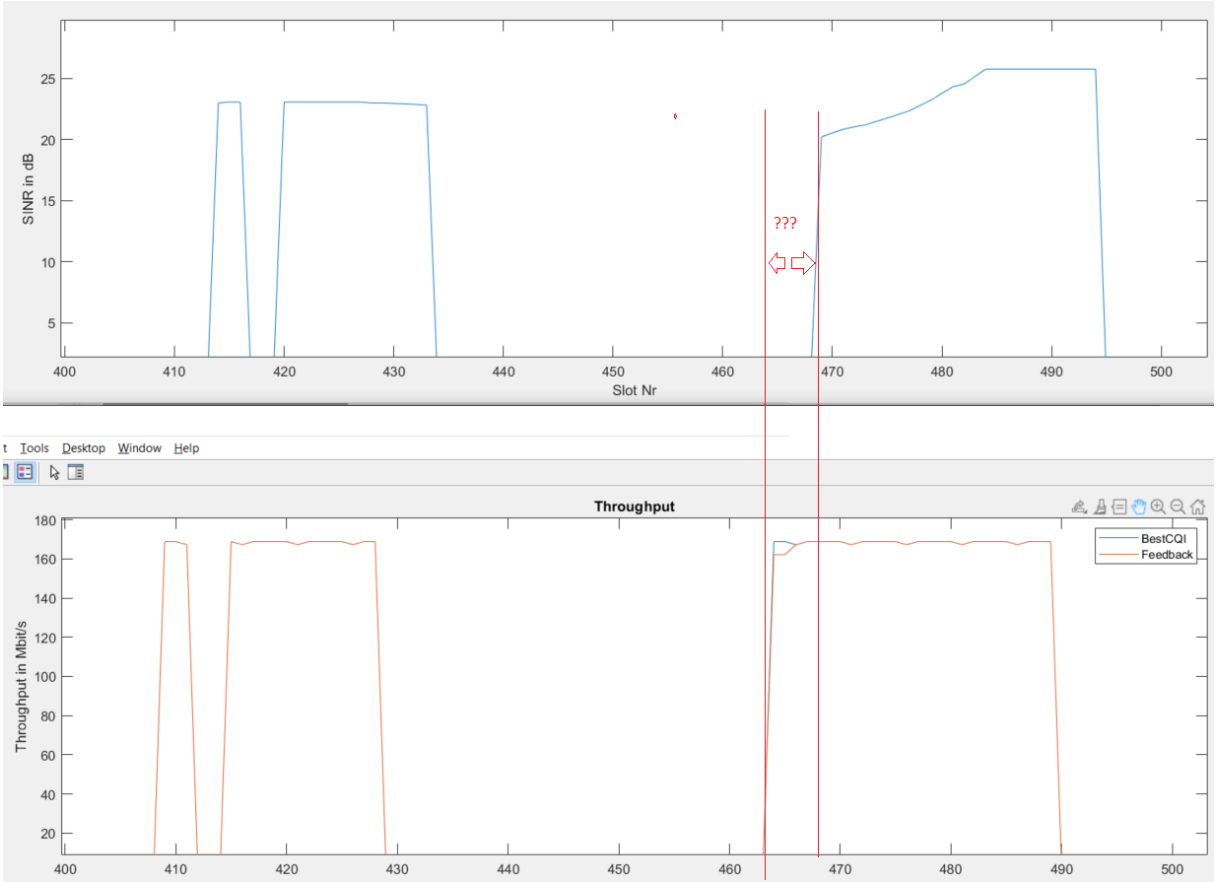
```
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=84): 1, User session START=6.840646e+01,DURATION=1.763271e+01
Sesja usera 1, slot=84 AKTYWNA!
nBitsQueue=Inf, sentBits=168816
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=85): 1, User session START=6.840646e+01,DURATION=1.763271e+01
Sesja usera 1, slot=85 AKTYWNA!
nBitsQueue=Inf, sentBits=168816
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=86): 1, User session START=6.840646e+01,DURATION=1.763271e+01
Sesja usera 1, slot=86 AKTYWNA!
nBitsQueue=Inf, sentBits=167216
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=87): 1, User session START=7.730400e+01,DURATION=1.310456e+01
Sesja usera 1, slot=87 AKTYWNA!
nBitsQueue=Inf, sentBits=168816
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=88): 1, User session START=7.730400e+01,DURATION=1.310456e+01
Sesja usera 1, slot=88 AKTYWNA!
```

Należy jednoznacznie rozwiązać problem zaokrągleń.

Dodano zaokrąglenia wewnątrz User.m/retrieveSessionOfUser() oraz **usunięto zaokrąglenie** przy sprwadzaniu, czy sesja powinna się już zacząć.

```
function a = retrieveSessionOfUser(obj, slot)
    % funkcja generateSessions() wcześniej jest uruchamiana w
    % SimulationSetup.m, na etapie tworzenia userów
    if (obj.id>0) && (~isempty(obj.arrival))
        tmp = find(obj.arrival.class == obj.id); % list of user sessions generate accordint to poisson process

        if (~isempty(tmp))
            a = [obj.arrival.T(tmp(1)), obj.arrival.Th(tmp(1)), obj.arrival.class(tmp(1))]; % always take the first record
            %if(slot<obj.duration)
                obj.setSessionTiming(round(a(1)), round((a(2)-a(1))*10)); % we set the NEW session based on current reading
            %end
            %if obj.isActiveTraffic(slot)
            if ((slot+1)>(obj.start+obj.duration)) % if "true" the current session of user.id will end in 1 slot from now...
                obj.arrival.T(tmp(1))     = -1;    % clear arrival time of a user tmp(1)
                obj.arrival.Th(tmp(1))    = -1;    % clear departure time of a user tmp(1)
                obj.arrival.class(tmp(1))= -1;    % clear arrival class of a user tmp(1)
            end
        else
            % do nothing - this user id has no data in the
```

Dla porównania ostatniej 100-tki slotów w symulacji z pierwszą:

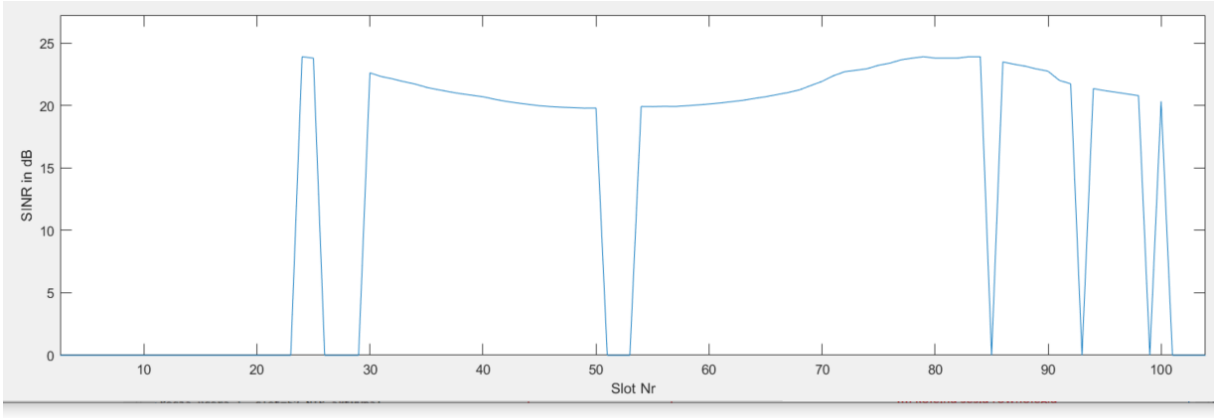


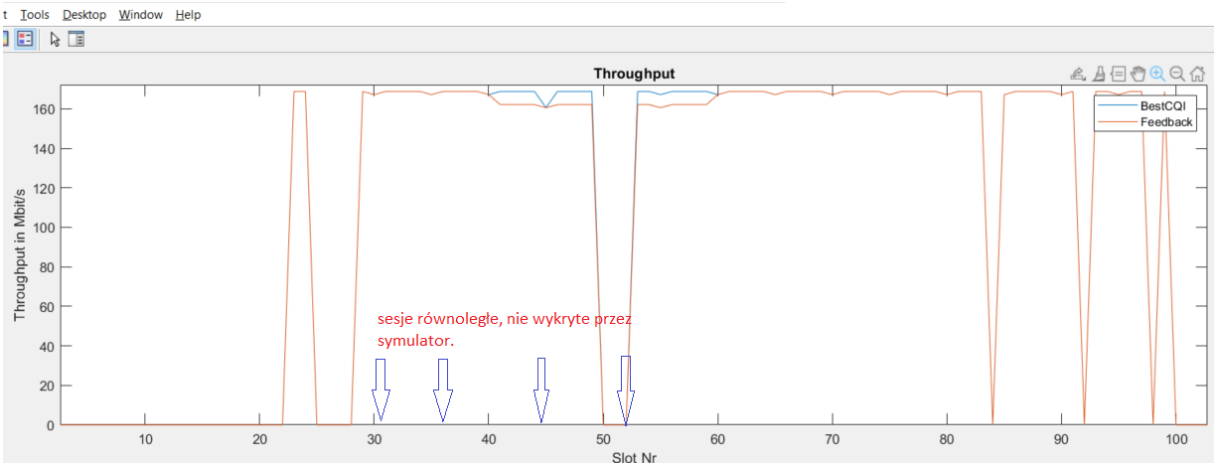
OBSERWACJA: ujawnia się problem przesunięcia wykresu przepływności (Throughput) względem SINR.

7. LauncherFeedback (**Try7**)

Realizujemy powtórzenie podejścia **Try6** po zmianie zaokrągleń w User.m/retrieveSessionOfUser()

I od razu zoom na pierwsze 100-slotów:





Tutaj ujawnił się mocno **problem sesji równoległych, które jak się okazuje nie są obsługiwane oryginalnie w 5G SLS Vienna.**

```
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=48): 1, User session START=30,DURATION=21
Sesja usera 1, slot=48 AKTYWNA!
nBitsQueue=Inf, sentBits=162136
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=49): 1, User session START=30,DURATION=21
Sesja usera 1, slot=49 AKTYWNA!
nBitsQueue=Inf, sentBits=162136
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=50): 1, User session START=30,DURATION=21
Sesja usera 1, slot=50 AKTYWNA!
nBitsQueue=Inf, sentBits=162136
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=51): 0, User session START=31,DURATION=2
Sesja usera 1, slot=51 NIE-aktywna!
nBitsQueue=Inf, sentBits=0
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=52): 0, User session START=36,DURATION=12
Sesja usera 1, slot=52 NIE-aktywna!
nBitsQueue=Inf, sentBits=0
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=53): 0, User session START=44,DURATION=1
Sesja usera 1, slot=53 NIE-aktywna!
nBitsQueue=Inf, sentBits=0
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=54): 1, User session START=52,DURATION=11
Sesja usera 1, slot=54 AKTYWNA!
nBitsQueue=Inf, sentBits=162136
Aktywni userzy (buffer only): 1, Aktywni userzy (sesja AND buffer)(slot=55): 1, User session START=52,DURATION=11
Sesja usera 1, slot=55 AKTYWNA!
nBitsQueue=Inf, sentBits=162136
```

kończy się sesja która miała start=30

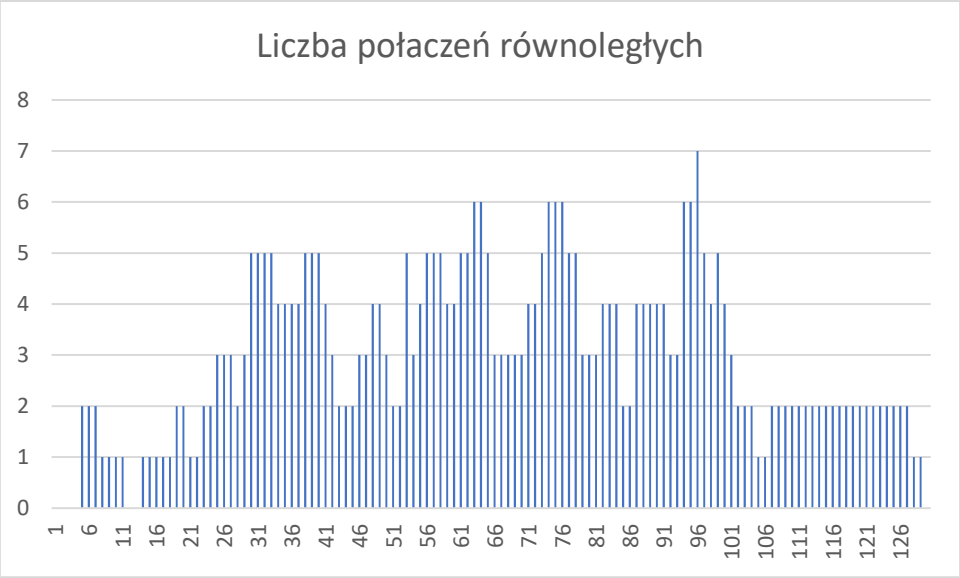
..ale okazuje się że w slot=31 powinna była zacząć się sejsa Start=31. Ale nie było dla niej performSessionCAC

...i kolejna sesja równoległa

... i kolejna sesja równoległa

...i kolejna sesja równoległa

Tego wariantu na obecnym etapie nie obsługujemy… poza tym trzeba pamiętać że czas trwania sesji ma teraz „duration” powiększone x10. Toteż ryzyko równoległych sesji jest większe.



8. LauncherFeedback (**Try8**)

Po kilku dniach zmagań z materią równoległe sesje zostały dodane do symulatora. Natomiast pojawia się następujący problem:

1) dwie sesje zostały wygenerowane dla tego samego slotu i należą do TEJ SAMEJ klasy
2) na ten moment kod obsługuje tworzenie/aktualizację JEDNEJ klasy na raz trzeba to zmienić

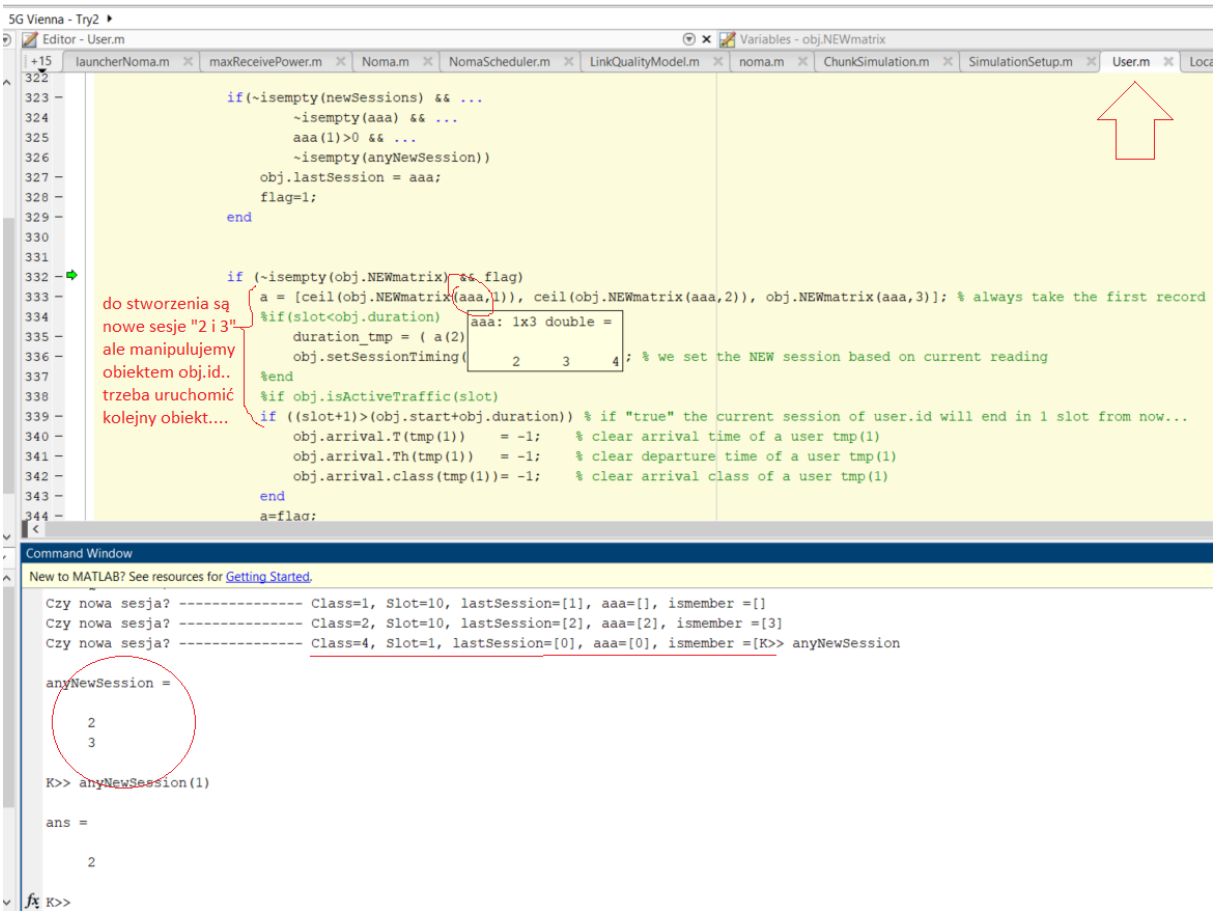
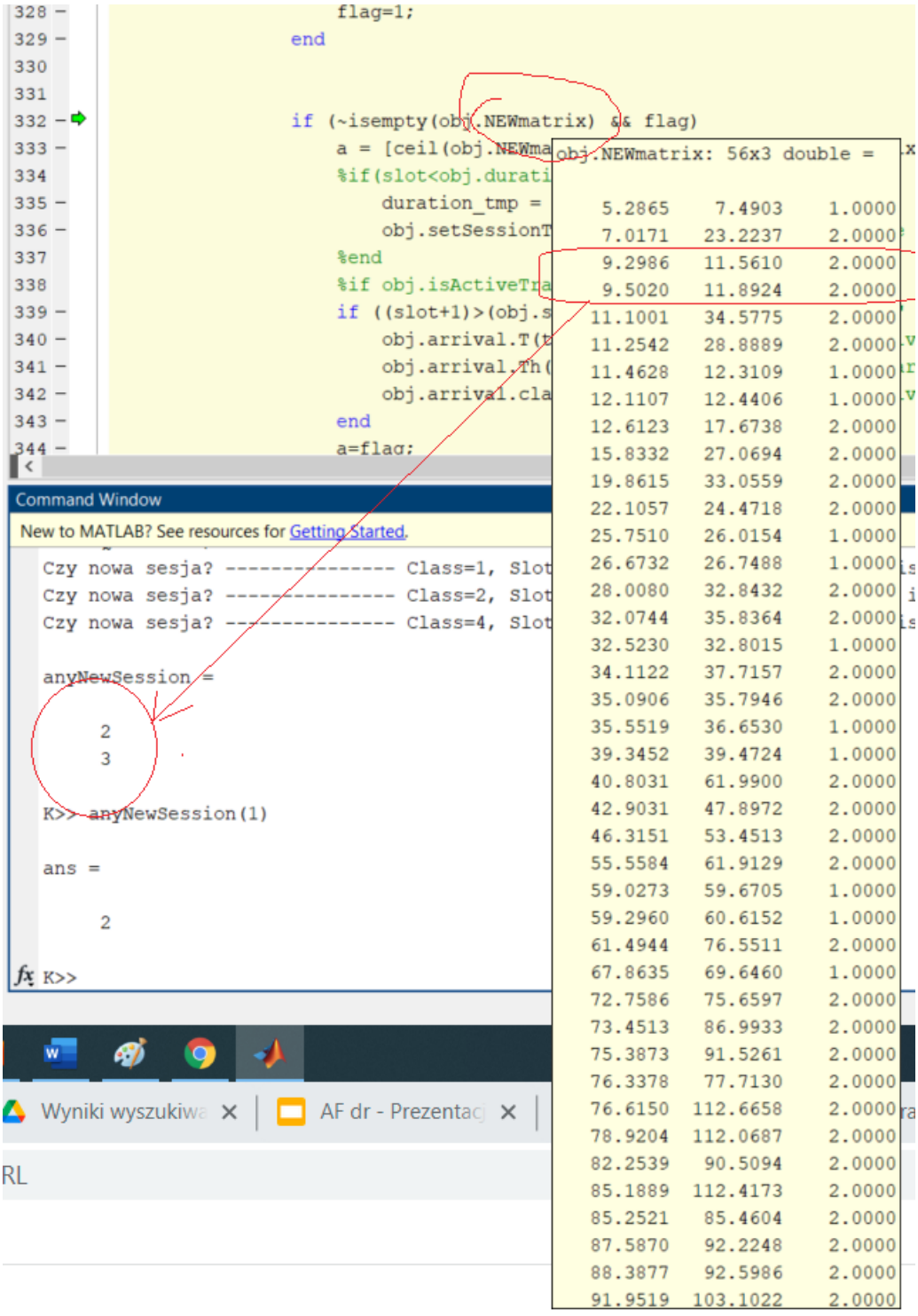
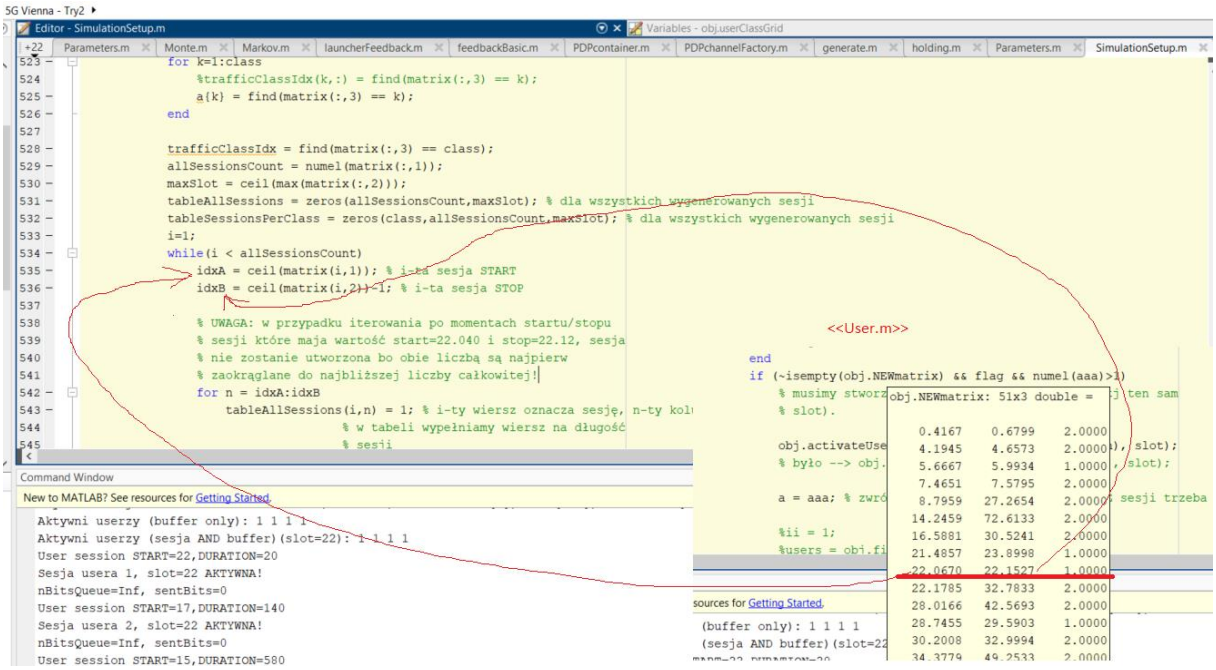


Tabela sesji wygląda tak



UWAGA: w przypadku sesji które mają wylosowany ten sam timeslot jako start/stop (tzn po zaokrągleniu), sesja NIE zostanie utworzona ze względu na zaokrąglanie w górę (ceil), które jest realizowane na tablicy „NEWmatrix” przed zapisaniem czasów start/stop do obiektu user. Ale to jest OK.



ZMIENIŁEM NA: inne wyliczenie „duration” dla sesji.

```
549 -           tableSessionsPerClass = zeros(class,allSessionsCount,maxSlot); % dla wszystkich wygenerowanych sesji
550 -           i=1;
551 -           while(i < allSessionsCount)
552 -               idxA = ceil(matrix(i,1)); % i-ta sesja START
553             % OLD 07.01.2022 idxB = ceil(matrix(i,2))-1; % i-ta sesja STOP
554 -           idxB = ceil(matrix(i,1))+(ceil(matrix(i,2))-ceil(matrix(i,1)))*3; % i-ta sesja STOP
555
556             % UWAGA: w przypadku iterowania po momentach startu/stopu
557             % sesji które maja wartość start=22.040 i stop=22.12, sesja
```

OBSERWACJA: Inny problem to to, że jeśli pojawia się za duża liczba sesji przychodzących podczas sprawdzania usera np. ii=2 tj. user(ii), wtedy czasem dochodzi do czegoś takiego:

```
nBitsQueue=Inf, sentBits=0
Czy nowa sesja? ---------- Class=1, Slot=23, lastSession=[ 8 ], aaa=[8 ], ismember =[1 ]
Czy nowa sesja? ---------- Class=2, Slot=23, lastSession=[ 5 6 7 ], aaa=[5 6 7 10 ], ismember =[1 1 1 0 ]
Output argument "users" (and maybe others) not assigned during call to "simulation/ChunkSimulation/findIdleUEs".

Error in simulation.ChunkSimulation/runSimulation (line 303)
                usrTemp = obj.findIdleUEs(aa-1);
```

Czyli wygląda, że wewnątrz funkcji „findIfleUEs” brakowało obiektów user’a.

A teraz analiza przychodzących połączeń dla luncherNOMA:

```
Czy nowa sesja? ---------- Class=1, Slot=9, lastSession=[ ], aaa=[], ismember =[]
Czy nowa sesja? ---------- Class=2, Slot=9, lastSession=[ 1 ], aaa=[1 2 ], ismember =[1 0 ]
--------------------------------> Admission control dla user2. slot=9
Czy nowa sesja? ---------- Class=3, Slot=9, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Czy nowa sesja? ---------- Class=4, Slot=9, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Aktywni userzy (buffer only): 1 1 1 1
Aktywni userzy (sesja AND buffer)(slot=9): 1 1 1 1
User session START=9,DURATION=70
```

```
Czy nowa sesja? --------------- Class=1, Slot=10, lastSession=[ ], aaa=[], ismember =[]
Czy nowa sesja? --------------- Class=2, Slot=10, lastSession=[ 1 2 ], aaa=[1 2 ], ismember =[1 1 ]
Czy nowa sesja? --------------- Class=3, Slot=10, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Czy nowa sesja? --------------- Class=4, Slot=10, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Aktywni userzy (buffer only): 1 1 1 1
Aktywni userzy (sesja AND buffer)(slot=10): 1 1 1 1
User session START=9,DURATION=70
```

```
nBitsQueue=Inf, sentBits=48
Czy nowa sesja? --------------- Class=1, Slot=11, lastSession=[ ], aaa=[], ismember =[]
Czy nowa sesja? --------------- Class=2, Slot=11, lastSession=[ 1 2 ], aaa=[1 2 ], ismember =[1 1 ]
Czy nowa sesja? --------------- Class=3, Slot=11, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Czy nowa sesja? --------------- Class=4, Slot=11, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Aktywni userzy (buffer only): 1 1 1 1
Aktywni userzy (sesja AND buffer)(slot=11): 1 1 1 1
User session START=9,DURATION=70
```

```
Czy nowa sesja? --------------- Class=1, Slot=12, lastSession=[ ], aaa=[], ismember =[]
Czy nowa sesja? --------------- Class=2, Slot=12, lastSession=[ 1 2 ], aaa=[1 2 ], ismember =[1 1 ]
Czy nowa sesja? --------------- Class=3, Slot=12, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Czy nowa sesja? --------------- Class=4, Slot=12, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Aktywni userzy (buffer only): 1 1 1 1
Aktywni userzy (sesja AND buffer)(slot=12): 1 1 1 1
```

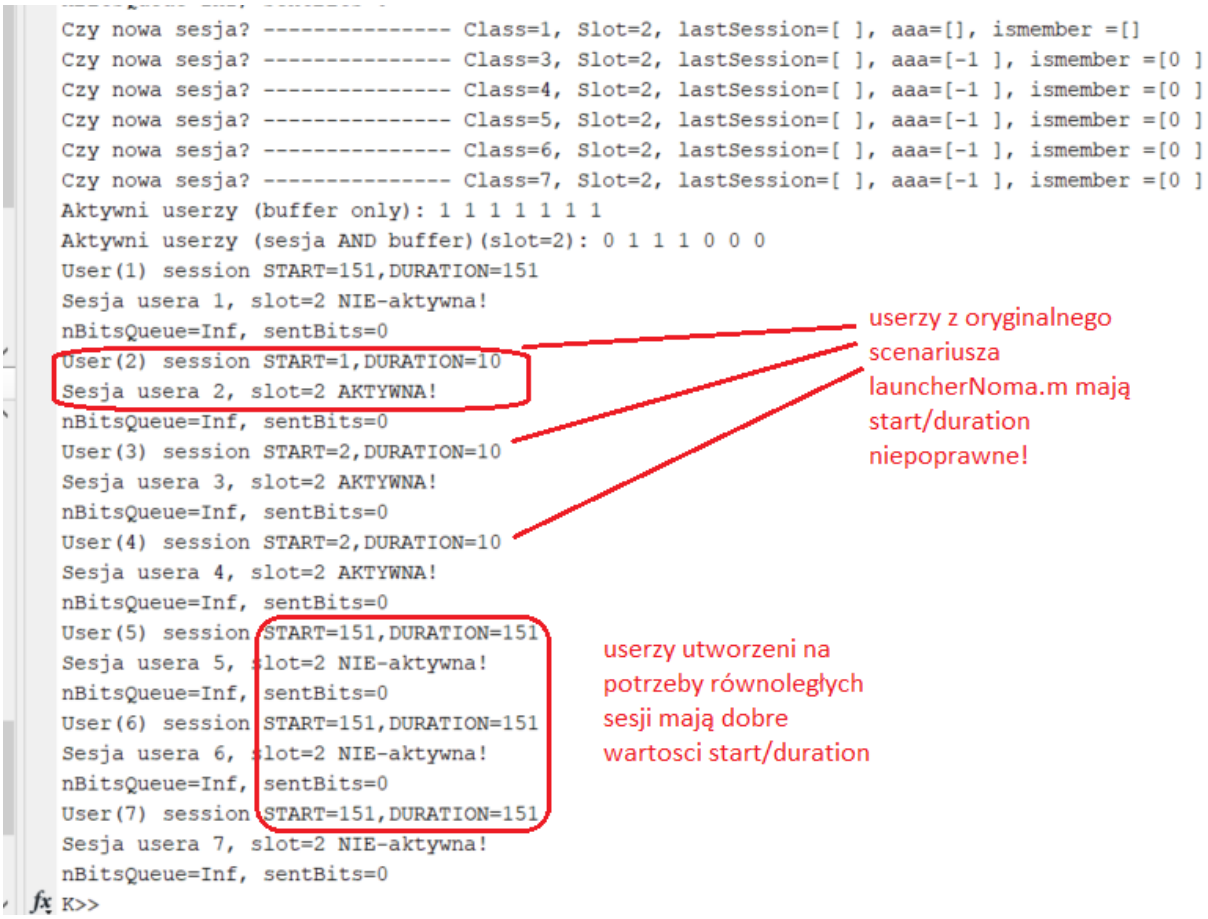
```
nBitsQueue=Inf, sentBits=48
Czy nowa sesja? --------------- Class=1, Slot=13, lastSession=[ ], aaa=[], ismember =[]
Czy nowa sesja? --------------- Class=2, Slot=13, lastSession=[ 1 2 ], aaa=[2 ], ismember =[1 ]
Czy nowa sesja? --------------- Class=3, Slot=13, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Czy nowa sesja? --------------- Class=4, Slot=13, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Aktywni userzy (buffer only): 1 1 1 1
Aktywni userzy (sesja AND buffer)(slot=13): 1 1 0 0
```

```
Czy nowa sesja? --------------- Class=1, Slot=14, lastSession=[ ], aaa=[], ismember =[]
Czy nowa sesja? --------------- Class=2, Slot=14, lastSession=[ 1 2 ], aaa=[2 ], ismember =[1 ]
Czy nowa sesja? --------------- Class=3, Slot=14, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Czy nowa sesja? --------------- Class=4, Slot=14, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Aktywni userzy (buffer only): 1 1 1 1
Aktywni userzy (sesja AND buffer)(slot=14): 1 1 0 0
User session START=9,DURATION=70
```

A tu jeszcze kwestia którą udało się zidentyfikować w konsoli systemowej Matlab – mianowicie, że wartości start/duration dla obiektu user są różne w różnych miejscach w kodzie ChunkSimulation.

Te uruchamiane dla CAC są widoczne poniżej, i to już wygląda dość dobrze. Natomiast te wyświetlane w środku pliku ChunkSimulation.m, gdzie sprawdzamy tylko poszczególne wartości start/duration są INNE – dlaczego? To wymaga dalszych analiz.

```
Czy nowa sesja? --------------- Class=1, Slot=2, lastSession=[ ], aaa=[], ismember =[]
Czy nowa sesja? --------------- Class=3, Slot=2, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Czy nowa sesja? --------------- Class=4, Slot=2, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Czy nowa sesja? --------------- Class=5, Slot=2, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Czy nowa sesja? --------------- Class=6, Slot=2, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Czy nowa sesja? --------------- Class=7, Slot=2, lastSession=[ ], aaa=[-1 ], ismember =[0 ]
Aktywni userzy (buffer only): 1 1 1 1 1 1 1
Aktywni userzy (sesja AND buffer)(slot=2): 0 1 1 1 0 0 0
```


Czy nowa sesja? ---------- Class=1, Slot=2, lastSession=[ ], aaa=[], ismember =[]
Czy nowa sesja? ---------- Class=3, Slot=2, lastSession=[ ], aaa=[-1], ismember =[0]
Czy nowa sesja? ---------- Class=4, Slot=2, lastSession=[ ], aaa=[-1], ismember =[0]
Czy nowa sesja? ---------- Class=5, Slot=2, lastSession=[ ], aaa=[-1], ismember =[0]
Czy nowa sesja? ---------- Class=6, Slot=2, lastSession=[ ], aaa=[-1], ismember =[0]
Czy nowa sesja? ---------- Class=7, Slot=2, lastSession=[ ], aaa=[-1], ismember =[0]
Aktywni userzy (buffer only): 1 1 1 1 1 1 1
Aktywni userzy (sesja AND buffer)(slot=2): 0 1 1 1 0 0 0
User(1) session START=151,DURATION=151
Sesja usera 1, slot=2 NIE-aktywna!
nBitsQueue=Inf, sentBits=0
User(2) session START=1,DURATION=10
Sesja usera 2, slot=2 AKTYWNA!
nBitsQueue=Inf, sentBits=0
User(3) session START=2,DURATION=10
Sesja usera 3, slot=2 AKTYWNA!
nBitsQueue=Inf, sentBits=0
User(4) session START=2,DURATION=10
Sesja usera 4, slot=2 AKTYWNA!
nBitsQueue=Inf, sentBits=0
User(5) session START=151,DURATION=151
Sesja usera 5, slot=2 NIE-aktywna!
nBitsQueue=Inf, sentBits=0
User(6) session START=151,DURATION=151
Sesja usera 6, slot=2 NIE-aktywna!
nBitsQueue=Inf, sentBits=0
User(7) session START=151,DURATION=151
Sesja usera 7, slot=2 NIE-aktywna!
nBitsQueue=Inf, sentBits=0
K>>
userzy z oryginalnego scenariusza launcherNoma.m mają start/duration niepoprawne!
userzy utworzeni na potrzeby równoległych sesji mają dobre wartosci start/duration

9. LauncherFeedback (**Try8**)

Po wielu zmianach:

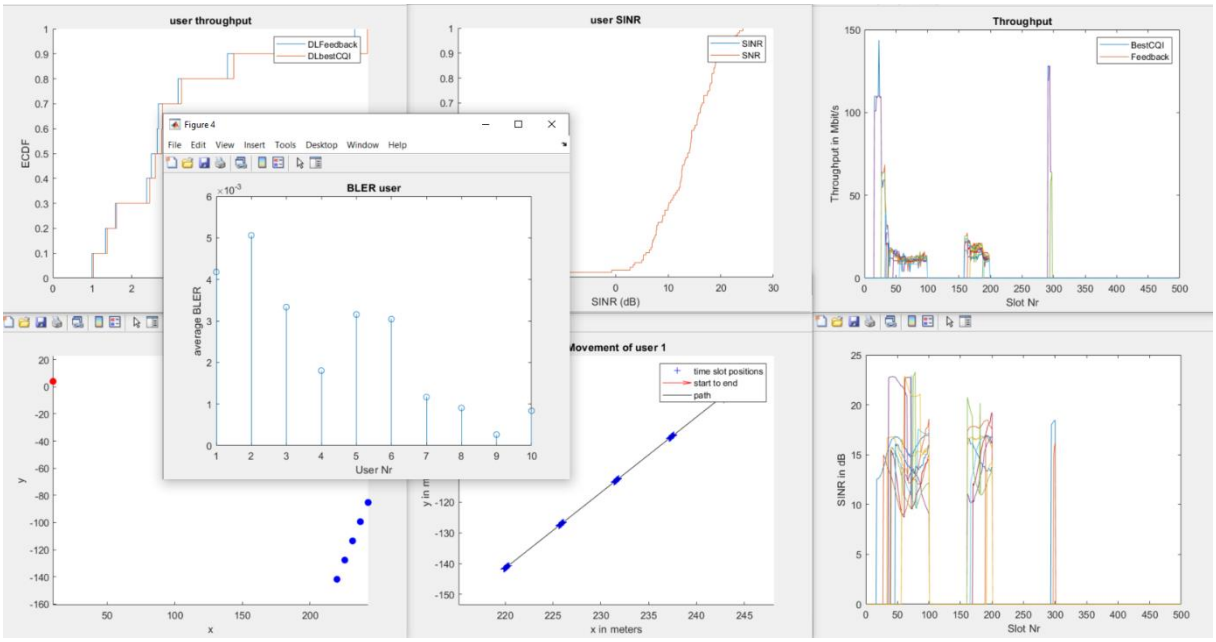
- tworzenie sesji praktycznie działa
- persistent NEWmatrix, działa
- naprawione różne błędy, etc

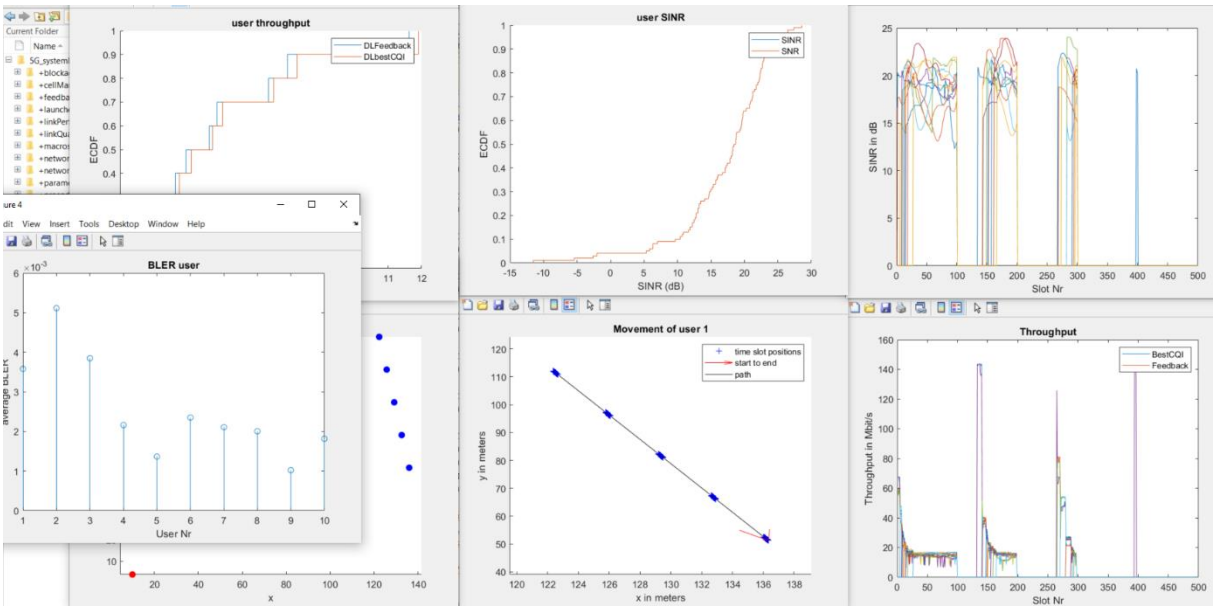
chunkSimulation	`poissonGenerator=1;`

feedbackBasic	`params.time.numberOfChunks = 5;`
	TrafficModel = **FullBuffer**
	
	`reszta bez zmian (pod kątem Try1)`

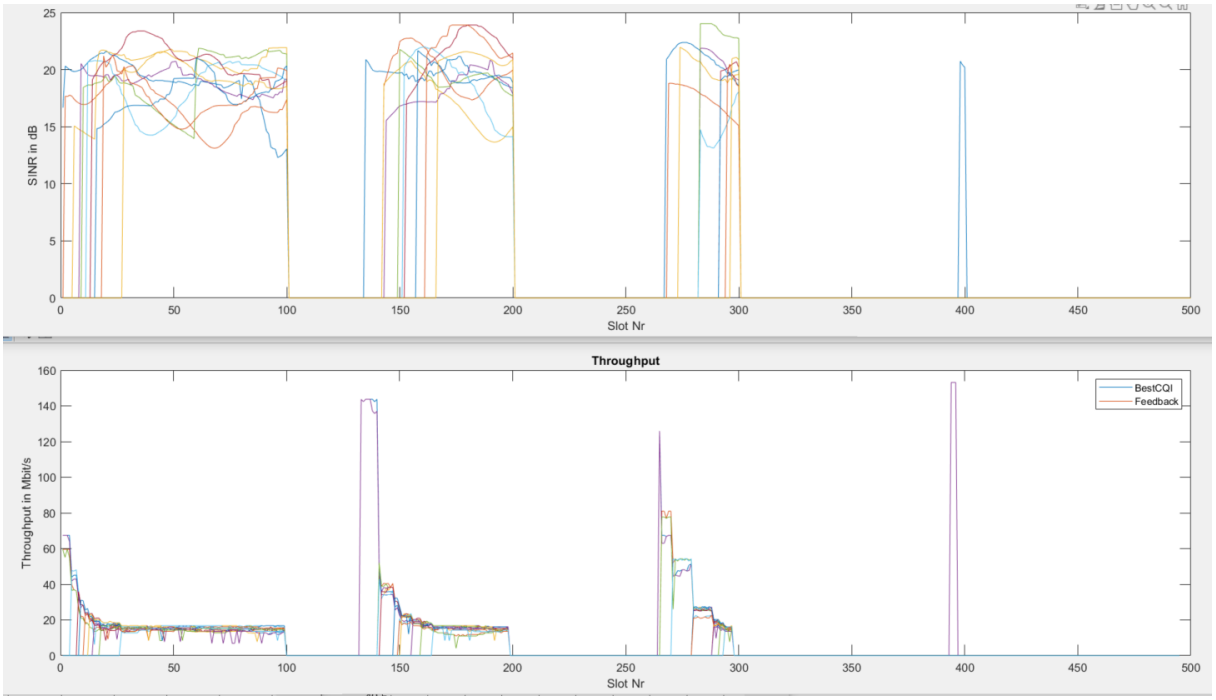
launcherFeedback	No changes

Poniższe serie pomiarowe pokazują, że w chwili obecnej generowanie nowych sesji jest już obsługiwane poprawnie.

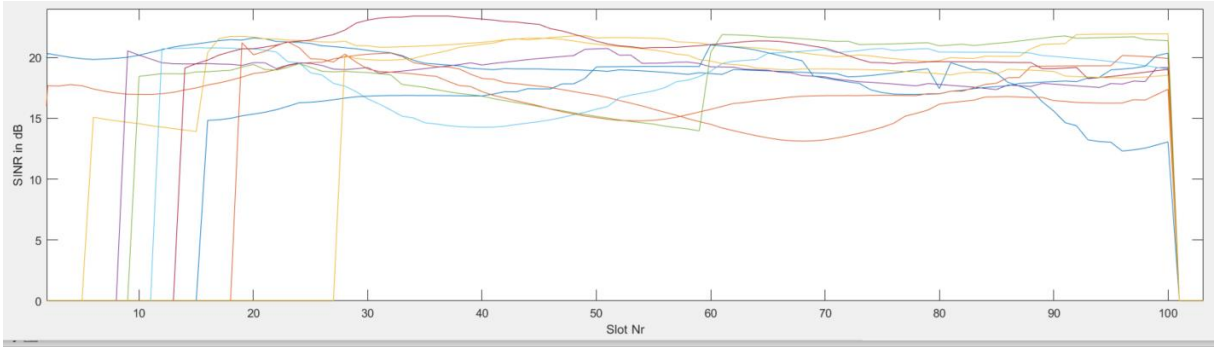




Zoom in



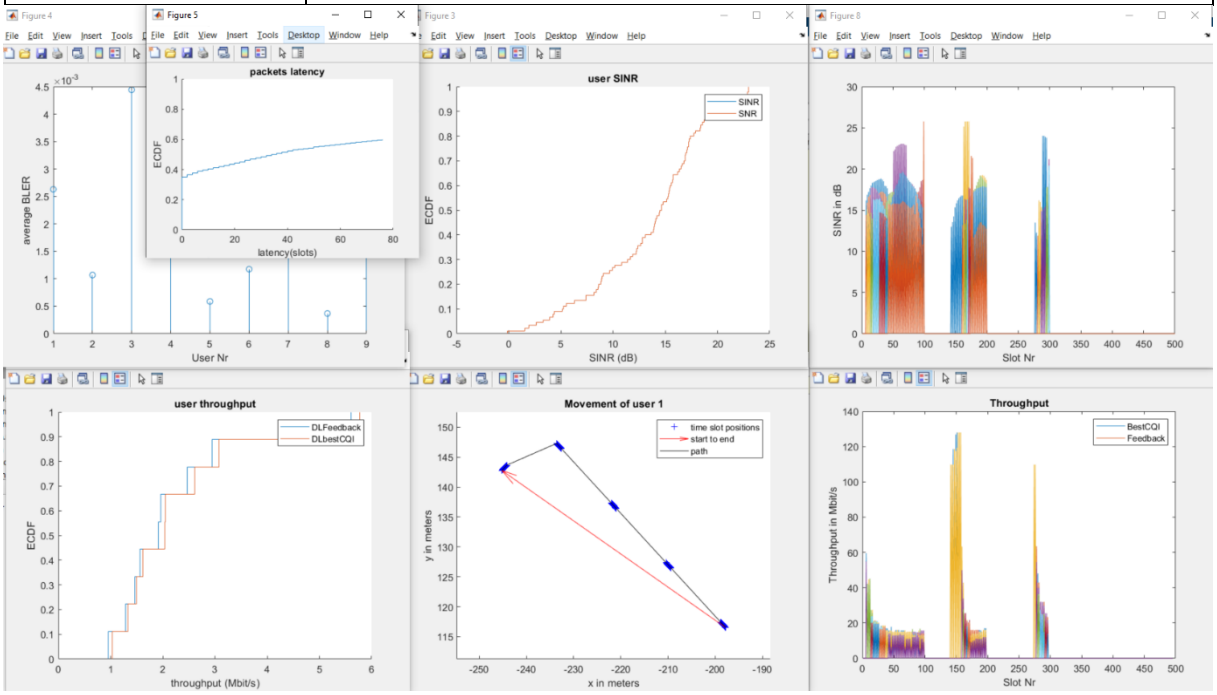
Zoom in (pierwsze 100 slotów)



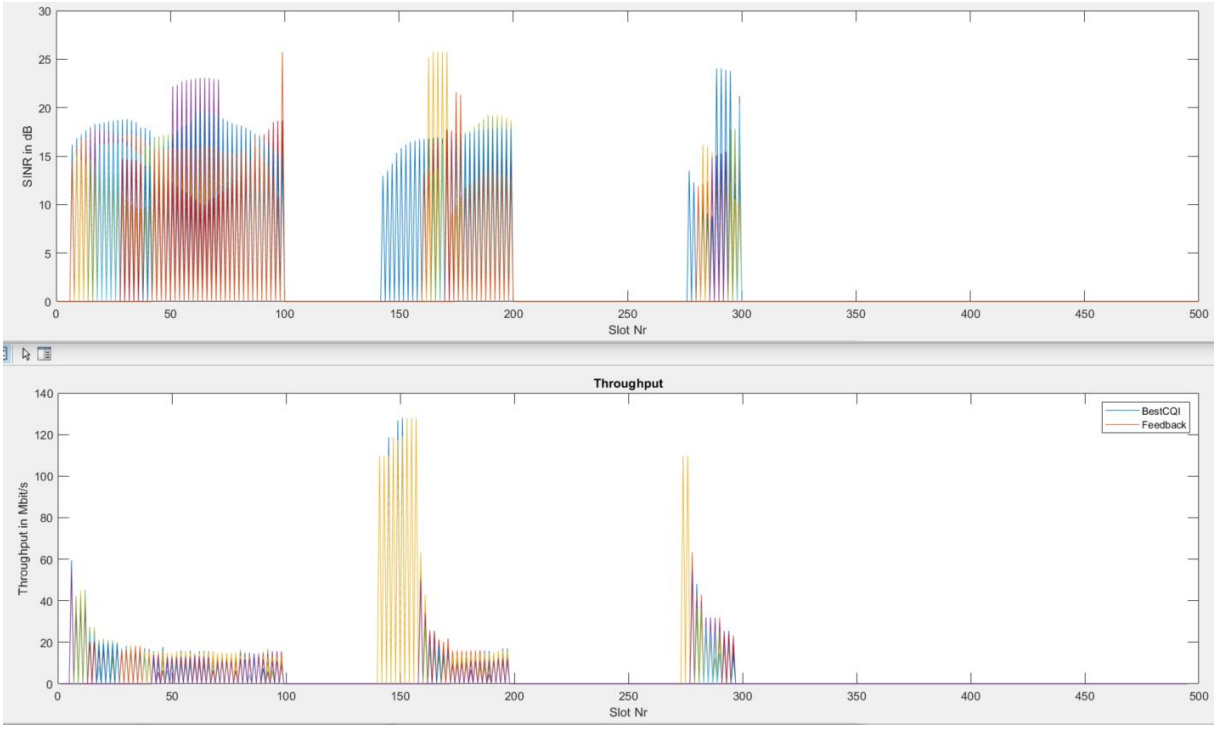
10. LauncherFeedback (**Try9**)

Tym razem chcemy sprawdzić wpływ ruchu o stałej przepływności w ramach generowania sesji, a nie jak poprzednio w modelu “FullBuffer”.

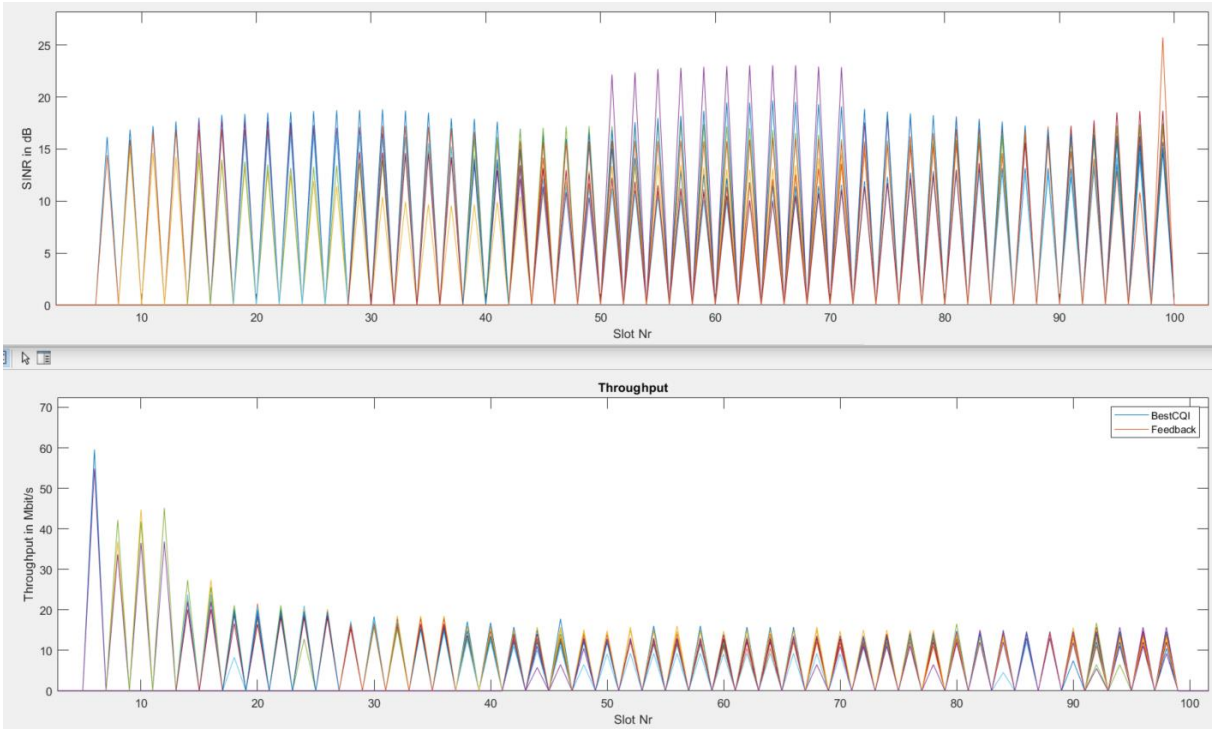
chunkSimulation	`poissonGenerator=1;`
feedbackBasic	`params.time.numberOfChunks = 5;` TrafficModel = **ConstantBitrate** reszta bez zmian (pod kątem Try1)
launcherFeedback	No changes



Zoom in:



Zoom in (pierwsze 100 slotów)

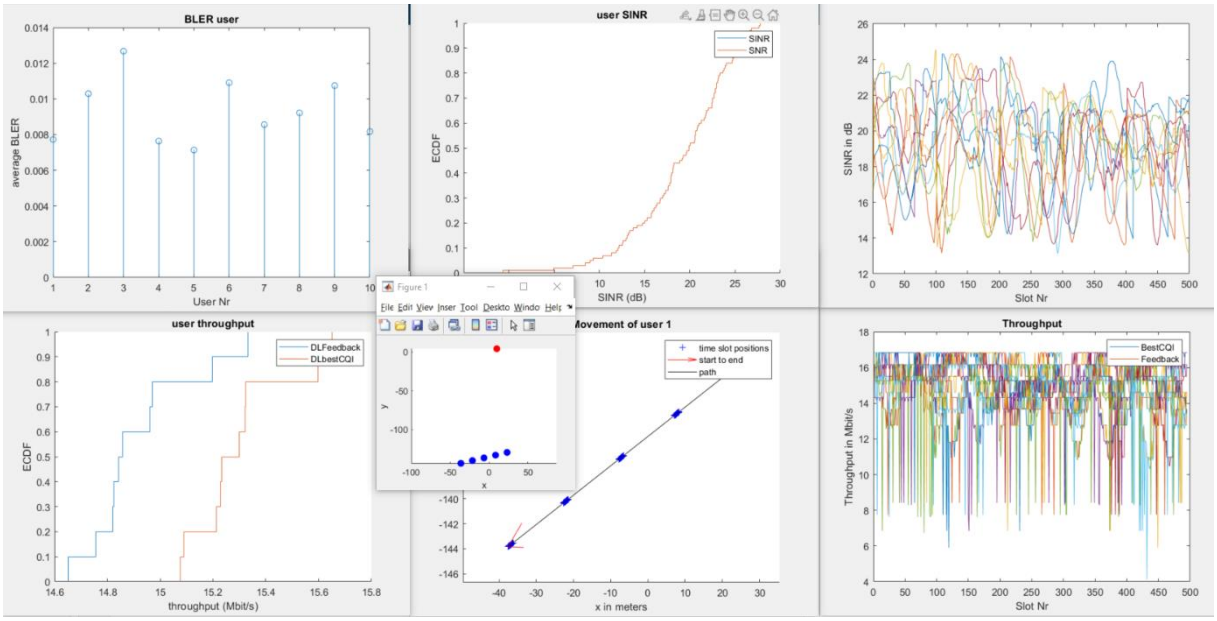


11. LauncherFeedback (**Try10**)

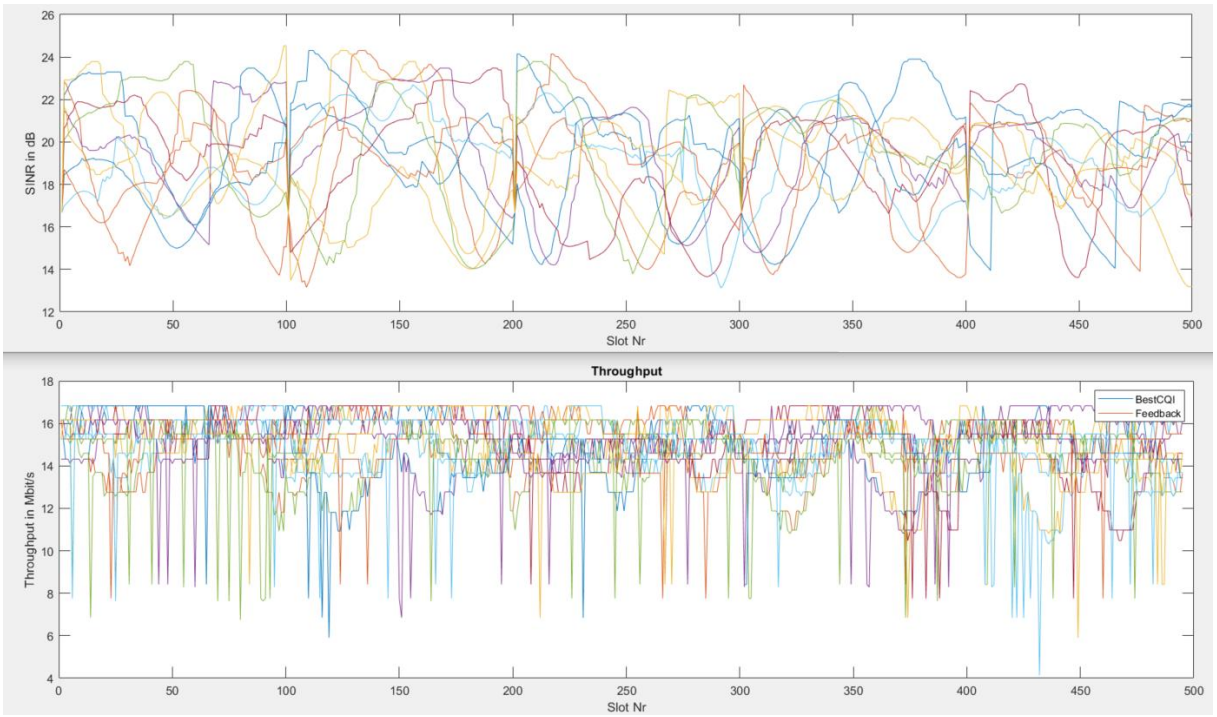
I ponownie ruch generowany w modelu “Fuill buffer”, ale już bez sesji generowanych losowo.

chunkSimulation	`poissonGenerator=0;`
feedbackBasic	`params.time.numberOfChunks  = 5;`

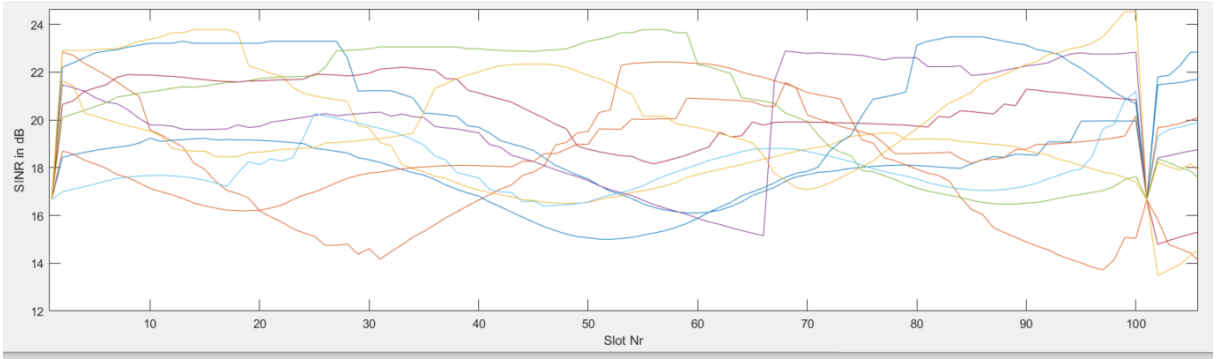
	TrafficModel = **FullBuffer** reszta bez zmian (pod kątem Try1)
launcherFeedback	No changes

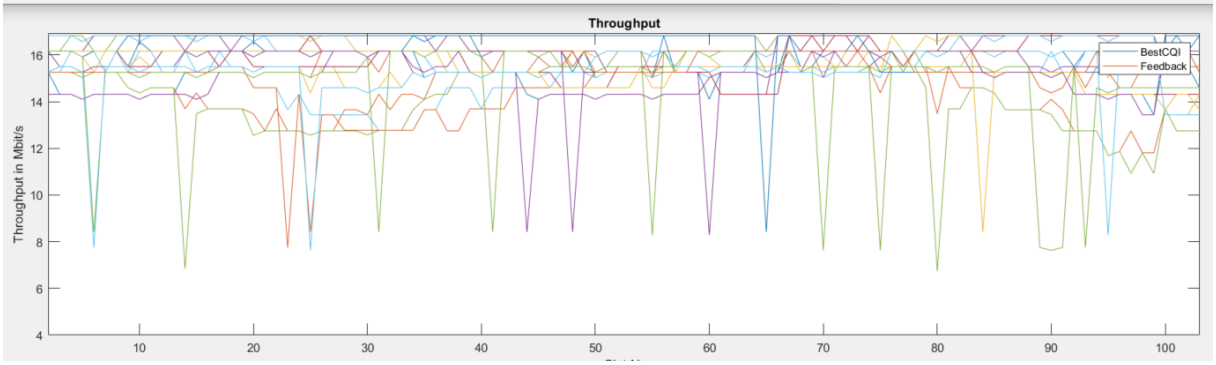


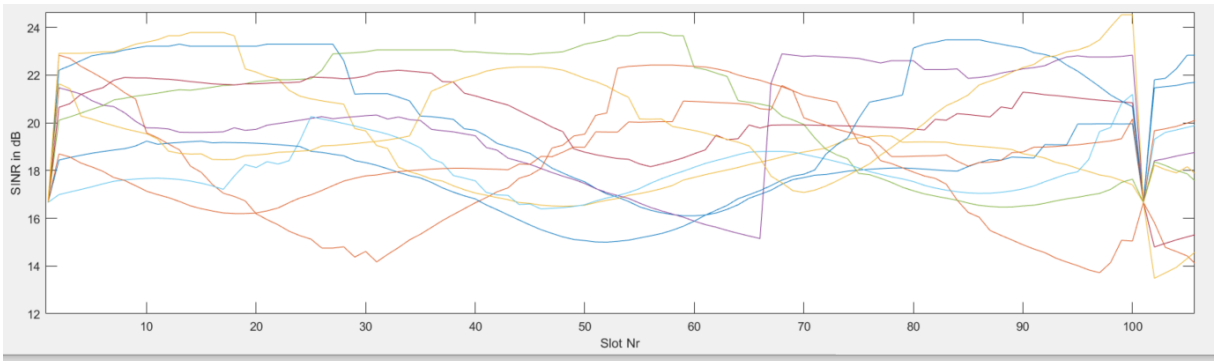
Zoom in

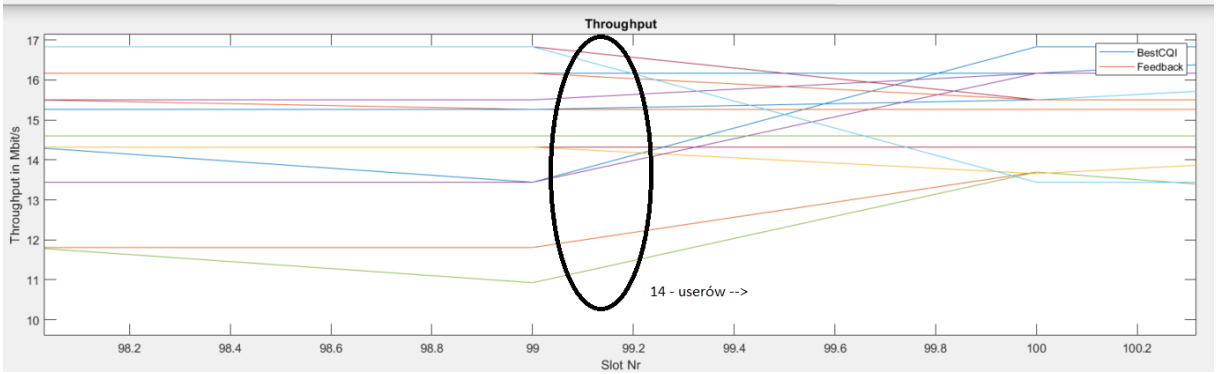


Zoom in (pierwsze 100 slots)





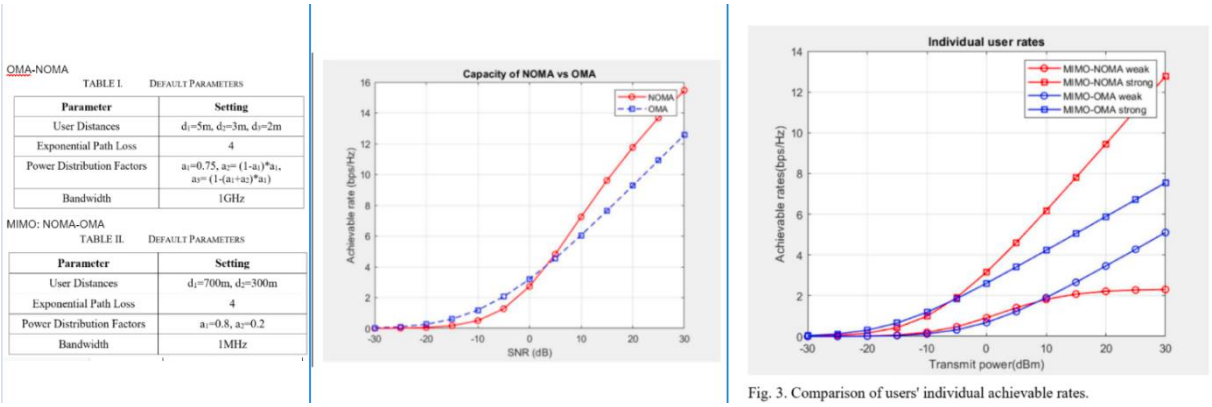




Dalsze działania już po:
- dodaniu mechanizmu wspierającego sesje
- naprawieniu problemu clearFinishedSession (24.01)

Testy pod kątem porównania z „**Paper1**”

Poniżej główne wyniki z papera. Pierwszy od lewej wykres jest dla SISO (choć trzeba się domyślać bo wprost autorzy nie piszą, ale jest to w sekcji „OMA, NOMA” a dopiero kolejna sekcja nazywa się „OMA,NOMA – MIMO”.



OMA-NOMA

TABLE I DEFAULT PARAMETERS

Parameter	Setting
User Distances	d_1=5m, d_2=3m, d_3=2m
Exponential Path Loss	4
Power Distribution Factors	a_1=0.75, a_2= (1-a_1)*a_1, a_3= (1-(a_1+a_2)*a_1)
Bandwidth	1GHz

MIMO: NOMA-OMA

TABLE II DEFAULT PARAMETERS

Parameter	Setting
User Distances	d_1=700m, d_2=300m
Exponential Path Loss	4
Power Distribution Factors	a_1=0.8, a_2=0.2
Bandwidth	1MHz

Fig. 3. Comparison of users' individual achievable rates.

Parametr	**Paper1**	**Ustawienia w ramach projektu RATfor5G+**	**Comment**
User distances	D1=5m; D2=3m; D3=2m	PL1=80dB PL2=120dB	Domyślna symulacja jest względem pathloss (ale można przejść na distance)
path loss exponent	4	4	-
Power distribution factor	0,75 -> 0,20->0,5	0,8 -> 0.2	Nie wiem czemu pokazują 3 userów..
BW	1Ghz (?????)	1.4MHz	Wydaje się ze ich szerokość pasma jest błędna..

Testy pod kątem porównania z „**Paper12**”

TABLE I. SIMULATION PARAMETERS

S.No.	*Parameters*	*Values*
1	Base Station Power(dBm)	43 [14]
2	Number of Bits	1000
3	SNR(in dB)	0 to 25
4	No. of users	2
5	System bandwidth (Hz)	1
6	No. of Transmitter Antennas	1
7	No. of Receiver Antennas	1

1. Test z ustawieniami:

Scenariusz	Ustawienia	Komentarz
launcherNOMA	SISO 5G (na szybko LTE wyniki wychodzą podobnie) BW 1,4MHz 2xOMA + 4xNOMA UE1,2 = pathloss ca 80-85 UE3,4 = pathloss ca 115-120 Channel=**PedA** Antena = Omni	
	`antennaOmni.height = 0;`	Default setting

UWAGA: na praktycznie wszystkich wykresach poniżej **wartości na osi Y (opisane jako „Throughput") oznaczają pojemność w „bit/s/Hz",** ponieważ podzielono wyniki przepływności podane w Mbit/s przez 1.4Mhz (tyle wynosi pasmo w symulacji).

Maximum throughput of LTE DL SISO:

Bandwidth: 1.4 MHz
Modulation: 64QAM
MIMO: without MIMO (SISO)

Maximum throughput: 4.392 Mbps

```
simple4G5G.m | launcherSimulationTime.m | launcherSimple4G5G.m | UserNoma.m | CqiParameters.m | LteCqiParametersTS36213NonBLCEUE1withoutCalibration.m | LteCqiParametersTS36213NonBLCEUE1
ods
function obj = LteCqiParametersTS36213NonBLCEUE1(transmissionParameters)
    % class constructor - sets CQI table according to TX 36.213 V13.2.0 (2016-06) (i.e., Table 7.2.3-1)
    %
    % input:
    %   transmissionParameters: [1x1]handleObject parameters.transmissionParameters.TransmissionParameters

    % set CQI table according to standard
    cqi                     = [0      1      2      3      4      5      6      7      8      9      10     11     12     13     14     15];
    obj.modulationType      = [1      2      2      2      2      2      2      3      3      3      4      4      4      4      4      4];
    obj.modulationOrder     = [0      2      2      2      2      2      2      4      4      4      6      6      6      6      6      6];
    obj.modulationName      = {'None' 'QPSK' 'QPSK' 'QPSK' 'QPSK' 'QPSK' 'QPSK' '16QAM' '16QAM' '16QAM' '64QAM' '64QAM' '64QAM' '64QAM' '64QAM' '64QAM'};
    obj.codingRateX1024     = [0      78     120    193    308    449    602    378    490    616    466    567    666    772    873    948];
    obj.efficiency          = [0      0.1523 0.2344 0.3770 0.6016 0.8770 1.1758 1.4766 1.9141 2.4063 2.7305 3.3223 3.9023 4.5234 5.1152 5.5547];
    obj.betaMIESMCalibration = [1     3.07   4.41   0.6    1.16   1.06   1.06   0.87   1.01   1.04   1.03   1.11   1.01   1.07   1      1.05];
    obj.nCqi                = 16;

    obj.blerCurveFiles      = cell(obj.nCqi,1);
    obj.blerCurveFiles{1}   = ''; % no bler curve for zero CQI
```

Natomiast pytanie, czy wszystkie ustawienia scenariusza są wystarczająco zbliżone do **Paper1**.

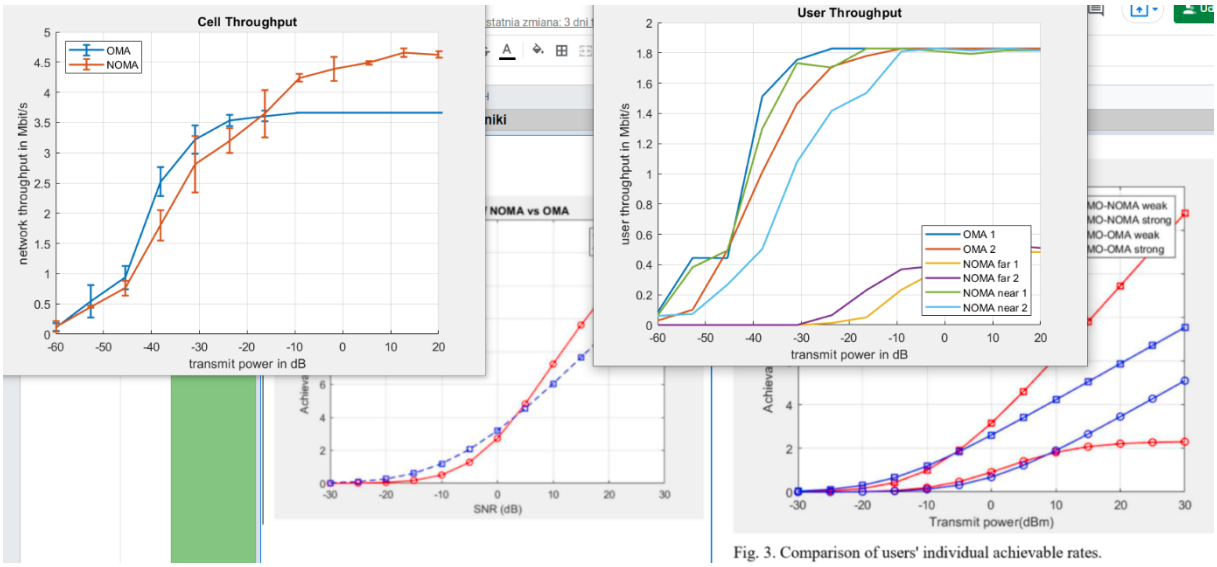
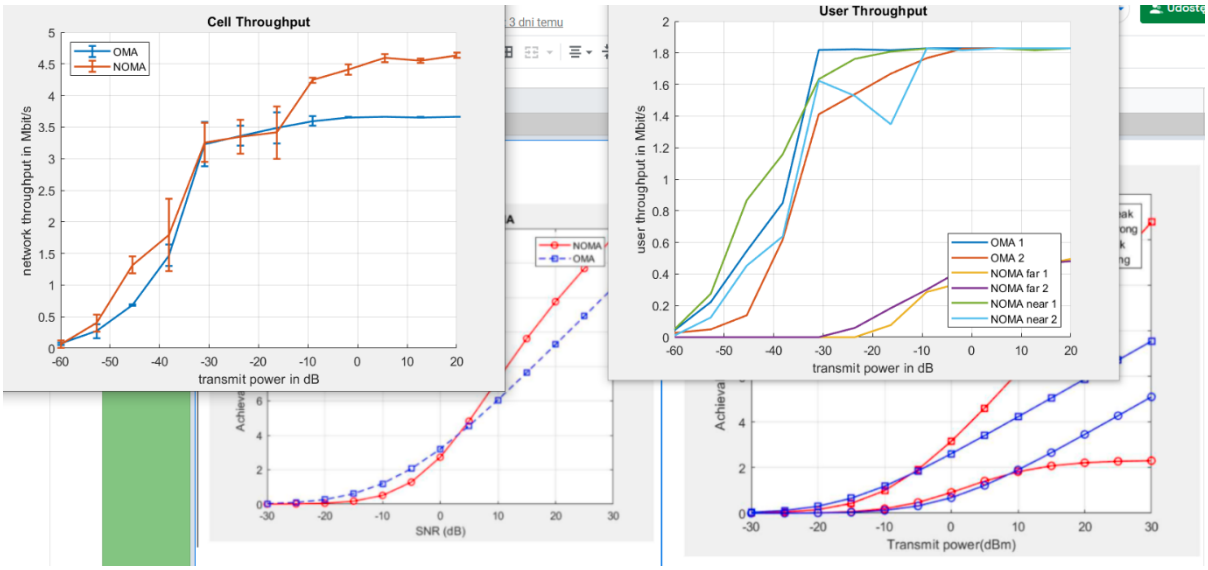


Fig. 3. Comparison of users' individual achievable rates.

WNIOSKI: wyniki w symulatorze 5G SLS Vienna pokazują 3-4 niższe capacity niż w artykule odniesienia.

2. Sprawdzamy wpływ wysokości stacji bazowej

Scenariusz	Ustawienia	Komentarz
launcherNOMA	**SISO** 5G BW 1,4MHz 4xOMA + 2xNOMA UE1,2 = pathloss ca 80 UE3,4 = pathloss ca 115-120 Channel=PedA	
	`antennaOmni.height = 20;`	



Wniosek: Wpływ wysokości stacji bazowej na wyniki jest POMIJALNY. Ale trzeba pamiętać że u nas na wykresie jest „BS Tx power”.

Paper12

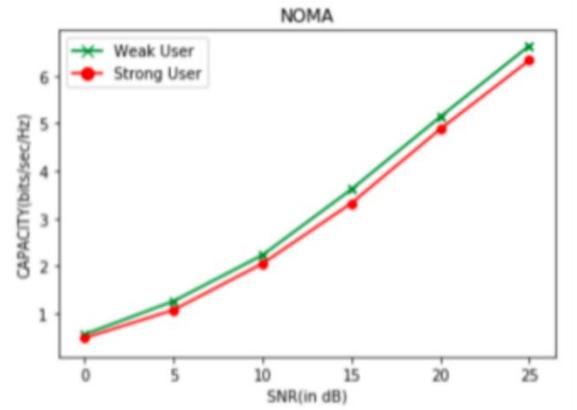


Fig. 3. Capacity analysis using PD-NOMA

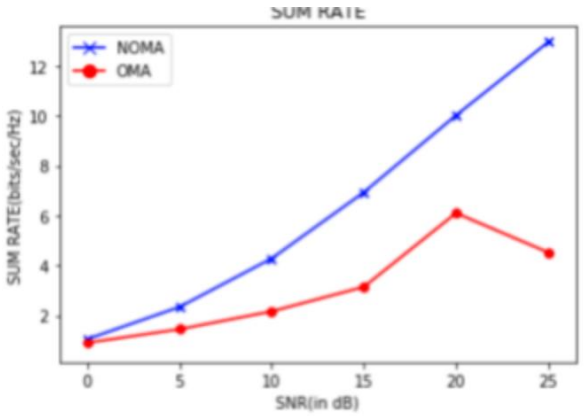
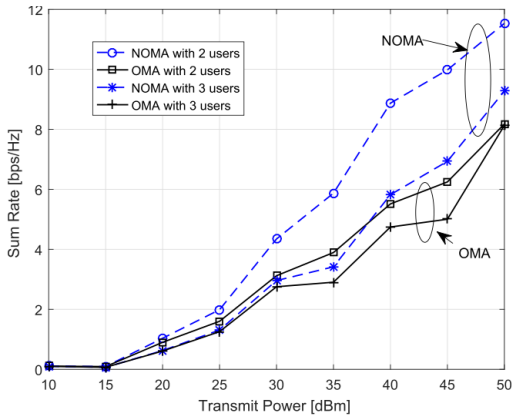


Fig. 4. Comparative analysis of sum rate

Paper17



ig. 5. Sum rates of MIMO-NOMA and MIMO-OMA vs. the transmit ower.

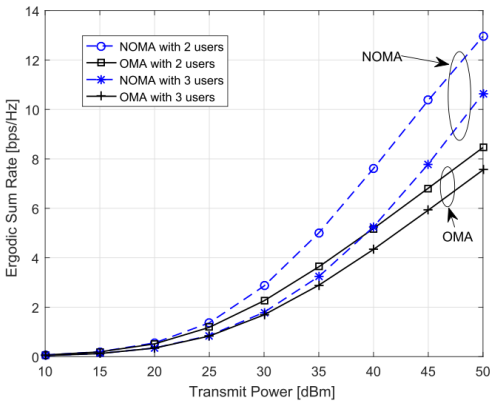
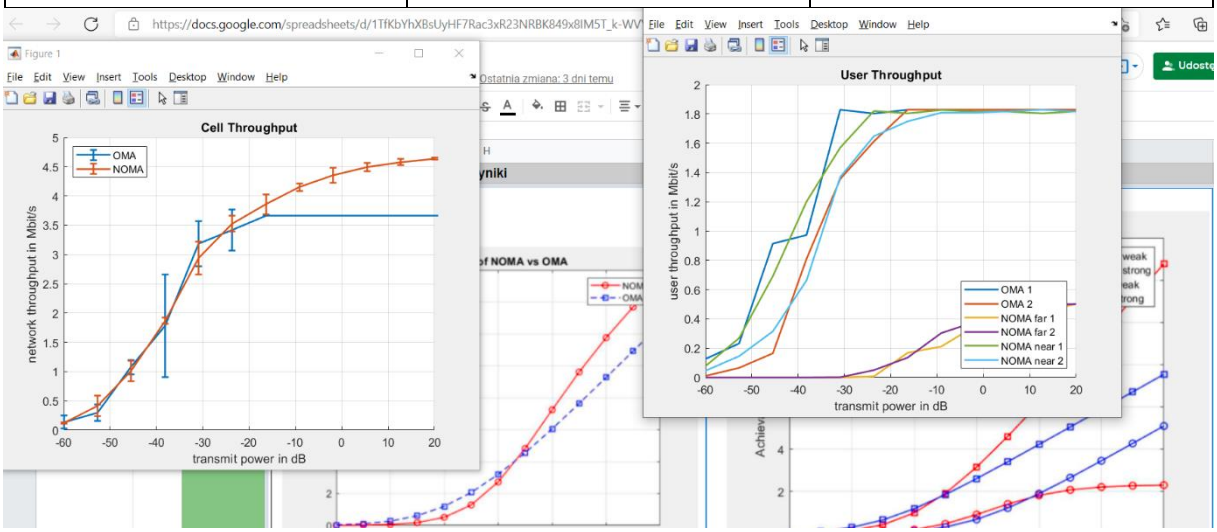


Fig. 6. Ergodic sum rates of MIMO-NOMA and MIMO-OMA vs. the transmit power.

3. Teraz test po zmianie kanału na **Rayleigh**

Scenariusz	Ustawienia	Komentarz
launcherNOMA	SISO 5G BW 1,4MHz 4xOMA + 2xNOMA UE1,2 = pathloss ca 80 UE3,4 = pathloss ca 115-120 Channel=Rayleigh antennaOmni.height=0;	



4. Teraz z kanałem Rayleigh + MIMO (2x2)

Scenariusz	Ustawienia	
launcherNOMA	MIMO 5G BW 1,4MHz 4xOMA + 2xNOMA UE1,2 = pathloss ca 80 UE3,4 = pathloss ca 115-120 Channel=Rayleigh antennaOmni.height=0;	

Poniżej przypomnienie maksymalnej przepływności jaka jest spodziewana dla takich ustawień.

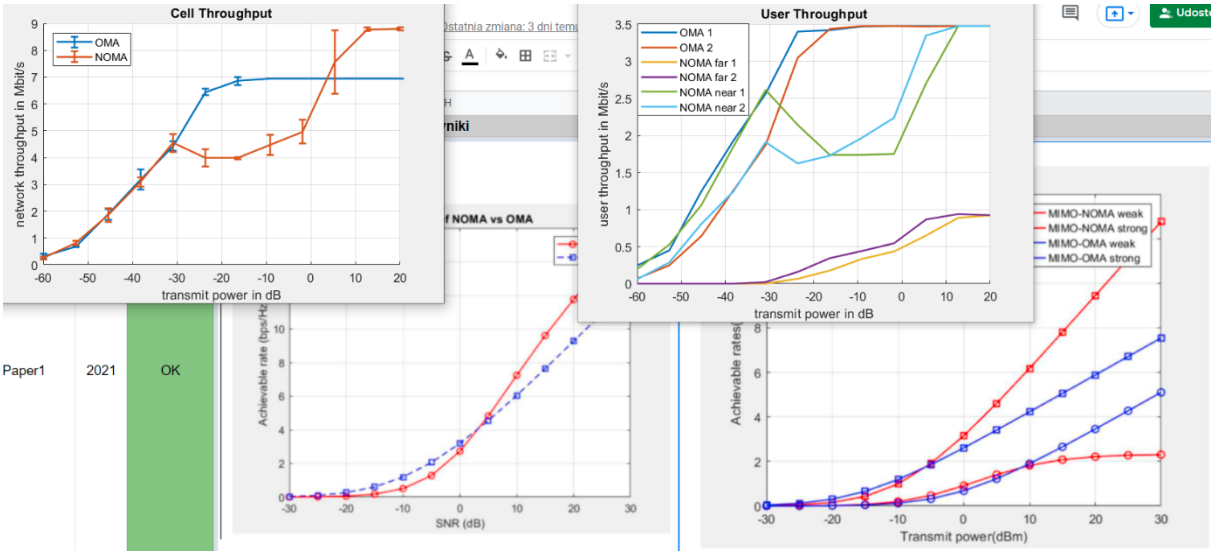
Maximum LTE DL throughput:

Bandwidth: 1.4 MHz
Modulation: 64QAM
MIMO: MIMO 2x2

Maximum throughput: 8.784 Mbps

```
simple4G5G.m | launcherSimulationTime.m | launcherSimple4G5G.m | UserNoma.m | CqiParameters.m | LteCqiParametersTS36213NonBLCEUE1withoutCalibration.m | LteCqiParametersTS36213NonBLCEUE1.m
ods
function obj = LteCqiParametersTS36213NonBLCEUE1(transmissionParameters)
    % class constructor - sets CQI table according to TX 36.213 V13.2.0 (2016-06) (i.e., Table 7.2.3-1)
    %
    % input:
    %   transmissionParameters: [1x1]handleObject parameters.transmissionParameters.TransmissionParameters

    % set CQI table according to standard
    cqi                     = [0      1      2      3      4      5      6      7      8      9      10     11     12     13     14     15];
    obj.modulationType      = [1      2      2      2      2      2      2      3      3      3      4      4      4      4      4      4];
    obj.modulationOrder     = [0      2      2      2      2      2      2      4      4      4      6      6      6      6      6      6];
    obj.modulationName      = {'None' 'QPSK' 'QPSK' 'QPSK' 'QPSK' 'QPSK' 'QPSK' '16QAM' '16QAM' '16QAM' '64QAM' '64QAM' '64QAM' '64QAM' '64QAM' '64QAM'};
    obj.codingRateX1024     = [0      78     120    193    308    449    602    378    490    616    466    567    666    772    873    948];
    obj.efficiency          = [0      0.1523 0.2344 0.3770 0.6016 0.8770 1.1758 1.4766 1.9141 2.4063 2.7305 3.3223 3.9023 4.5234 5.1152 5.5547];
    obj.betaMIESMCalibration = [1     3.07   4.41   0.6    1.16   1.06   1.06   0.87   1.01   1.04   1.03   1.11   1.01   1.07   1      1.05];
    obj.nCqi                = 16;

    obj.blerCurveFiles      = cell(obj.nCqi,1);
    obj.blerCurveFiles{1}   = ''; % no bler curve for zero CQI
```

Poniżej porównanie wyników - górne wykresy RATfor5G+ a dolne artykuł “Paper1”.

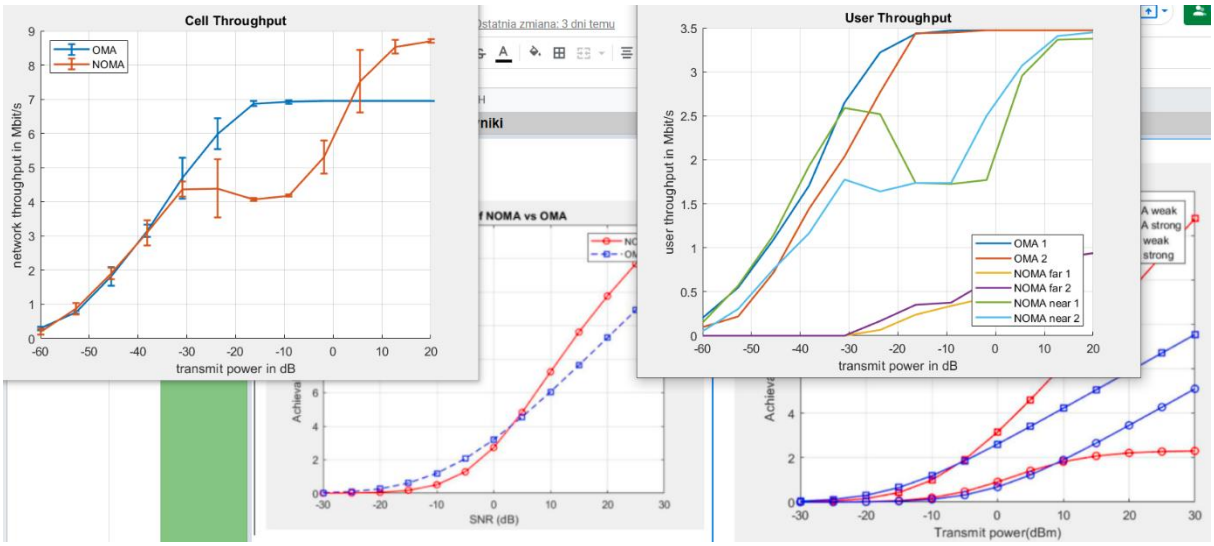


WNIOSKI: jak widać jest duża różnica jeśli chodzi o capacity! Praktycznie dwukrotna, natomiast wartości są już bardziej zbliżone do capacity z artykułu. Wydaje się że wartości w artykule Paper1 są zbyt wyidealizowane.

5. Test z kanałem **PedA** + MIMO (2x2)

Scenariusz	Ustawienia	Komentarz
launcherNOMA	MIMO 5G BW 1,4MHz 4xOMA + 2xNOMA UE1,2 = pathloss ca 80 UE3,4 = pathloss ca 115-120 Channel=**PedA** antennaOmni.height=0;	

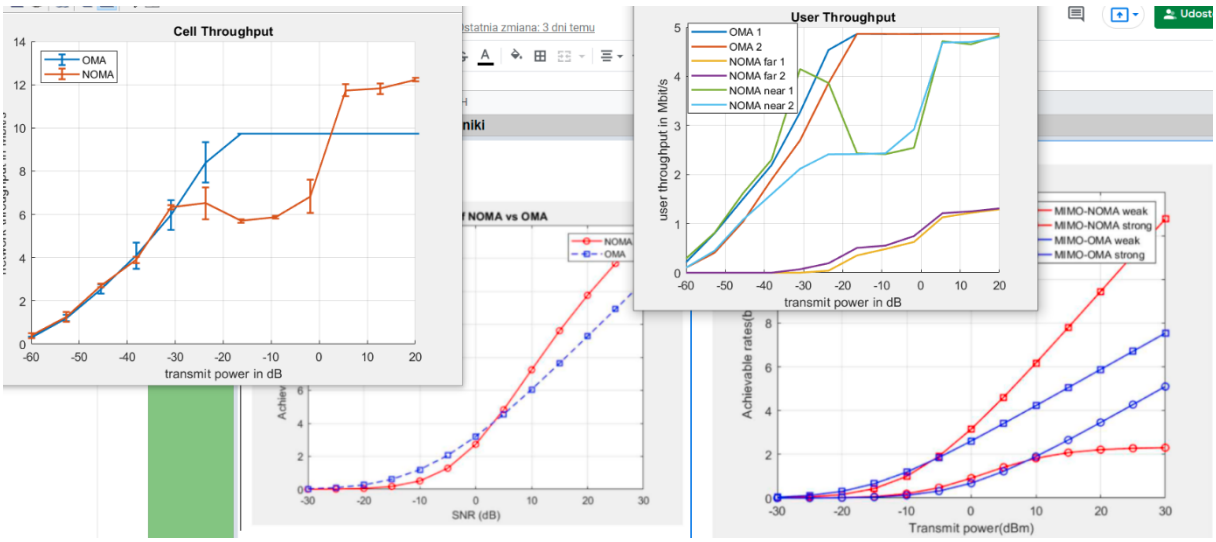
UWAGA: trzeba pamiętać że zarówno tak, jak i wcześniejsze symulacje wykorzystują dzielnik „/1.4” żeby uzyskać „capacity” zamiast przepływności.



6. Test z kanałem **PedA** + MIMO (2x2) + bez normalizacji pod kątem „capacity”

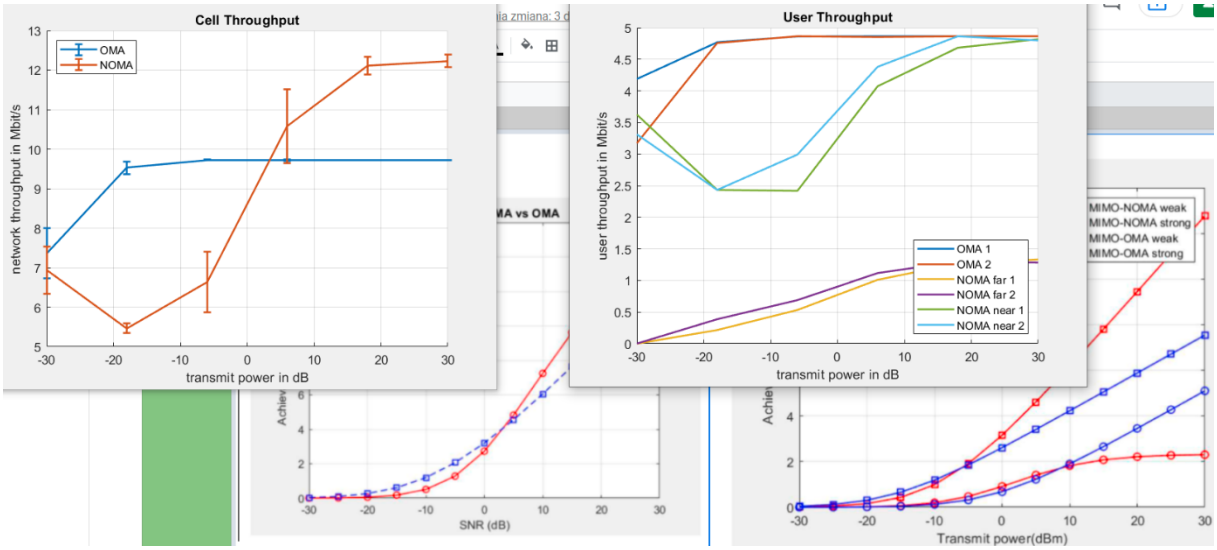
Scenariusz	Ustawienia	Komentarz
launcherNOMA	MIMO 5G BW 1,4MHz 4xOMA + 2xNOMA UE1,2 = pathloss ca 80 UE3,4 = pathloss ca 115-120 Channel=**PedA** antennaOmni.height=0;	
	Do tego **brak czynnika** normalizującego: tzn normalize=1	Chodzi o zmienną „normalize” w pliku „launcherNOMAOriginal”. To znaczy że wykresy **poniżej przedstawiają przepływność a NIE capacity.**

Górne wykresy to 5G SLS Vienna, a dolne artykuł “Paper1”.



UWAGA: na powyższym **wykresie faktycznie widać przepływności (Mbps) a nie capacity** jak było we wszystkich testach powyżej.

7. A teraz zmieniamy ustawienia mocy stacji bazowej - od -30 do 30dB.



8. A teraz zmieniamy pasmo z BW=1.4Mhz **na BW=10Mhz**, żeby się upewnić że wyniki zostaną mniej więcej takie same – skoro pokazujemy „capacity", a więc wartość znormalizowaną.

<u>Poniżej jest pokazana POJEMNOŚĆ a nie przepływność, czyli robię normalizację (dzielenie przez 1.4Mhz).</u>

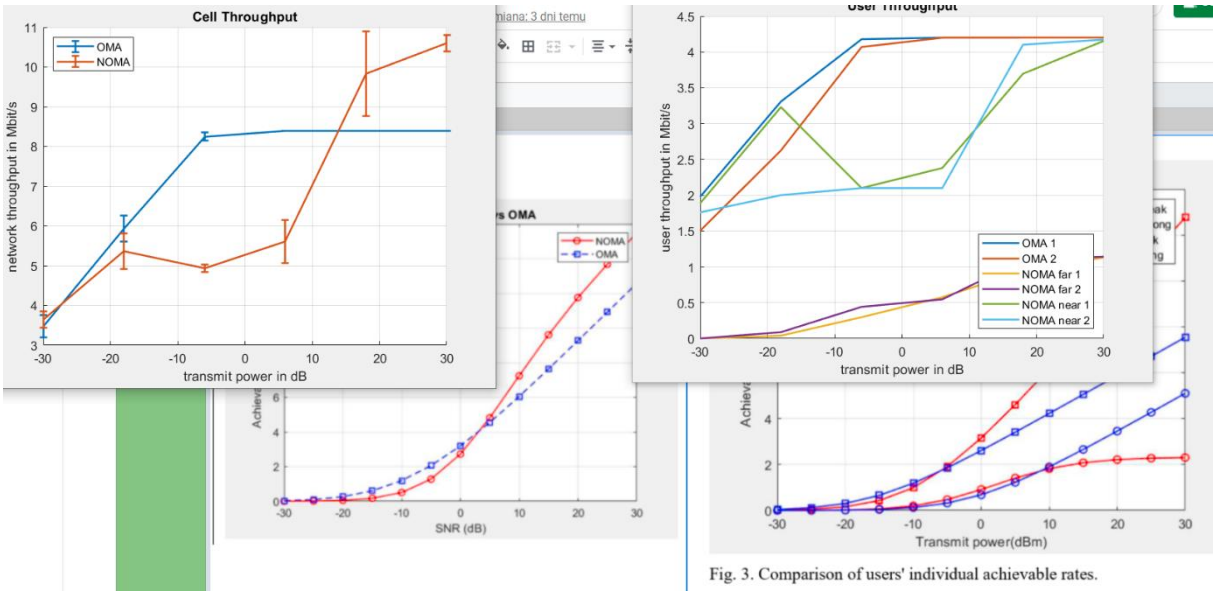


Fig. 3. Comparison of users' individual achievable rates.

<u>WNIOSEK</u>: jak widać pojemności **nieco spadły**, w porównaniu do symulacji przed zwiększeniem pasma 1.4->10Mhz. Pytanie – dlaczego, skoro jednostki są w bit/s/Hz… więc raczej nie powinno być zmiany przez samo zwiększenia pasma..

9. A teraz **usuwam normalizację** (przy szerokim paśmie = 10Mhz), żeby sprawdzić wynikową przepływność w Mbit/s, tj że się podniesie. **UWAGA tutaj normalizacja jest robiona sztucznie** (tj dzielenie przez 8, a nie przez 1.4Mhz).

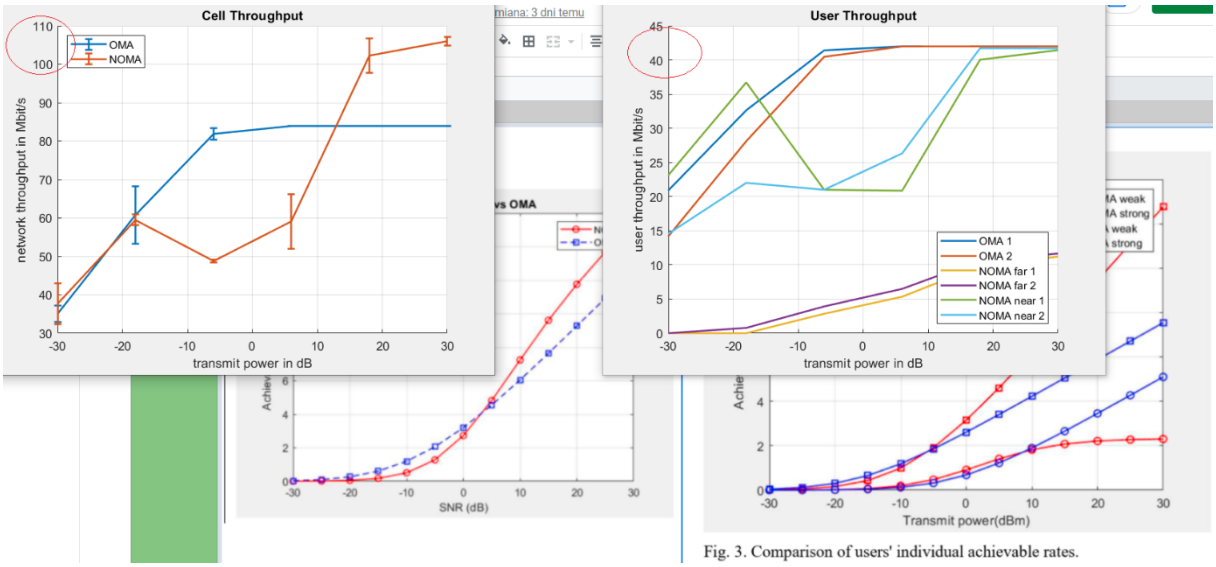


Fig. 3. Comparison of users' individual achievable rates.

WNIOSEK: faktycznie widać że przepływność idzie mocno do góry. Zatem warto stosować normalizację, żeby jasno i w zunifikowany sposób pokazywać wyniki.

Normalizacja przywrócona i BW=1.4MHz

10. A teraz **z powrotem kanał Rayleigh** zamiast obecnego PedA + MIMO.

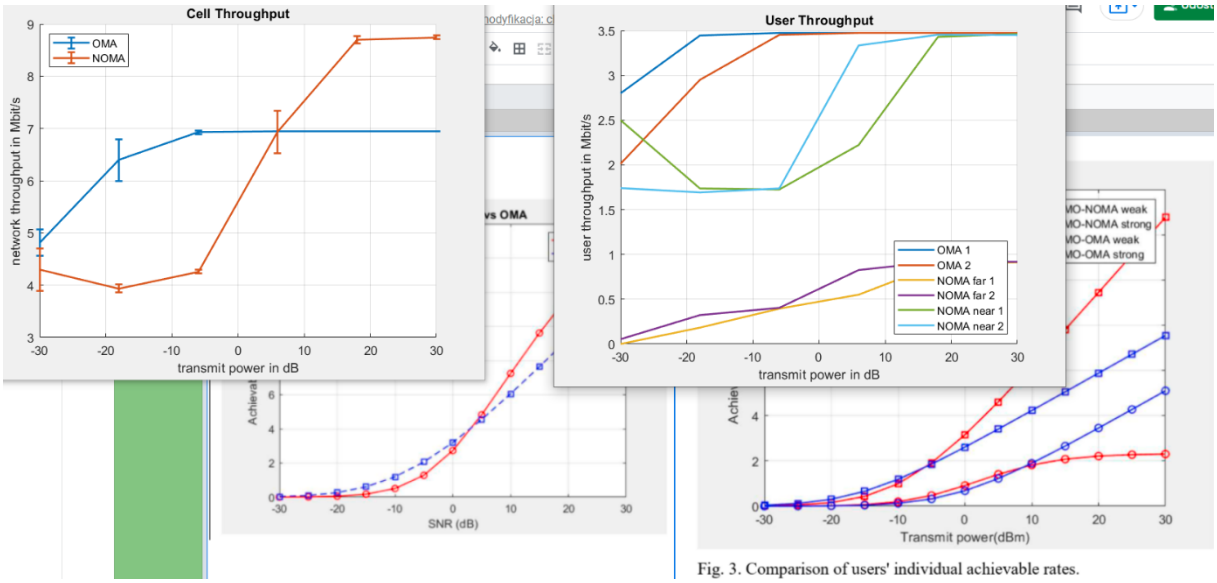


Fig. 3. Comparison of users' individual achievable rates.

11. A teraz ustawiam **z powrotem „SISO”**, reszta pozostaje bez zmian w stosunku do poprzedniej symulacji.

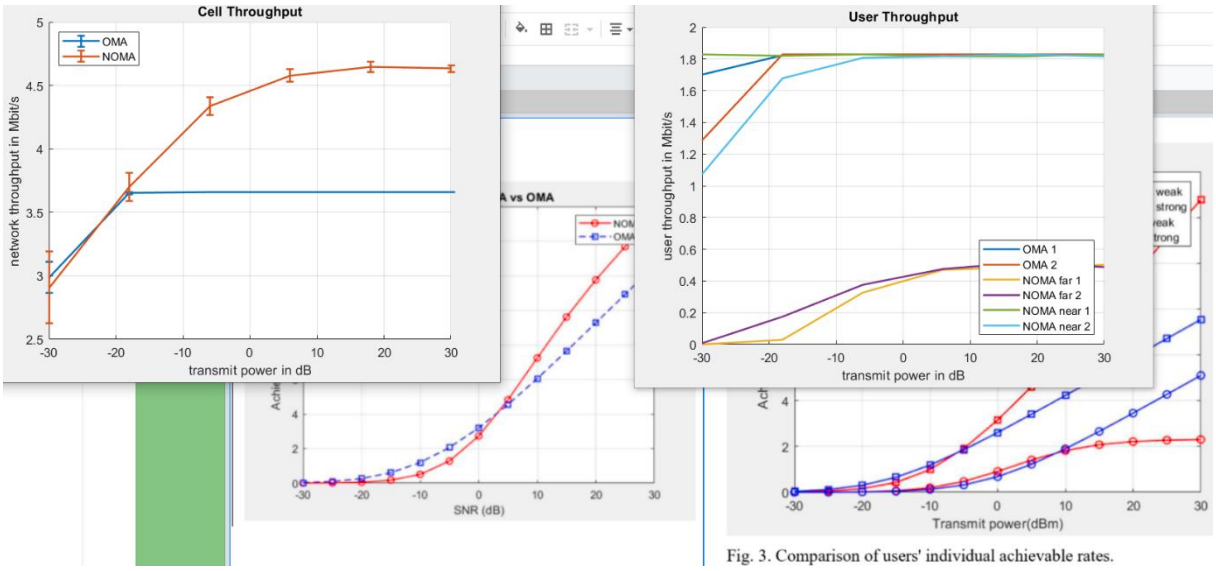


Fig. 3. Comparison of users' individual achievable rates.

12. A teraz zmieniam technologię z 5G **na LTE**.

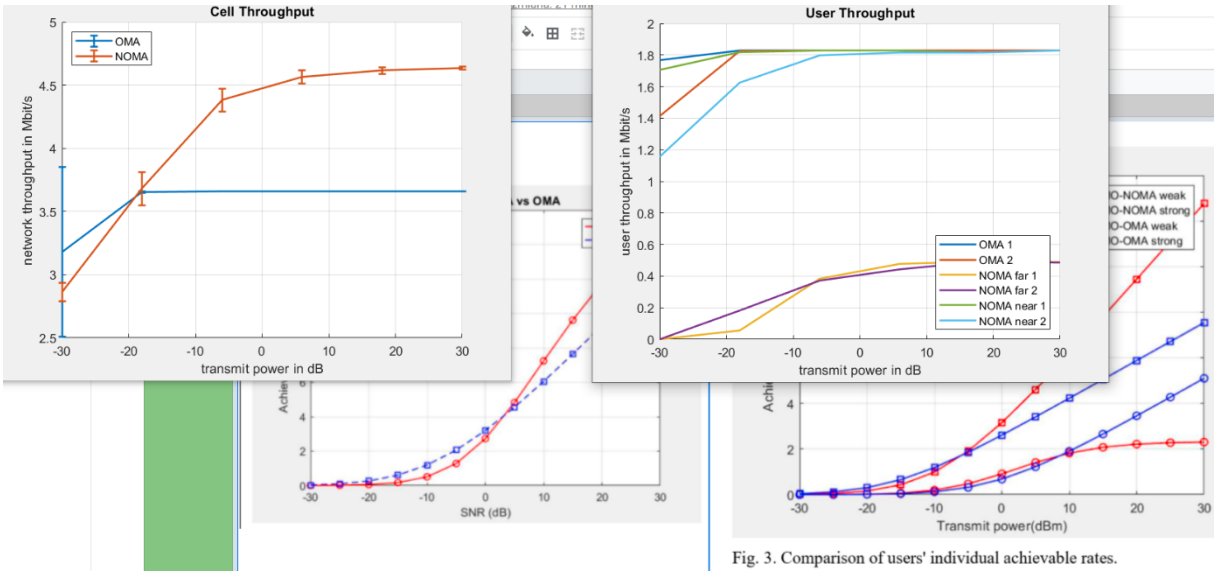
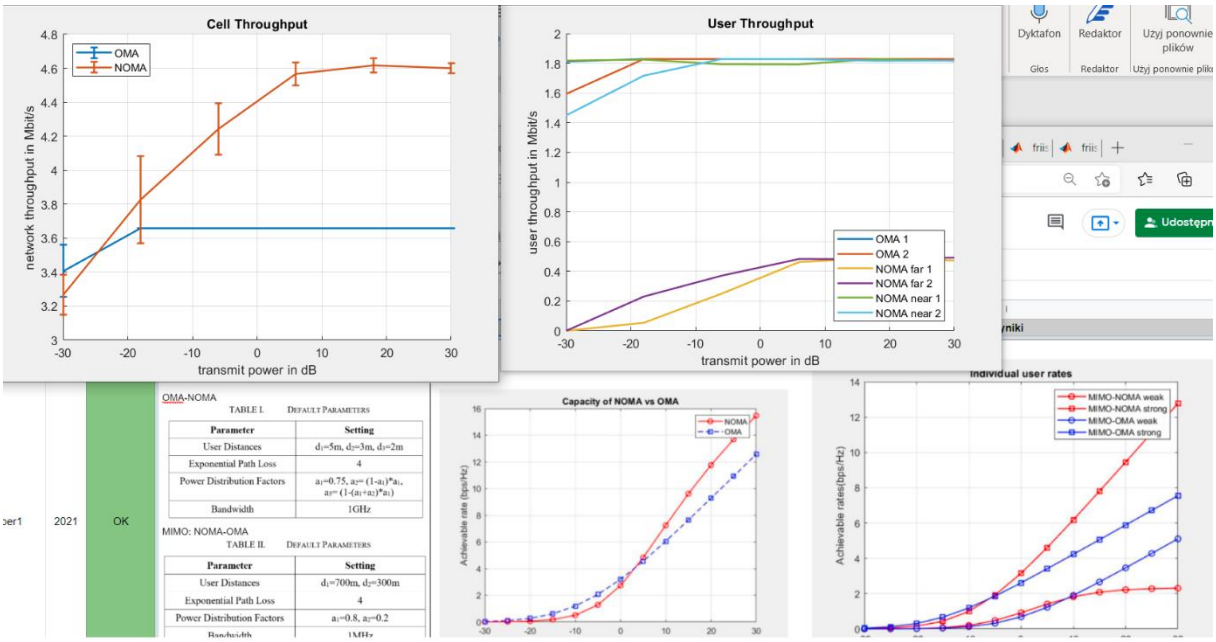


Fig. 3. Comparison of users' individual achievable rates.

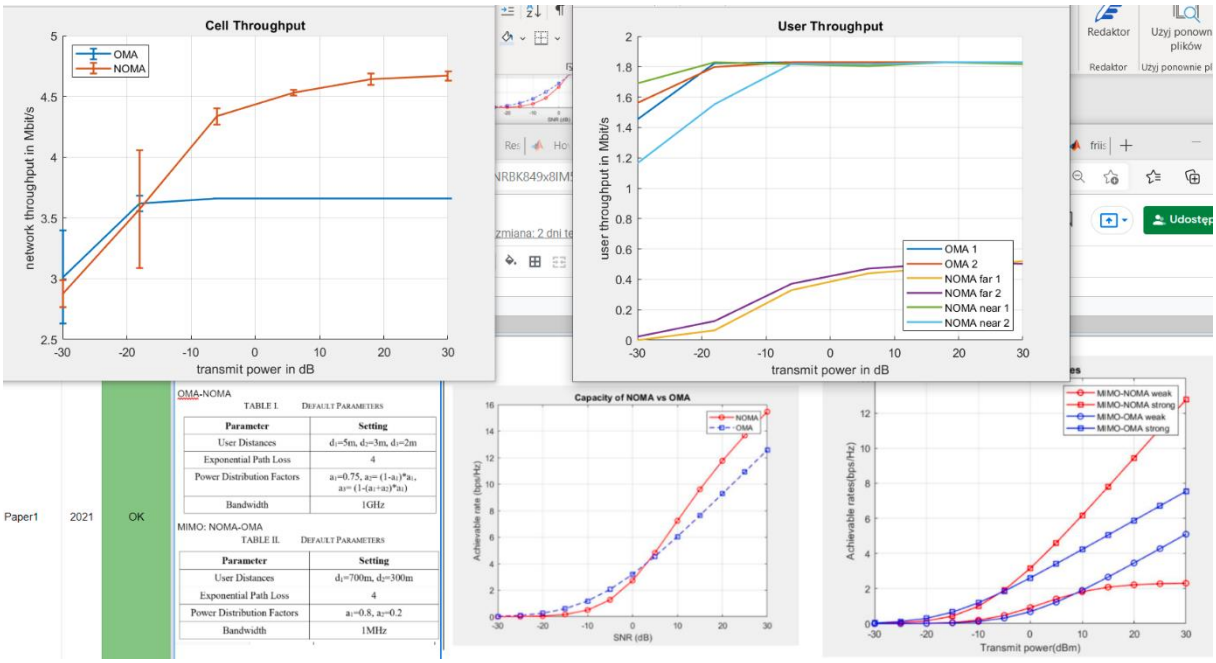
WNIOSEK: wygląda na to, że niestety **zmiana technologii z 5G-LTE-5G nie zmienia wyników**. Zatem pytanie o poprawność implementacji LTE/5G w symulatorze.

13. Kolejny test: Rayleigh + SISO + **alpha=4** + LTE

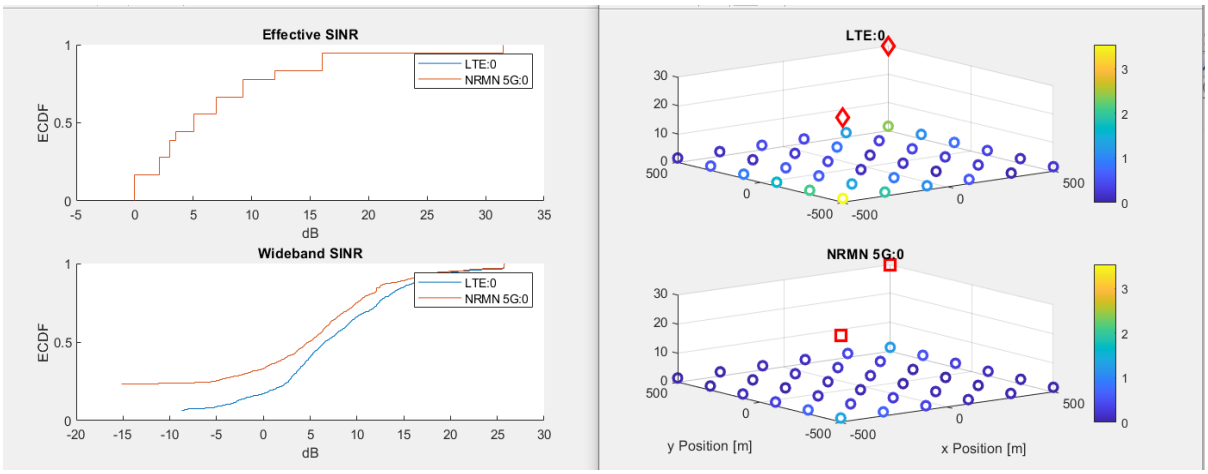
Główna zmiana -> path loss exponent



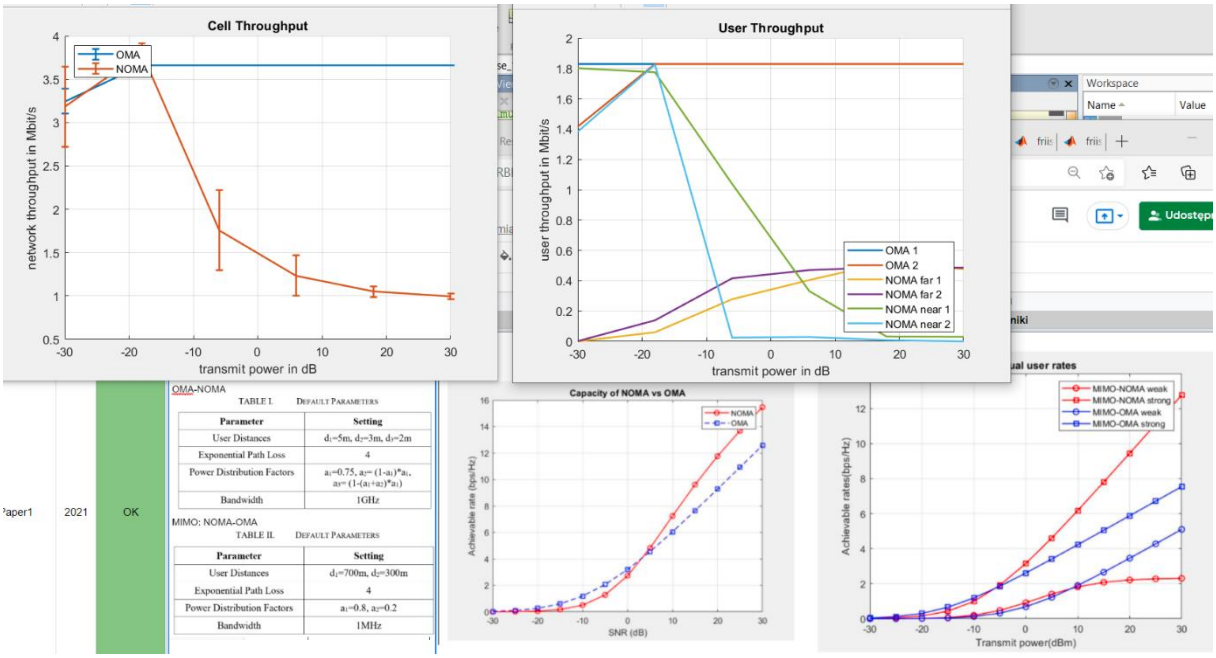
…jeszcze **zmiana LTE->5G** (przynajmniej tak pokazują parametry konfiguracyjne w symulatorze) i powtórka symulacji:



Jak widać nie ma wpływu niestety 5G vs LTE. W ramach testu, sprawdzono także **scenariusz porównujący ze sobą 4G i 5G** (ale nie-NOMA) i wygląda, że jedyna różnica na 7-miu wykresach, jaka występuje to jest Wideband SINR oraz przepływność (tu **różnica jest minimalna**). Przy okazji udało się naprawić jeden mały błąd z indeksowaniem tablicy userów (dla dowolnego scenariusza).

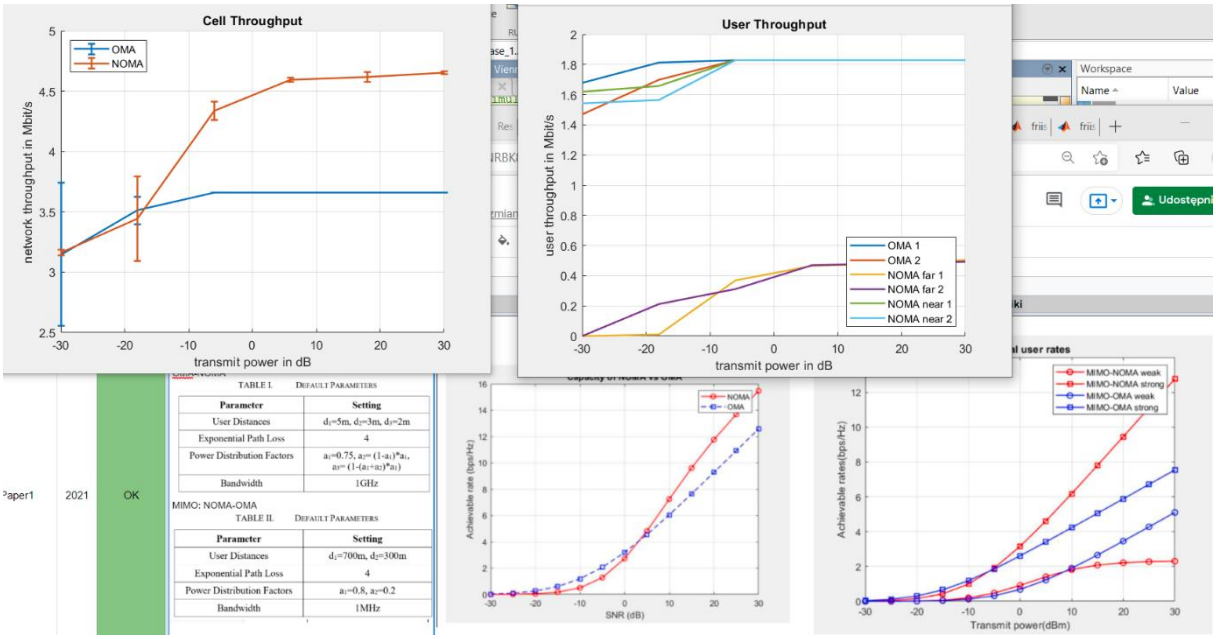


14. Wszystko zostaje tak samo tylko **zmieniam „maxFarCQI" z 6->10**, a zatem odległy użytkownik może mieć wyższą wartość CQI.



..widać że mocno spada przepływność użytkowników z dekodowaniem SIC/NOMA.

To teraz dla odmiany „maxFarCQI=4"

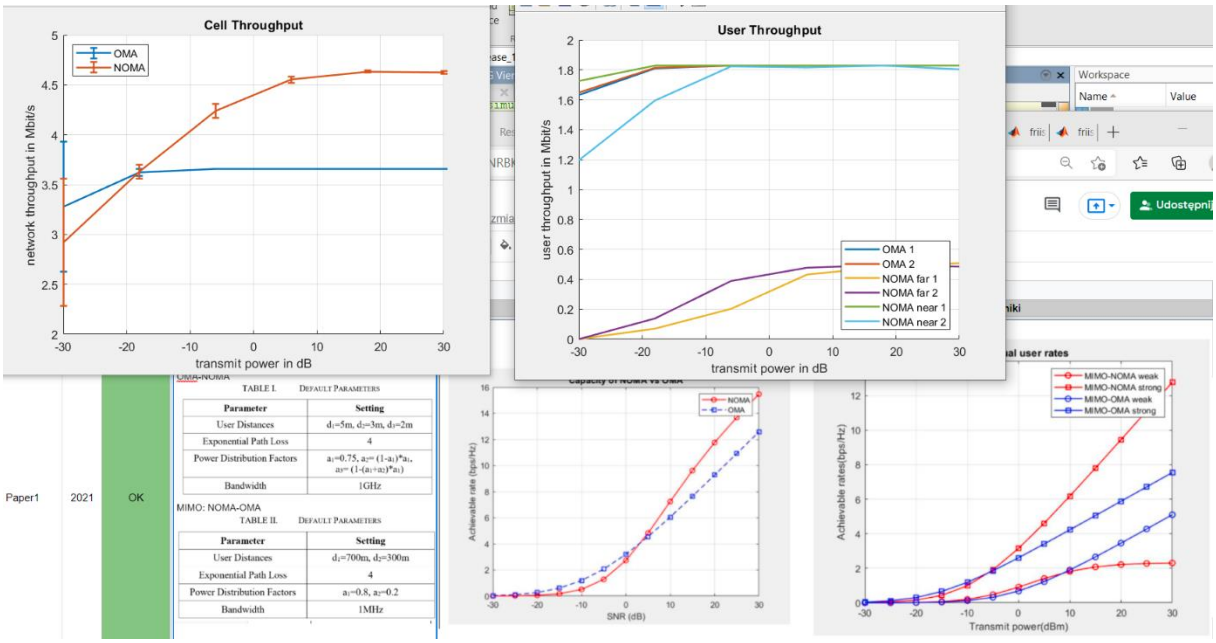


WNIOSEK: jak widać w tę stronę (tj. zmniejszanie wartości „maxFarCQI") już nie ma takiego wpływu. Najprawdopodobniej chodzi o to, że jeśli jest podana za duża wartość, to dopuszcza stosunkowo niskie modulacje i dlatego spada przepływność.

USTAWIENIA: zatem **przywracamy wartość domyślną** (maxFarCQI=6)

15. Następnie sprawdzam wpływ różnicy path loss która jest niezbędna do sparowania userów.

params.noma.deltaPairdB = 10; (domyślnie jest 15)

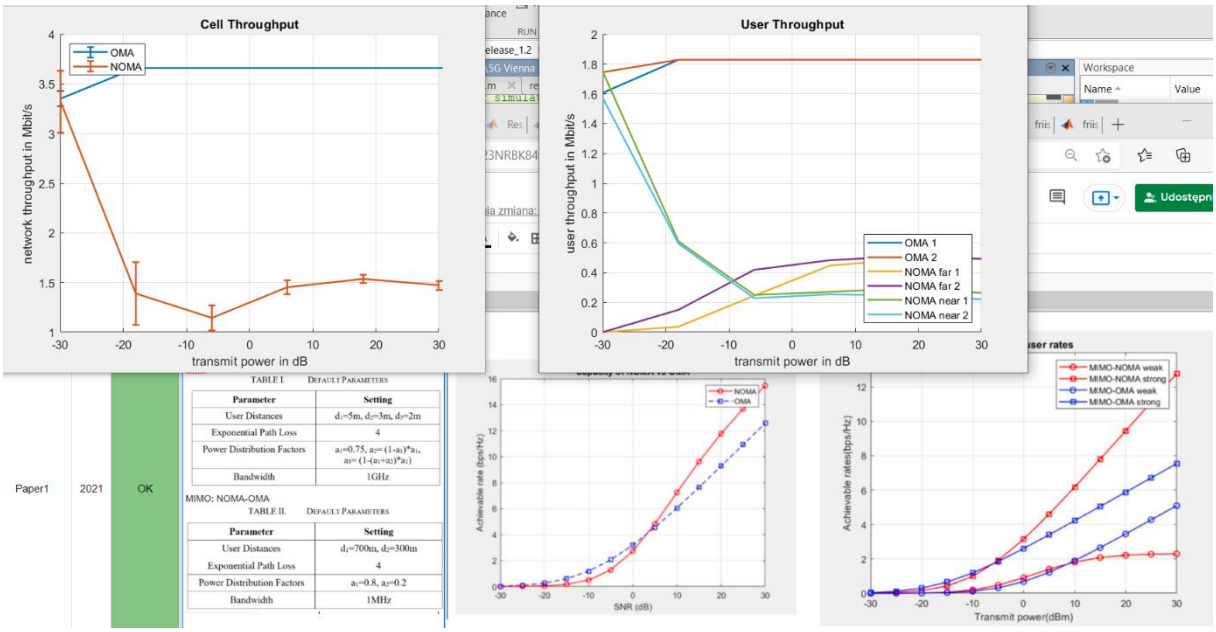


WNIOSEK: nie widać różnicy, ale to wynika na pewno z faktu że userzy są tylko 2+2 (far+near). To ustawienie **miałoby sens gdyby w symulacji było dużo userów**, w różnych odległościach od stacji bazowej.

USTAWIENIA: przywracam wartość domyślną params.noma.deltaPairdB = 15;

16. Sprawdzamy wpływ czynnika „nie-idealności" SIC

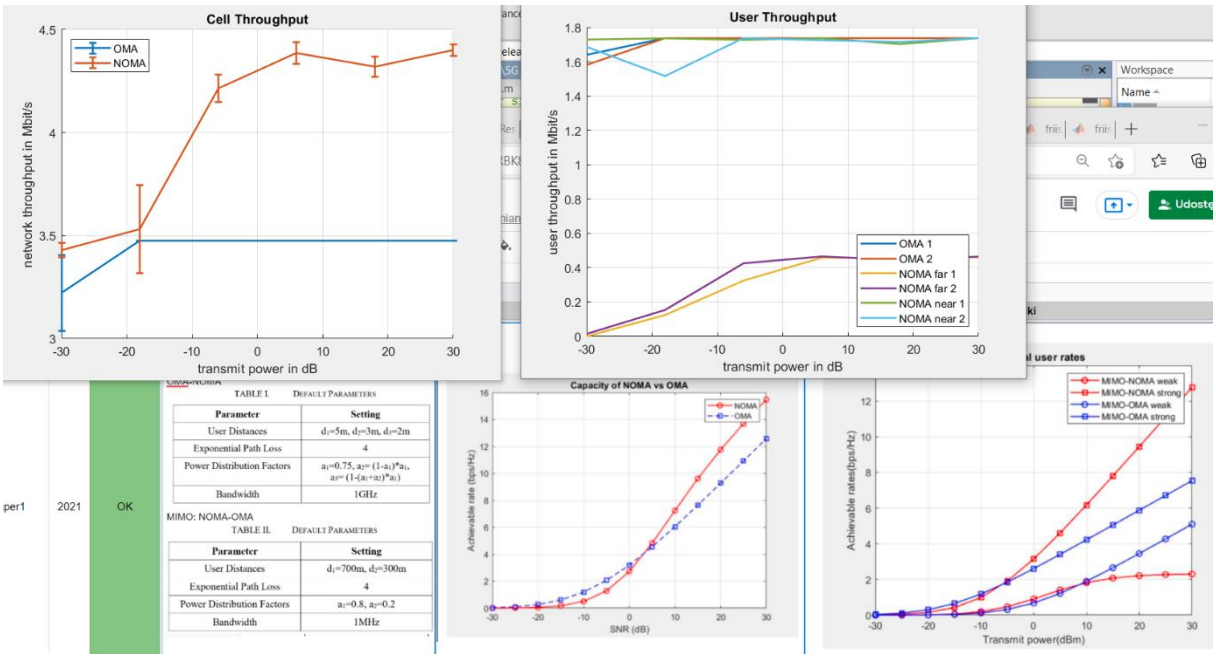
params.noma.**interferenceFactorSic = 0.3; (zamiast „0”)**



WNIOSEK: jak widać przepływność NOMA mocno spada przy nie idealnym działaniu SIC.

17. Test z ustawieniami dla **MISO**

Scenariusz	Ustawienia	Komentarz
launcherNOMA	**MISO** (2Tx, 1Rx) 5G (na szybko LTE wyniki wychodzą podobnie) BW 1,4MHz 4xOMA + 2xNOMA UE1,2 = pathloss ca 80-85 UE3,4 = pathloss ca 115-120 Channel=**Rayleigh** Antena = Omni Alpha = 4	

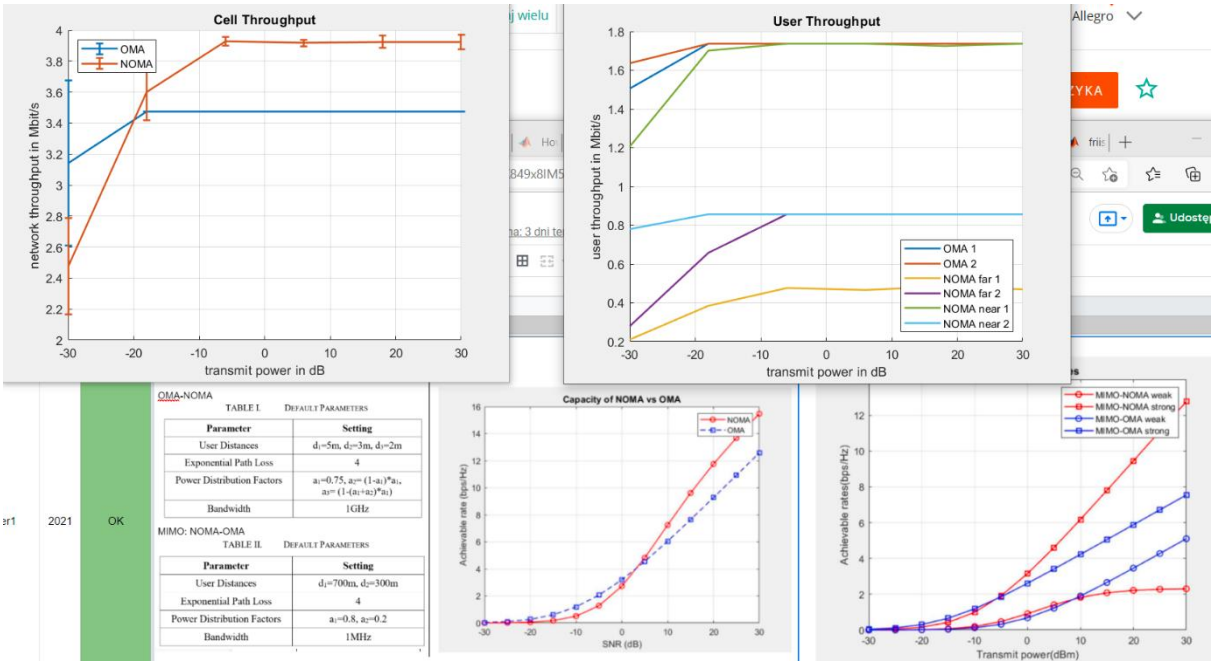


Wniosek: Capacity jest trochę niższe niż dla MIMO (zamiast 4.6 -> 4,4 dla NOMA, 3,6->3,4 OMA).

…jeszcze sprawdzenie, co się stanie jeśli „zbliżymy userów" pod kątem różnicy w tłumieniu pomiędzy ich kanałami..

User1noma = 105db PL (far user)

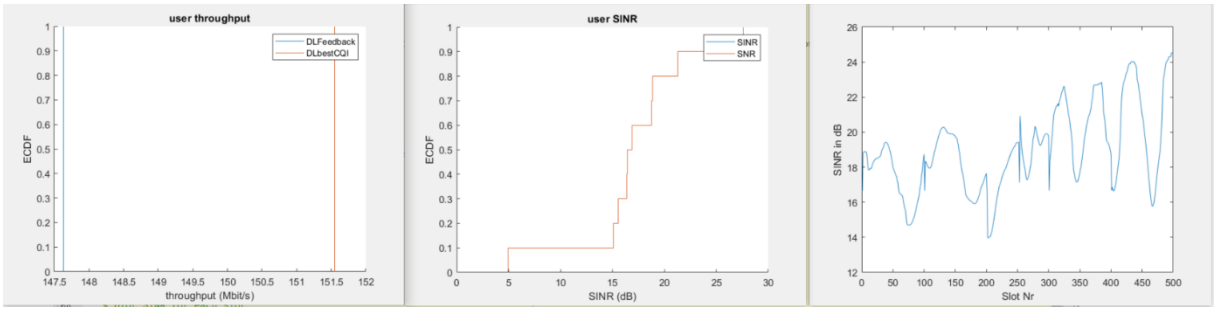
User2noma = 100db PL (far user)



WNIOSKI: przepływności „near usera" spadła, przepływność „far usera" wzrosła

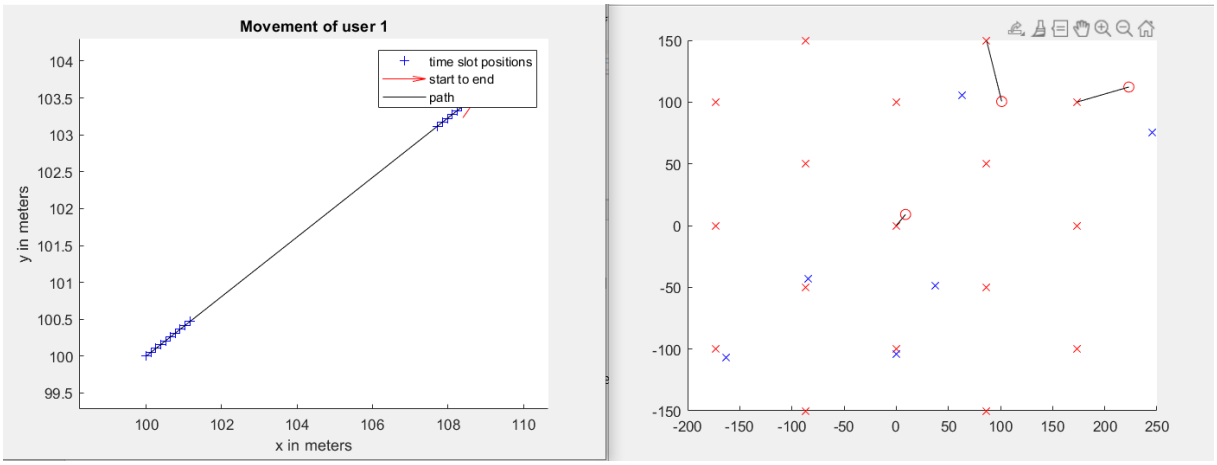
Paper1 --- podejście drugie (SNR)

1. Test dla „launcherFeedback.m” – żeby zobaczyć jakie są możliwe wykresy dostępne

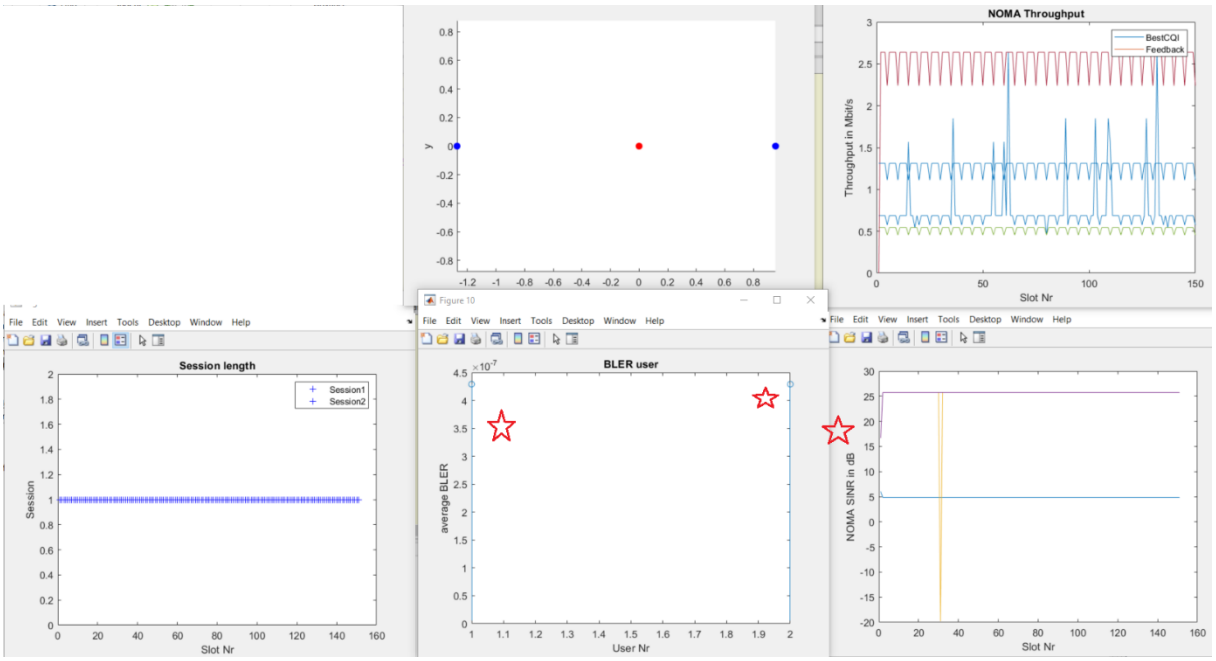


Warto także sprawdzić ResultsFull.m pod kątem wykresu SINR ale w powiązaniu z Thp

2. Uruchamiam jeszcze „LauncherUsermovement.m” żeby sprawdzić wyniki



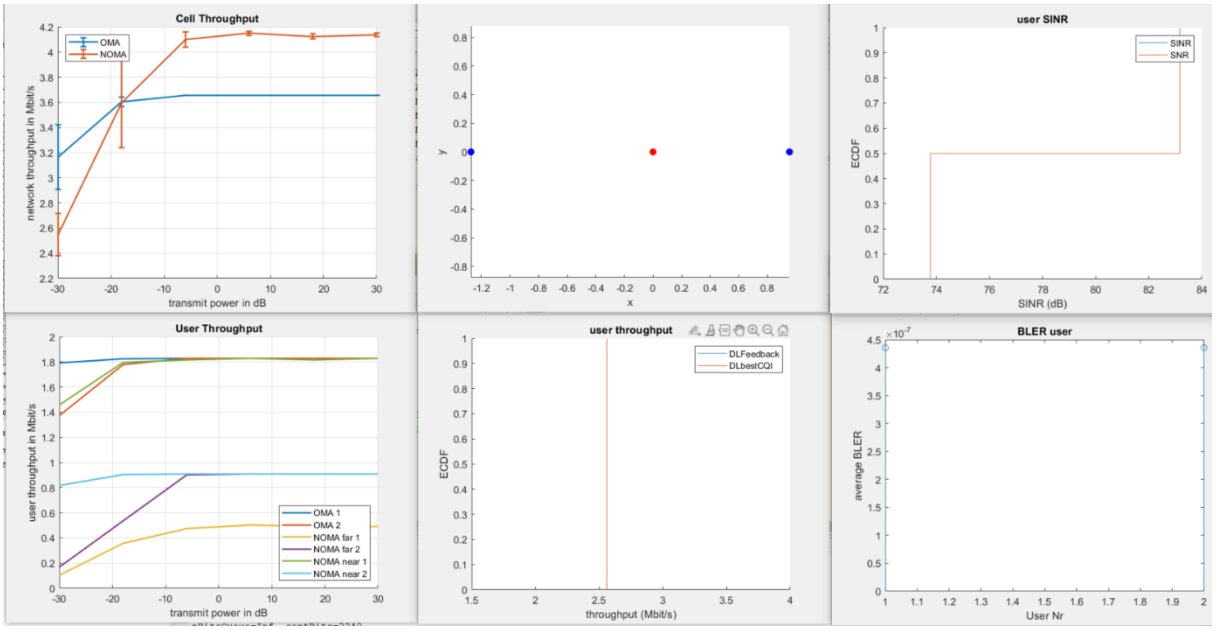
3. Wracam do launcherNOMA.m (to jest wersja zmodyfikowana) żeby sprawdzić wszystkie wykresy – zanim zostaną zmienione inne parametry.



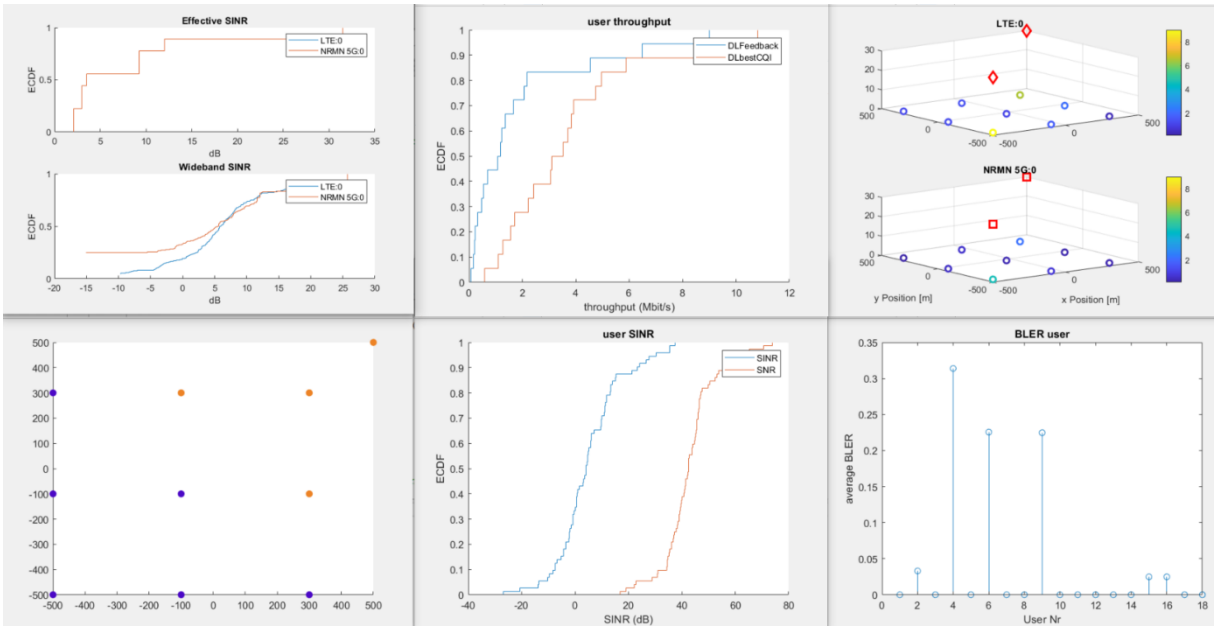
4. I jeszcze pomiar za pomocą pliku **oryginalnego** launcherNomaORIGINAL.m – czyli bez zmian jakichkolwiek, tylko defaultowe ustawienia

<u>Zatem bez:</u>

- dodatkowych wykresów (per NOMa, per OMA)
- pokazywania sesji userów



5. Jeszcze sprawdzam **launcherSimple4G5G**.m



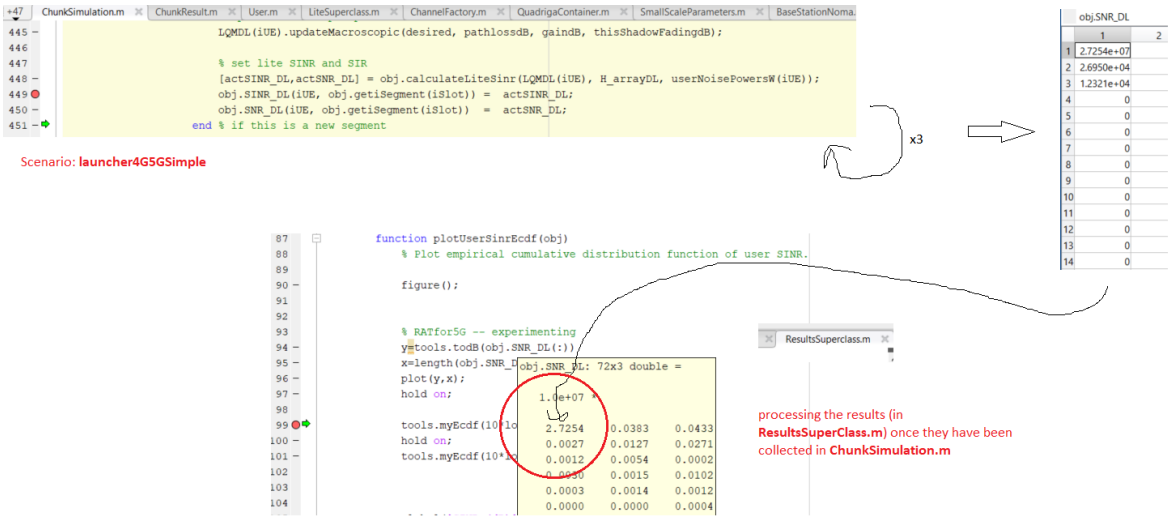
Najciekawszy jest wykres „Figure 5”

```
86      methods
87          function plotUserSinrEcdf(obj)
88              % Plot empirical cumulative distribution function of user SINR.
89
90              figure();
91
92              tools.myEcdf(10*log10(obj.SINR_DL(:)));
93              hold on;
94              tools.myEcdf(10*log10(obj.SNR_DL(:)));
95
96              xlabel('SINR (dB)');
97              ylabel('ECDF');
98              title('user SINR');
99              legend('SINR','SNR');
100         end
```

ResultsSuperclass.m

Analiza zapełniania obiektu danymi:

- w ChunkSimulation, co iSlot, zapełniane są dane dla danego usera jeśli chodzi o SINR, SNR
- są przekazywane na bieżąco do ResultsFull, oraz ResultsSuperclass
- na koniec są rysowane za pomocą ResultsFull->wykresy albo w środku samego launcherXYZ



KONFIGURACJA scenariusza

simple4G5G.m

```
% set user parameters
% defines a usergroup equiped with a LTE technology
usertype1 = parameters.user.PredefinedPositions();
usertype1.positions             = [positionX; positionY; positionZ];
usertype1.indoorDecision        = parameters.indoorDecision.Geometry();
usertype1.nRX                   = 1;
usertype1.channelModelTypes     = parameters.setting.ChannelModel.PedA;
usertype1.technology            = parameters.setting.NetworkElementTechnology.LTE;
usertype1.numerology            = 0;
usertype1.trafficModelType      =  parameters.setting.TrafficModelType.FullBuffer;
params.userParameters('user1')  = usertype1;

% set user parameters
% defines a usergroup equiped with a NRMN_5G technology
usertype2 = parameters.user.PredefinedPositions();
usertype2.positions           = [positionX; positionY; positionZ];
usertype2.indoorDecision      = parameters.indoorDecision.Geometry();
usertype2.nRX                 = 1;
usertype2.channelModelTypes   = parameters.setting.ChannelModel.PedA;
usertype2.technology          = parameters.setting.NetworkElementTechnology.NRMN_5G;
usertype2.trafficModelType    =  parameters.setting.TrafficModelType.FullBuffer;
usertype2.numerology          = 0;
params.userParameters('user2')= usertype2;
```

Jak widać jest dwóch userów w scenariuszu a potem (poniżej obrazek) wywołanie symulacji tylko raz.

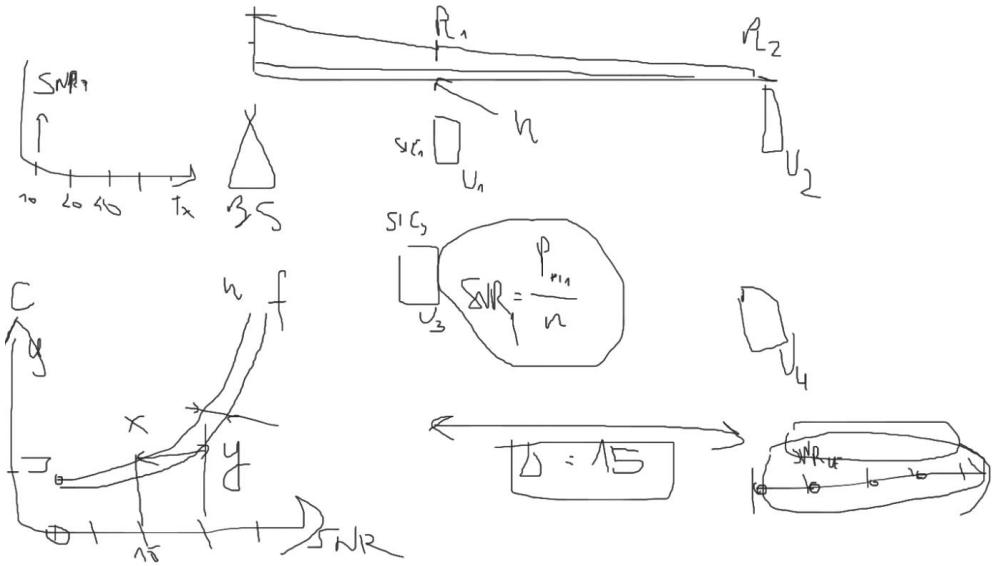
Symulacja – wywołanie:

launcherSimple4G5G.m

```
13
14      % This line launches the simulation with the scenario defined in
15      % scenarios
16      % For parallel simulations, see parameters.setting.SimulationType.
17 -    result = simulate(@scenarios.simple4G5G, parameters.setting.SimulationType.local);
18
19 -    disp("Start Postprocessing...")
20      % plot sinr values of each technologie respectively
21 -    figure();
22      % get sinr values
23 -    sinr = result.widebandSinr;
24 -    sinr = sinr.DL;
25 -    effSinr = result.effectiveSinr.DL;
26      % get users of each technology
27 -    users = [result.networkResults.userList];
```

Jak widać powyżej jest tylko jedna pętla symulacji…

Podeście do odpowiedniego pomiaru SNR (rozmowa z 04.02.2022) ? czyli żeby manipulować SNR jako „oś X” trzeba zmieniać moc nadawaną z eNB.



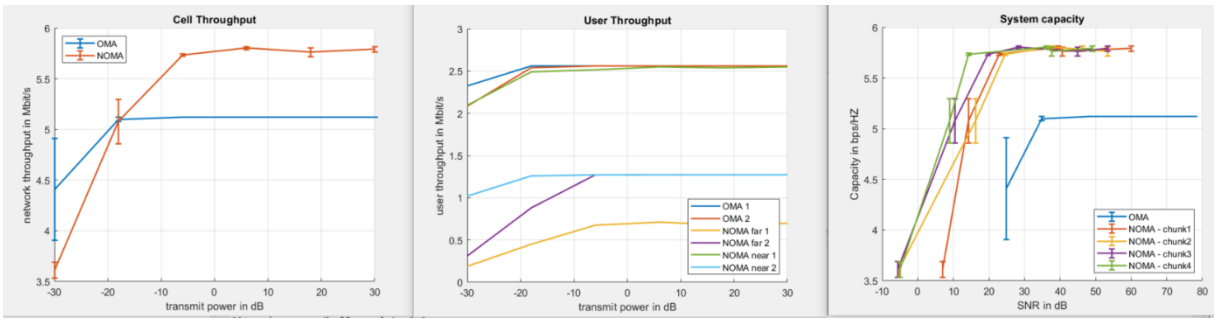
6. Testy dla scenariusza „launcherNomaMODIFIED.m”

testy SISO

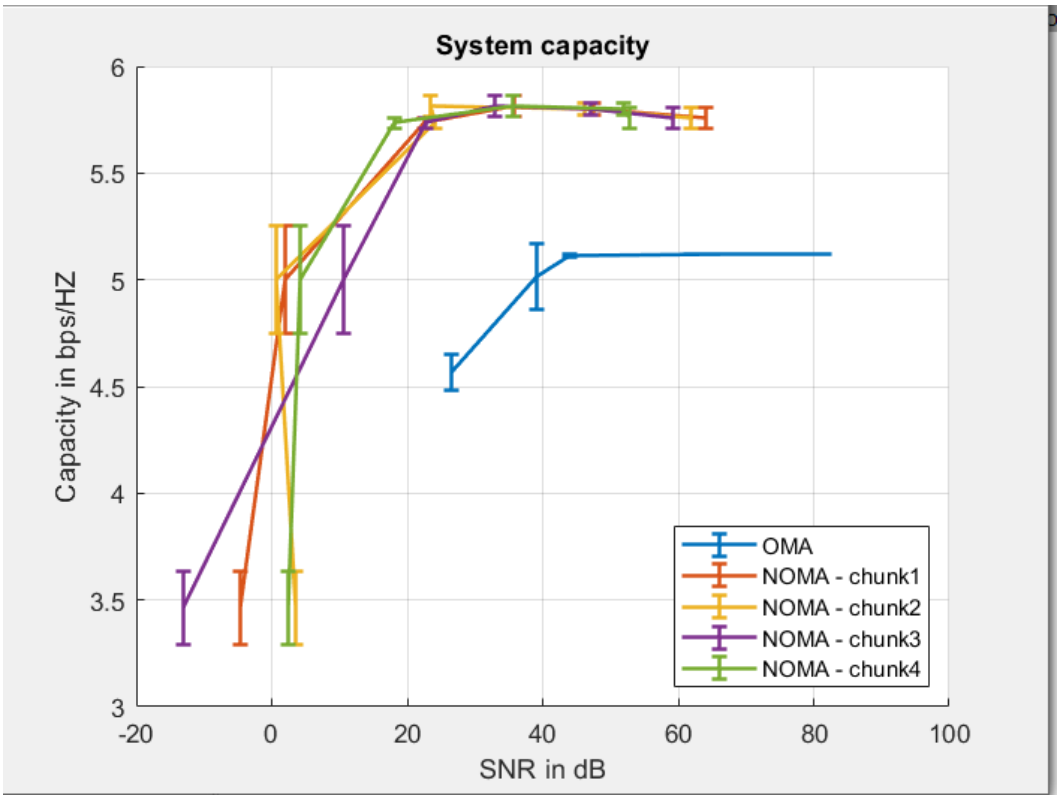
```
x = setPositionWithPathloss(params, freqGHz, 105, alpha);
x = setPositionWithPathloss(params, freqGHz, 100, alpha);
%channelModel=parameters.setting.ChannelModel.Rayleigh;
```

Test#1

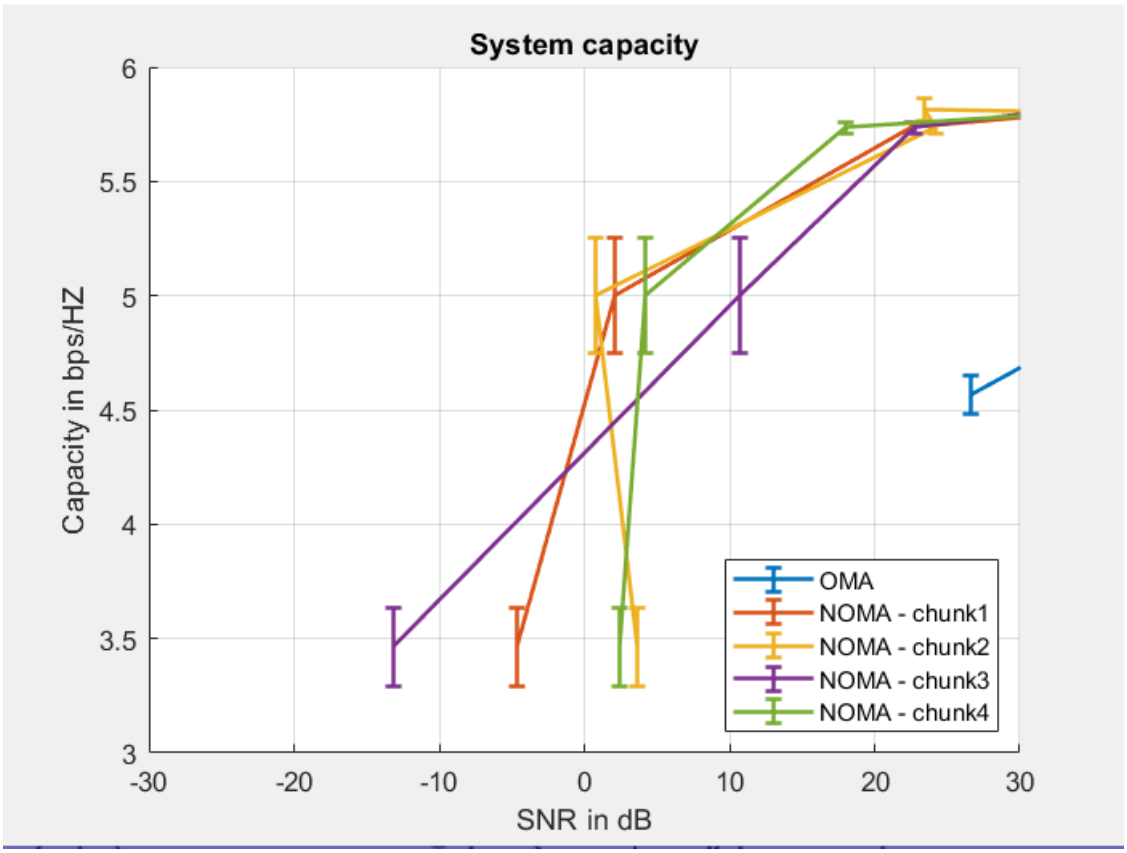
`Scheduler: RoundRobin`



Test#2

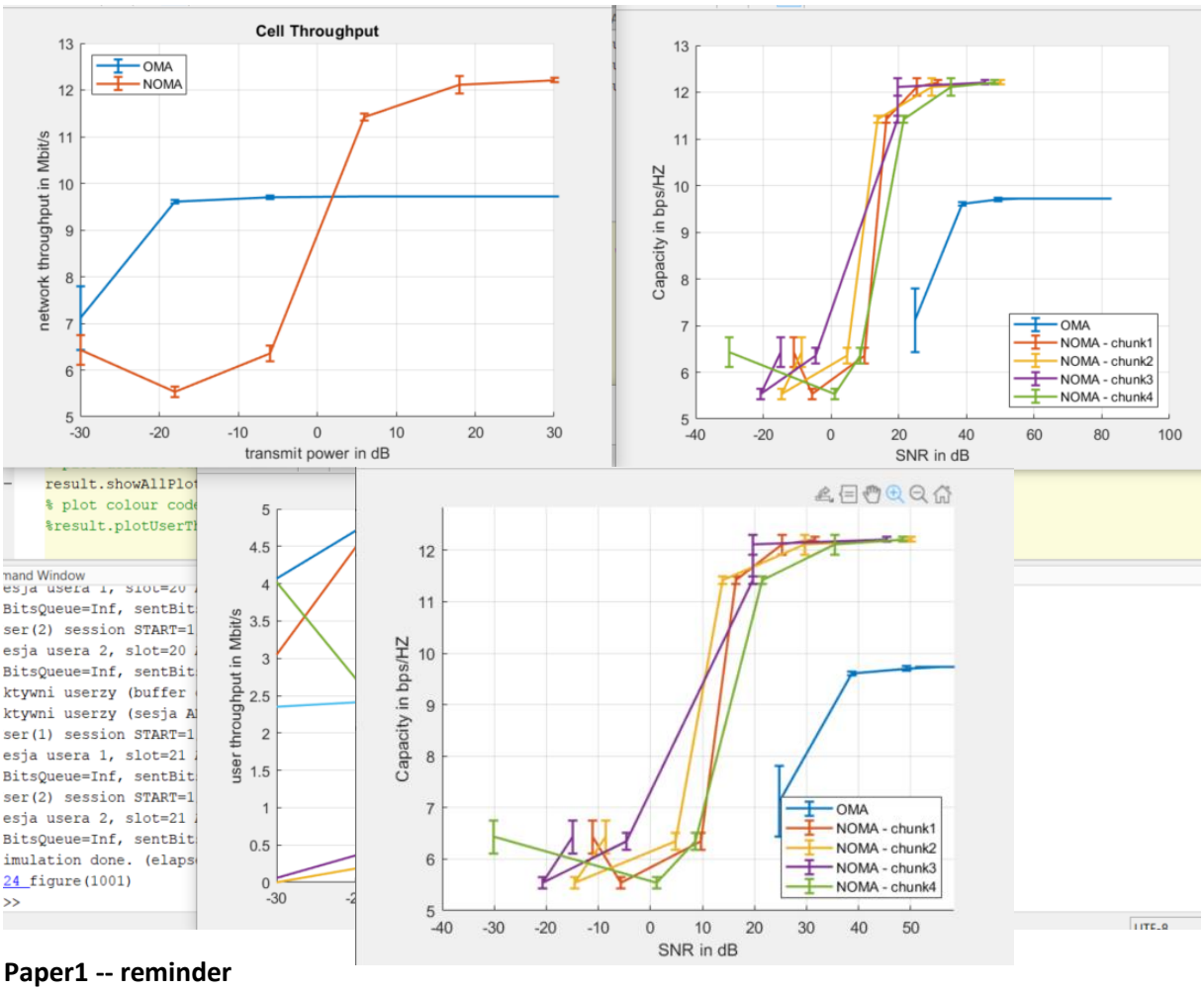


..przesunięcie wykresu do zakresu [-30,30]

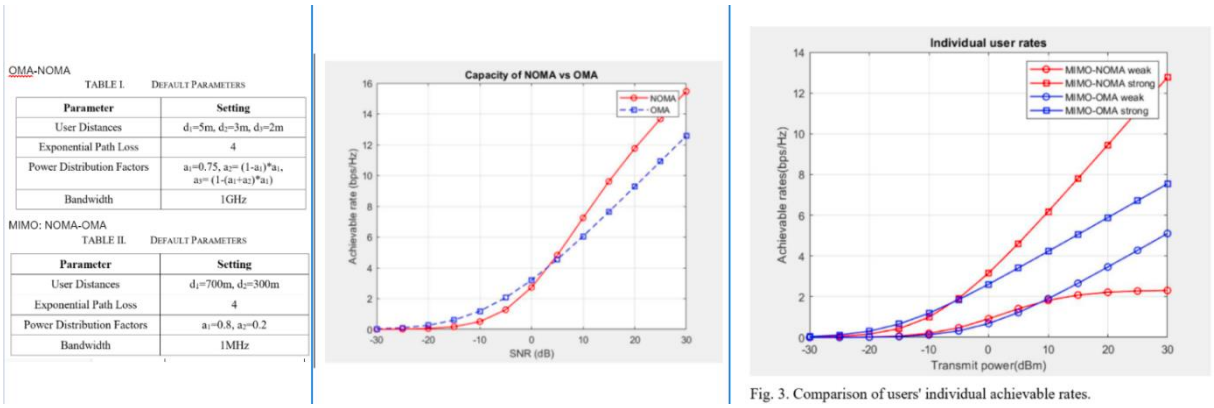


testy MIMO 2x2

```
x = setPositionWithPathloss(params, freqGHz, 120, alpha);
x = setPositionWithPathloss(params, freqGHz, 115, alpha);
channelModel     = parameters.setting.ChannelModel.Rayleigh
;
Scheduler: RoundRobin
```



Paper1 -- reminder



OMA-NOMA

TABLE I. DEFAULT PARAMETERS

Parameter	Setting
User Distances	d_1=5m, d_2=3m, d_3=2m
Exponential Path Loss	4
Power Distribution Factors	a_1=0.75, a_2= (1-a_1)*a_1, a_3= (1-(a_1+a_2)*a_1)
Bandwidth	1GHz

MIMO: NOMA-OMA

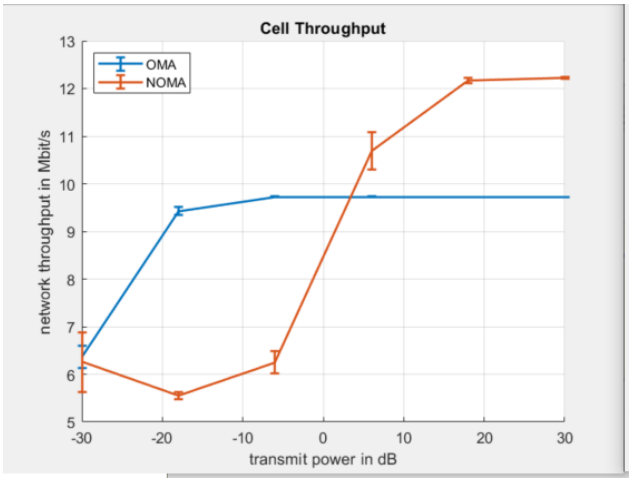
TABLE II. DEFAULT PARAMETERS

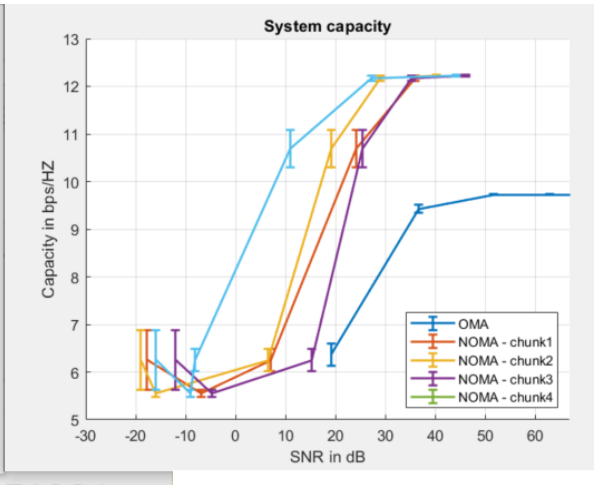
Parameter	Setting
User Distances	d_1=700m, d_2=300m
Exponential Path Loss	4
Power Distribution Factors	a_1=0.8, a_2=0.2
Bandwidth	1MHz

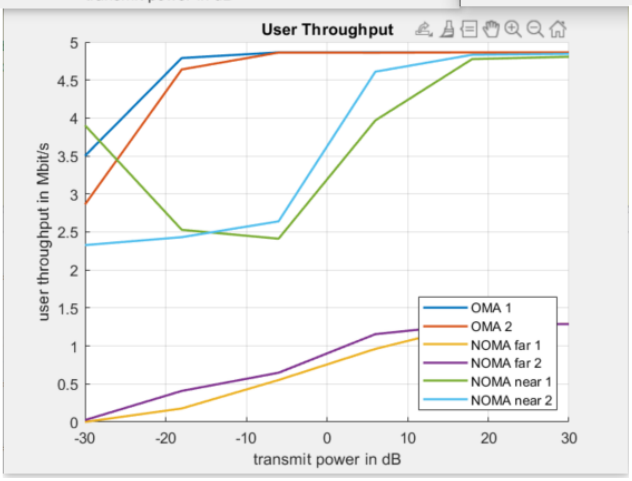
Fig. 3. Comparison of users' individual achievable rates.

testy MIMO 2x2

```
x = setPositionWithPathloss(params, freqGHz, 120, alpha);
x = setPositionWithPathloss(params, freqGHz, 115, alpha);
channelModel     = parameters.setting.ChannelModel.PedA
Scheduler: RoundRobin
;
```

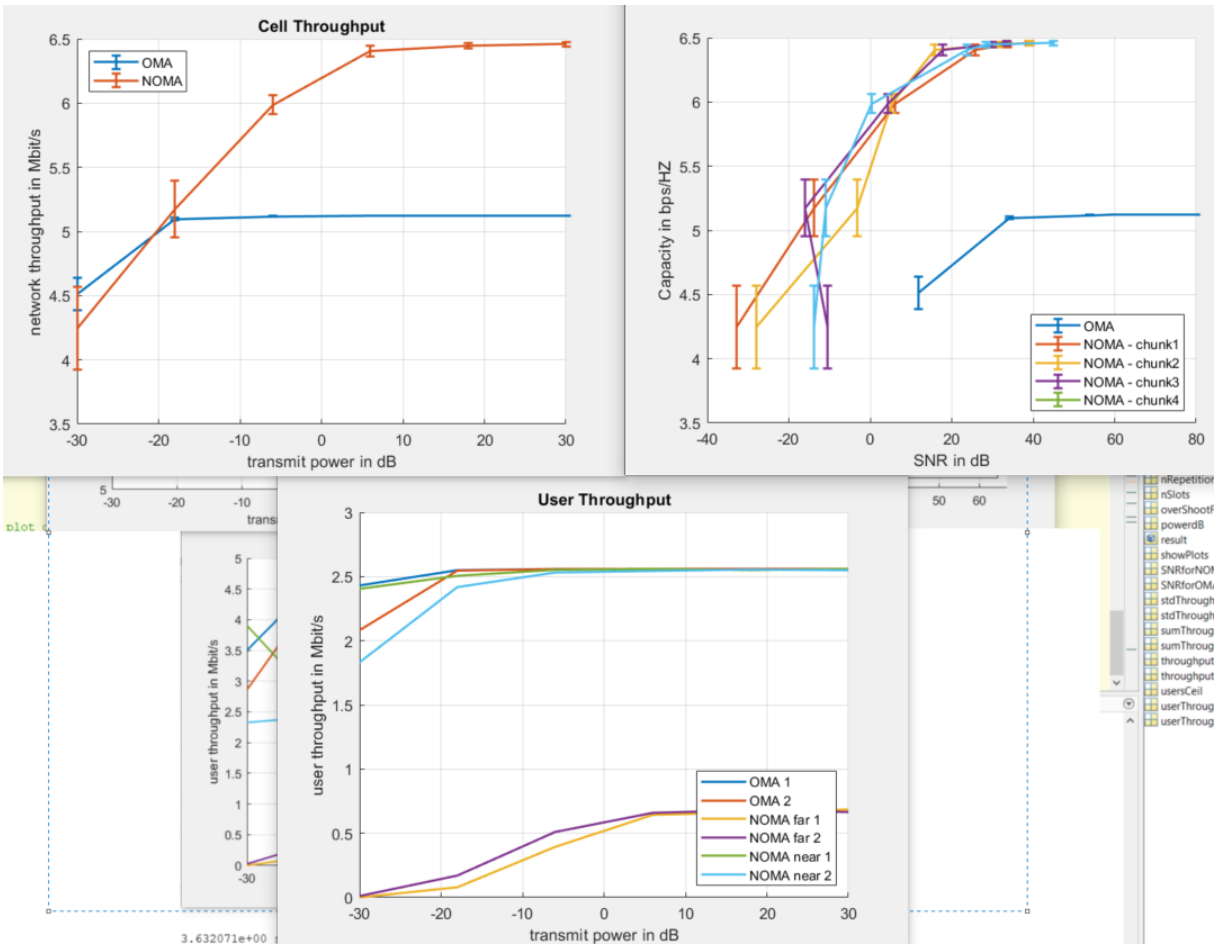






testy SISO

```
x = setPositionWithPathloss(params, freqGHz, 120, alpha);
x = setPositionWithPathloss(params, freqGHz, 115, alpha);
%channelModel=parameters.setting.ChannelModel.PedA;
Scheduler: RoundRobin
```



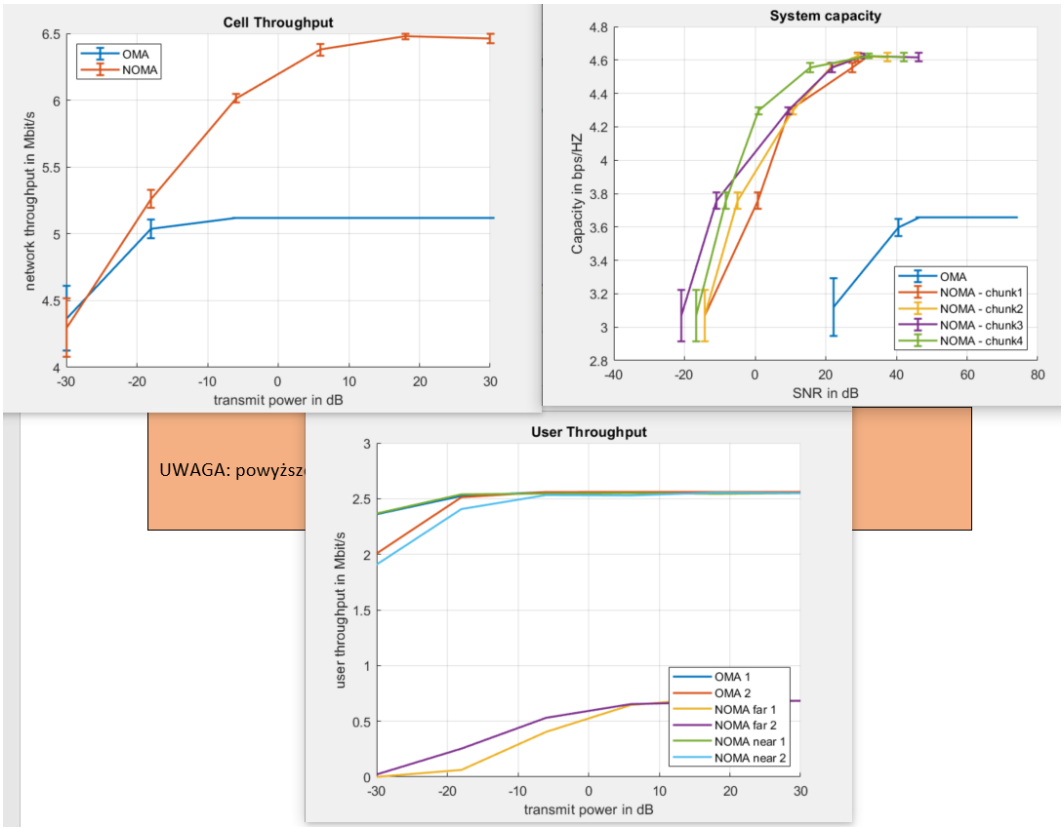
UWAGA: powyższe pomiary i wykresy były rysowane bez normalizacji.

esultsSuperclass.m | launcherNomaMODIFIED.m | launcherFeedback.m | TemporaryResult.m | Postp

```
normalize = 1.4;
% get mean throughput per simulation run

% reshape results
% [nPower x nRepetition x nUser x nSlots] -> [nUser x nPower x nRepetition x n
throughputBestOMA    = permute(throughputBestOMA, [3 1 2 4]);
throughputBestNOMA   = permute(throughputBestNOMA, [3 1 2 4]);

% get mean throughput per simulation run
throughputBestOMA    = mean(throughputBestOMA, 4)/1e6/1e-3;
throughputBestNOMA   = mean(throughputBestNOMA, 4)/1e6/1e-3;

throughputBestOMA_NORMALIZED     = throughputBestOMA / normalize;
throughputBestNOMA_NORMALIZED    = throughputBestNOMA / normalize;

% get mean throughput per user: [nUser x nPower]
userThroughputOMAbest    = mean(throughputBestOMA,3);
userThroughputNOMAbest   = mean(throughputBestNOMA,3);

userThroughputOMAbest_NORMALIZED     = mean(throughputBestOMA_NORMALIZED,3);
userThroughputNOMAbest_NORMALIZED    = mean(throughputBestNOMA_NORMALIZED,3);

% get average cell throughput and statistical values for error bars
```

testy SISO

```
x = setPositionWithPathloss(params, freqGHz, 120, alpha);
x = setPositionWithPathloss(params, freqGHz, 115, alpha);
%channelModel=parameters.setting.ChannelModel.PedA;
```



UWAGA: powyższ

OMA-NOMA

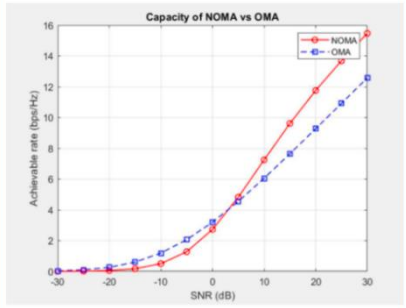
TABLE I. DEFAULT PARAMETERS

Parameter	Setting
User Distances	d_1=5m, d_2=3m, d_3=2m
Exponential Path Loss	4
Power Distribution Factors	a_1=0.75, a_2= (1-a_1)*a_1, a_3= (1-(a_1+a_2)*a_1)
Bandwidth	1GHz

MIMO: NOMA-OMA

TABLE II. DEFAULT PARAMETERS

Parameter	Setting
User Distances	d_1=700m, d_2=300m
Exponential Path Loss	4
Power Distribution Factors	a_1=0.8, a_2=0.2
Bandwidth	1MHz



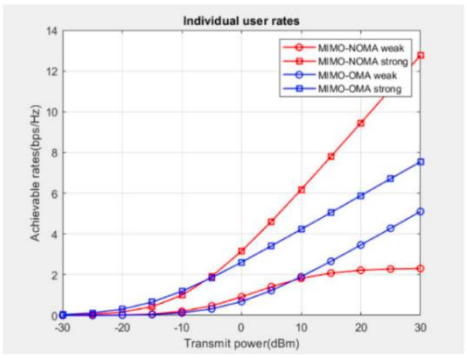
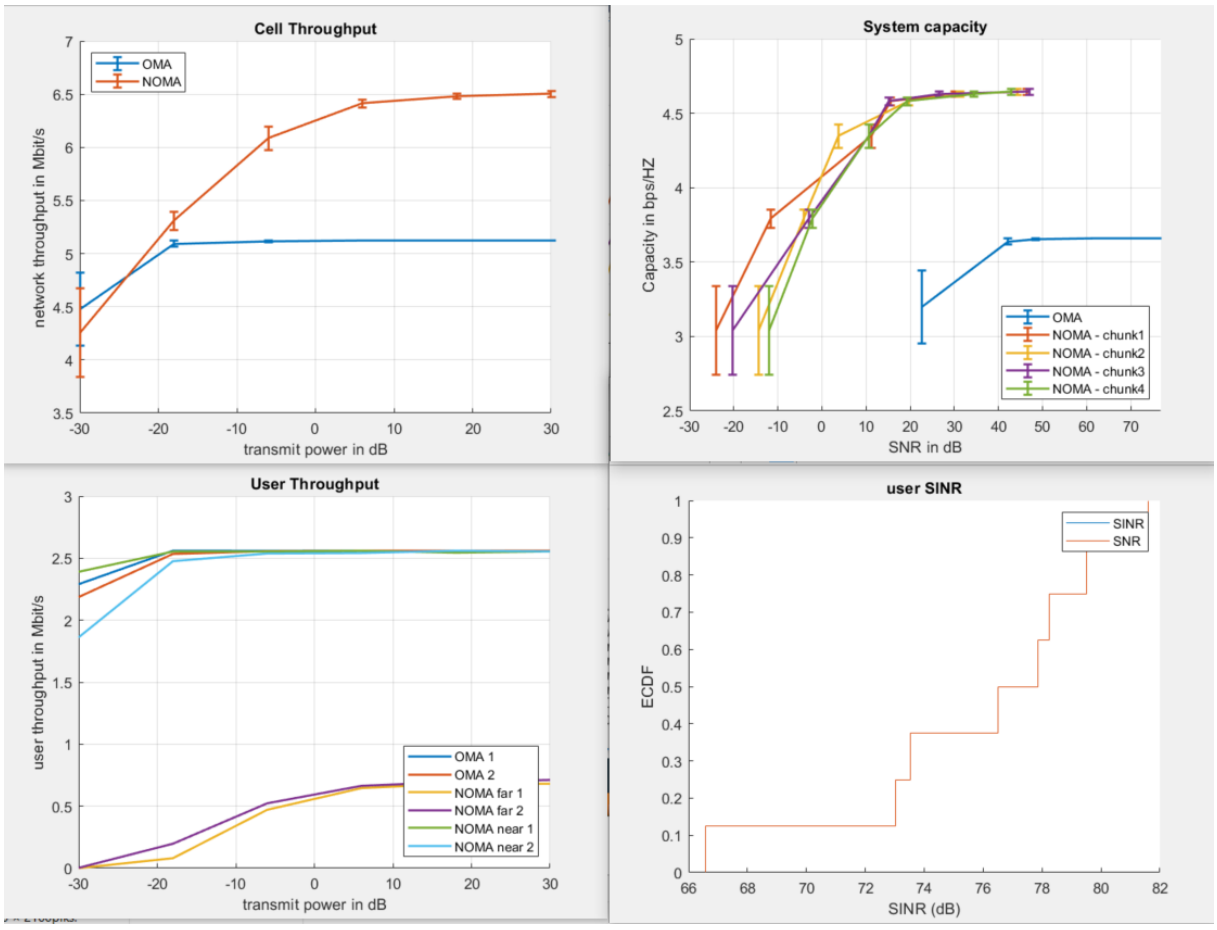
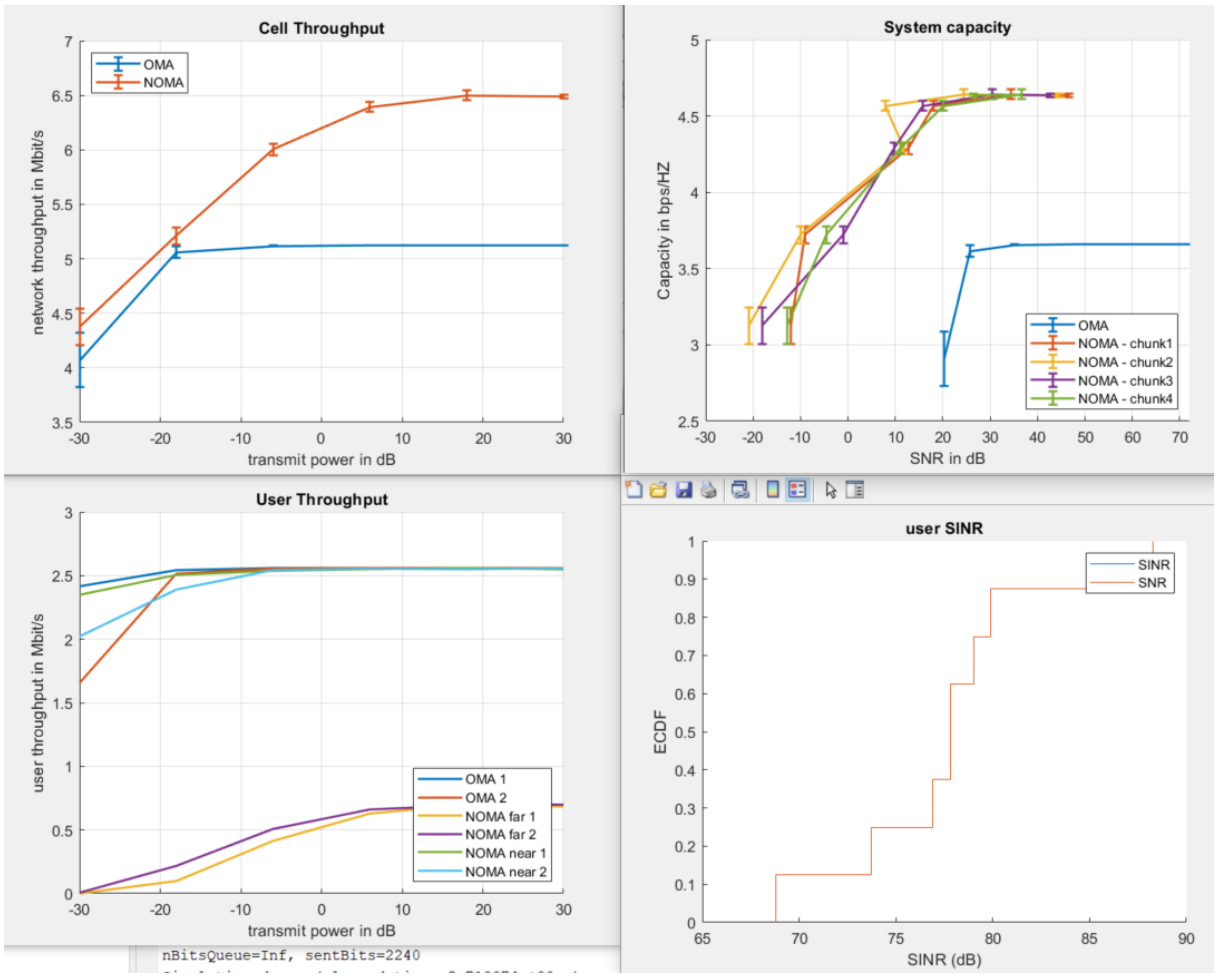


Fig. 3. Comparison of users' individual achievable rates.

testy SISO

```
x = setPositionWithPathloss(params, freqGHz, 120, alpha);
x = setPositionWithPathloss(params, freqGHz, 115, alpha);
%channelModel=parameters.setting.ChannelModel.Rayleigh;
Scheduler: RoundRobin
```





(drugie powtórzenie symulacji powyżej)

Maximum LTE DL throughput:

Bandwidth: 1.4 MHz
Modulation: 64QAM
MIMO: MIMO 2x2

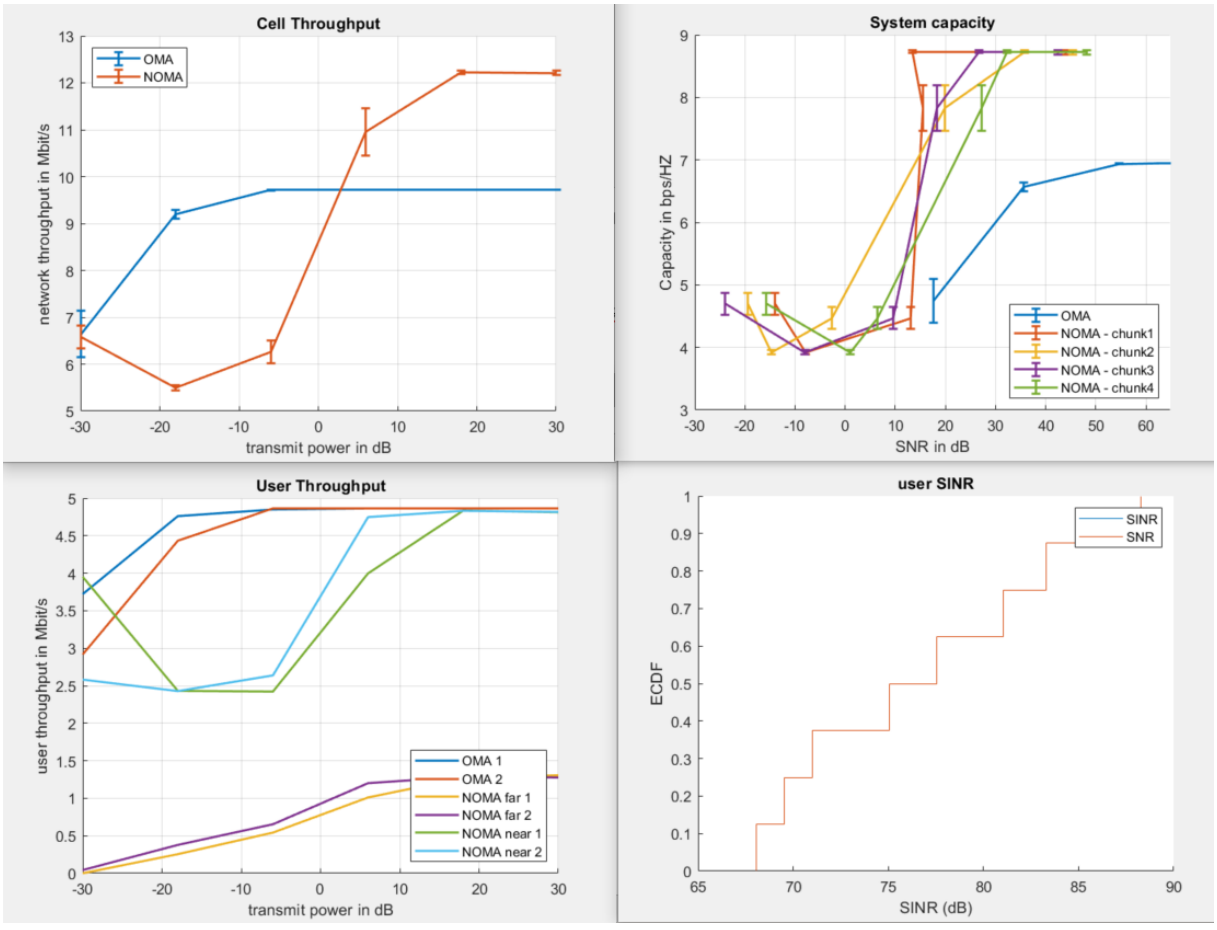
Maximum throughput: 8.784 Mbps

simple4G5G.m | launcherSimulationTime.m | launcherSimple4G5G.m | UserNoma.m | CqiParameters.m | LteCqiParametersTS36213NonBLCEUE1withoutCalibration.m | LteCqiParametersTS36213NonBLCEUE1.m

```
ods
function obj = LteCqiParametersTS36213NonBLCEUE1(transmissionParameters)
    % class constructor - sets CQI table according to TX 36.213 V13.2.0 (2016-06) (i.e., Table 7.2.3-1)
    %
    % input:
    %   transmissionParameters: [1x1]handleObject parameters.transmissionParameters.TransmissionParameters

    % set CQI table according to standard
    cqi                   = [0      1      2      3      4      5      6      7      8      9      10     11     12     13     14     15];
    obj.modulationType    = [1      2      2      2      2      2      2      3      3      3      4      4      4      4      4      4];
    obj.modulationOrder   = [0      2      2      2      2      2      2      4      4      4      6      6      6      6      6      6];
    obj.modulationName    = {'None' 'QPSK' 'QPSK' 'QPSK' 'QPSK' 'QPSK' 'QPSK' 'QPSK' '16QAM' '16QAM' '16QAM' '64QAM' '64QAM' '64QAM' '64QAM' '64QAM' '64QAM'};
    obj.codingRateX1024   = [0      78     120    193    308    449    602    378    490    616    466    567    666    772    873    948];
    obj.efficiency        = [0      0.1523 0.2344 0.3770 0.6016 0.8770 1.1758 1.4766 1.9141 2.4063 2.7305 3.3223 3.9023 4.5234 5.1152 5.5547];
    obj.betaMIESMCalibration = [1   3.07   4.41   0.6    1.16   1.06   1.06   0.87   1.01   1.04   1.03   1.11   1.01   1.07   1      1.05];
    obj.nCqi              = 16;

    obj.blerCurveFiles    = cell(obj.nCqi,1);
    obj.blerCurveFiles{1} = ''; % no bler curve for zero CQI
```

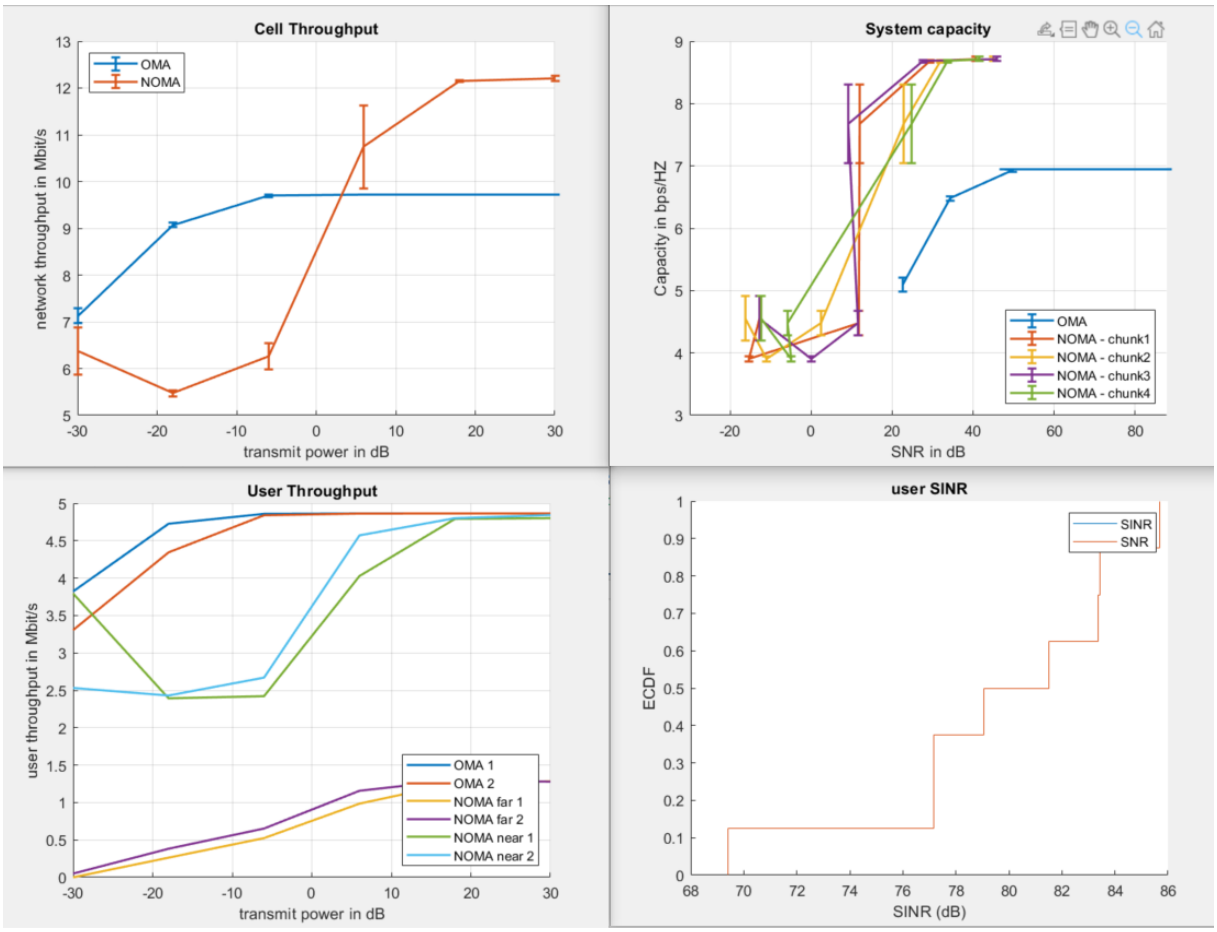
testy MIMO

```
x = setPositionWithPathloss(params, freqGHz, 120, alpha);
x = setPositionWithPathloss(params, freqGHz, 115, alpha);
%channelModel=parameters.setting.ChannelModel.Rayleigh;
Scheduler: RoundRobin
```



testy MIMO

```
x = setPositionWithPathloss(params, freqGHz, 120, alpha);
x = setPositionWithPathloss(params, freqGHz, 115, alpha);
%channelModel=parameters.setting.ChannelModel.PedA;
Scheduler: RoundRobin
```



A może problem że tak się wszystko wypłaszcza polega na modulacji 64QAM? ..zamiast jakiejś wyższej.

Table 7.2.3-1: 4-bit CQI Table

CQI index	modulation	code rate x 1024	efficiency
0	out of range		
1	QPSK	78	0.1523
2	QPSK	120	0.2344
3	QPSK	193	0.3770
4	QPSK	308	0.6016
5	QPSK	449	0.8770
6	QPSK	602	1.1758
7	16QAM	378	1.4766
8	16QAM	490	1.9141
9	16QAM	616	2.4063
10	64QAM	466	2.7305
11	64QAM	567	3.3223
12	64QAM	666	3.9023
13	64QAM	772	4.5234
14	64QAM	873	5.1152
15	64QAM	948	5.5547

Table 5.2.2.1-3: 4-bit CQI Table 2

CQI index	modulation	code rate x 1024	efficiency
0	out of range		
1	QPSK	78	0.1523
2	QPSK	193	0.3770
3	QPSK	449	0.8770
4	16QAM	378	1.4766
5	16QAM	490	1.9141
6	16QAM	616	2.4063
7	64QAM	466	2.7305
8	64QAM	567	3.3223
9	64QAM	666	3.9023
10	64QAM	772	4.5234
11	64QAM	873	5.1152
12	256QAM	711	5.5547
13	256QAM	797	6.2266
14	256QAM	885	6.9141
15	256QAM	948	7.4063

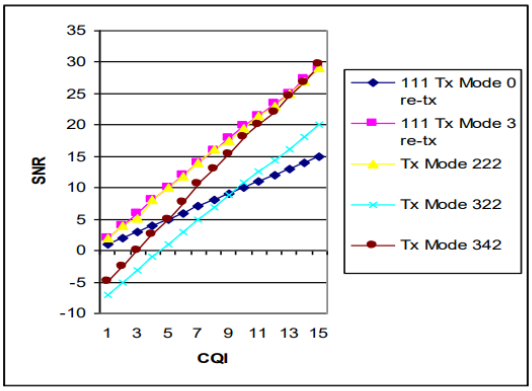


Fig. 6. SNR vs CQI for different Tx modes

TABLE III: OVERALL COMPARISON

CQI	SNR				
	Transmission Mode				
	111, 0 re-tx	111, 3 re-tx	222	322	342
1	1.95	2.00	-7.00	-3.10	-4.80
2	4.00	4.05	-5.00	-1.15	-2.60
3	6.00	5.10	-3.15	1.50	0.00
4	8.00	8.00	-1.00	4.00	2.60
5	10.00	10.00	1.00	6.00	4.95
6	11.95	11.80	3.00	8.90	7.60
7	14.05	13.90	5.00	12.70	10.60
8	16.00	16.10	6.90	14.90	12.95
9	17.90	17.45	8.90	17.50	15.40
10	19.90	19.50	10.85	20.50	18.10
11	21.50	21.50	12.60	22.45	20.05
12	23.45	23.10	14.35	23.20	22.00
13	25.00	24.90	16.15	24.90	24.55
14	27.30	27.00	18.15	27.00	26.80
15	29.00	29.10	20.00	29.10	29.60

Fig. Source [1]

Table III is formed where for each CQI value from 1 to 15 noting SNR values that fulfill the underlying criterion of 10% BLER. The transmission setting 342 implies the use of transmission mode 3 with 4 transmitting antennas and 2 receiving antennas. The transmission settings, 111, 222, 322 and 342 were considered for simulation. In case of transmission mode 3; both 322 and 342 have been considered for simulation using [8]

particular antenna technique. The choice of transmission mode may depend on the instantaneous radio channel conditions and may be adapted semi-statically [6].

1) Transmission Mode 1- Using of a single antenna at eNodeB
2) Transmission Mode 2- Transmit Diversity (TxD)
3) Transmission Mode 3- SU-MIMO Spatial Multiplexing: Open-Loop
4) Transmission Mode 4- SU-MIMO Spatial Multiplexing: Closed-Loop
5) Transmission Mode 5- MU-MIMO Spatial Multiplexing
6) Transmission Mode 6- Beamforming using Closed-Loop Rank-1 Precoding: It can also be seen as a special case of SU-MIMO Spatial Multiplexing.
7) Transmission Mode 7- Beamforming using UE-Specific Reference Signals

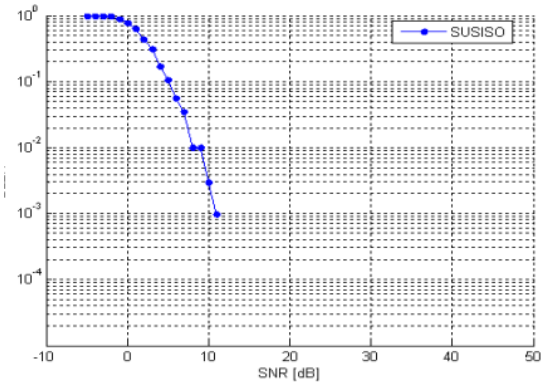


Fig. 3. Transmission mode 2 (222)

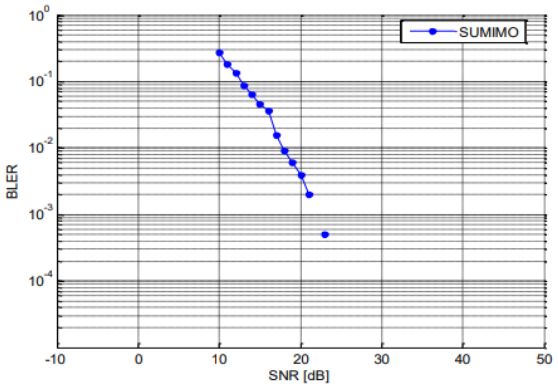


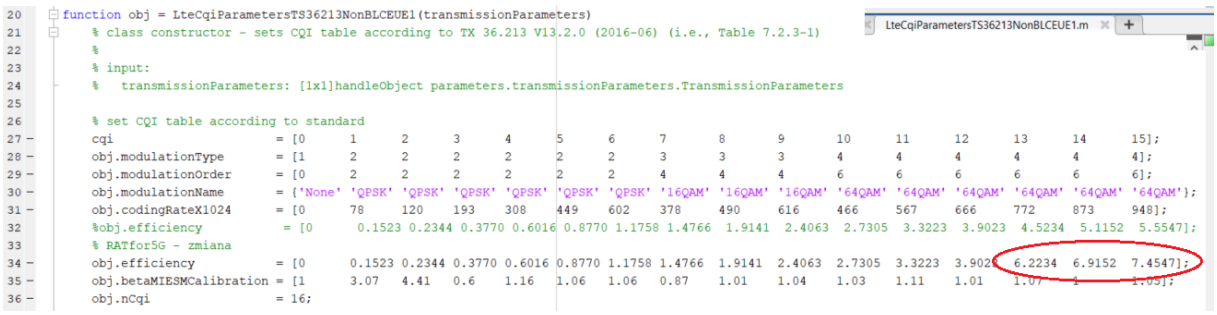
Fig. 4. Transmission mode 3 (322)

Table 2. CQI, MCS and SNR mapping for 3GPP NR.

CQI	MCS	Spectral efficiency	SNR in dB
0	out of range		
1	QPSK, 78/1024	0.15237	-9.478
2	QPSK, 120/1024	0.2344	-6.658
3	QPSK, 193/1024	0.377	-4.098
4	QPSK, 308/1024	0.6016	-1.798
5	QPSK, 449/1024	0.877	0.399
6	QPSK, 602/1024	1.1758	2.424
7	16QAM, 378/1024	1.4766	4.489
8	16QAM, 490/1024	1.9141	6.367
9	16QAM, 616/1024	2.4063	8.456
10	16QAM, 466/1024	2.7305	10.266
11	16QAM, 567/1024	3.3223	12.218
12	16QAM, 666/1024	3.9023	14.122
13	16QAM, 772/1024	4.5234	15.849
14	16QAM, 873/1024	5.1152	17.786
15	16QAM, 948/1024	5.5547	19.809

Source [2]

7. Dlatego robię zmianę typu „brute force” jeśli chodzi o wydajność widmową modulacji dla 256QAM

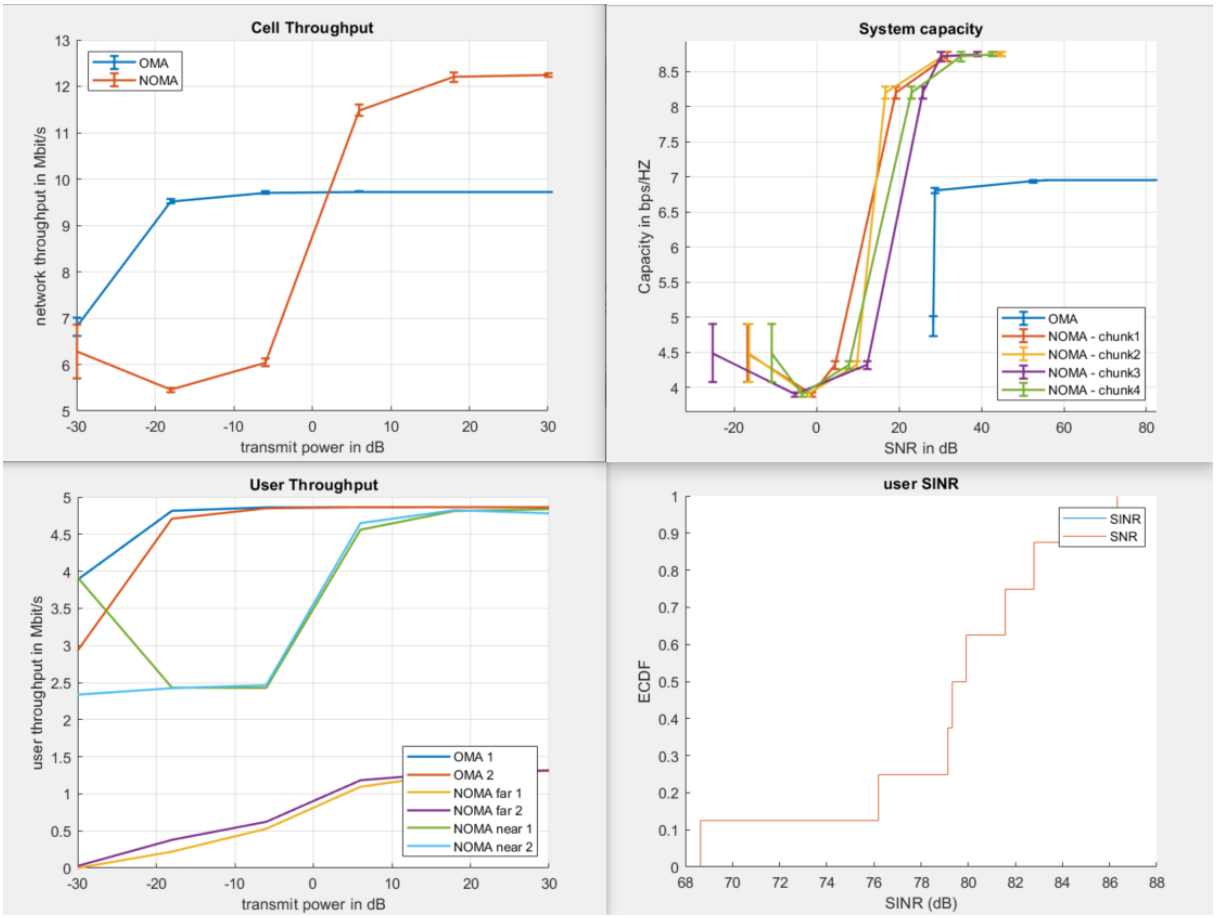


Sprawdzimy czy uda się łatwo obejść obecne ustawiania symulatora… jeśli chodzi o poziom wypłaszczenia na wykresach throughput i capacity.

Bez zmian w stosunku do powyższej symulacji.

testy MIMO

```
x = setPositionWithPathloss(params, freqGHz, 120, alpha);
x = setPositionWithPathloss(params, freqGHz, 115, alpha);
%channelModel=parameters.setting.ChannelModel.PedA;
Scheduler: RoundRobin
```



WNIOSEK: jak widać taka prosta zmiana nie zadziałała.

Funkcje powiązane z kalkulacją przepływności (per RB):

- LinkPerformanceModel.m:
  - **calculateThroughput**(obj, cqi, sinrList, nCodeword, currentTime, assignedRBs, nTX, nLayer)
- Scheduler.m
  - **getTBSizeBits**(assignedRBs, nLayers, nCodewords, iSlot, cqiParameters, resourceGridParameters, layerMapping, nCRCBits, nTX, cqi)

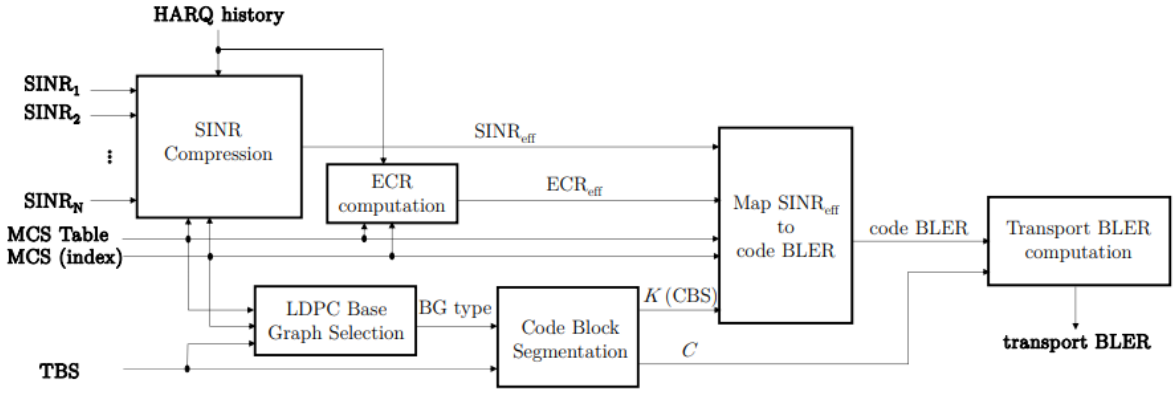


Fig. 1: NR PHY abstraction model.

MIESM model for 5G NR [3]

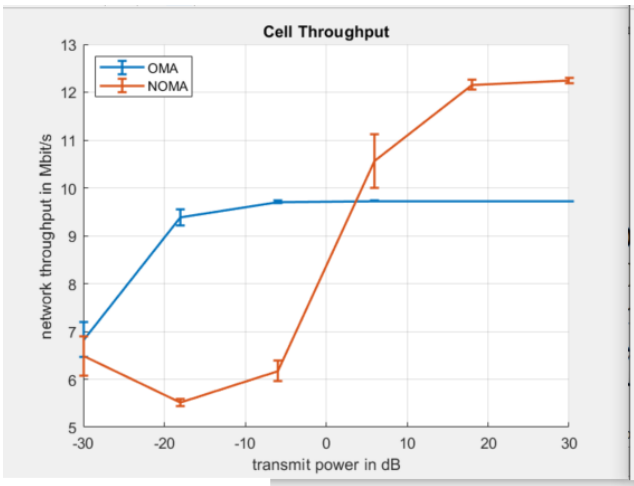
8. Testy dla scenariusza „launcherNomaMODIFIED.m”

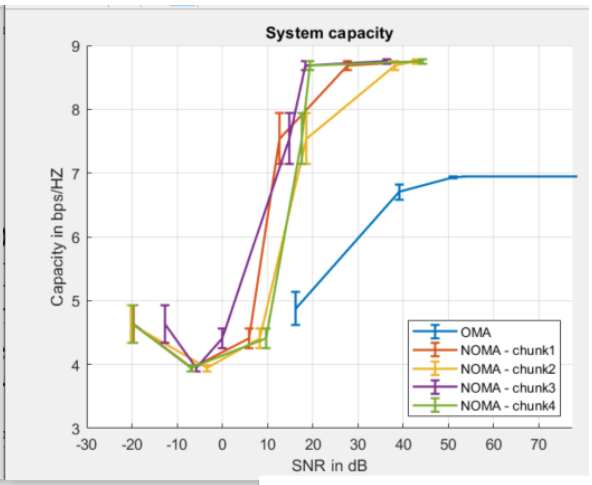
testy MIMO

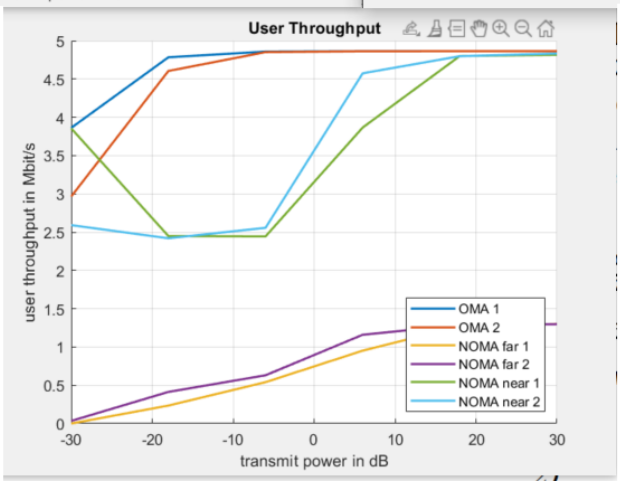
```
x = setPositionWithPathloss(params, freqGHz, 120, alpha);
x = setPositionWithPathloss(params, freqGHz, 115, alpha);
%channelModel=parameters.setting.ChannelModel.PedA;
```
nRepetition = 35;

chunk = 4;

```
Scheduler: RoundRobin
```







testy MIMO

```
x = setPositionWithPathloss(params, freqGHz, 120, alpha);
x = setPositionWithPathloss(params, freqGHz, 115, alpha);
%channelModel=parameters.setting.ChannelModel.PedA;
```
nRepetition = 35;

chunk = 12;

Symulacja została przerwana.

testy SISO

```
x = setPositionWithPathloss(params, freqGHz, 120, alpha);
x = setPositionWithPathloss(params, freqGHz, 115, alpha);
%channelModel=parameters.setting.ChannelModel.PedA;
```

nRepetition = 35;

chunk = 8; ⍰ żeby się szybciej liczyło

Poniższe wykresy wynikają z testowania przebiegu SNR na potrzeby walidacji pomiaru system capacity – **SISO oraz niskie wartości repetition/slots/chunks** żeby było szybciej

UWAGA --- najpierw testowanie SNR

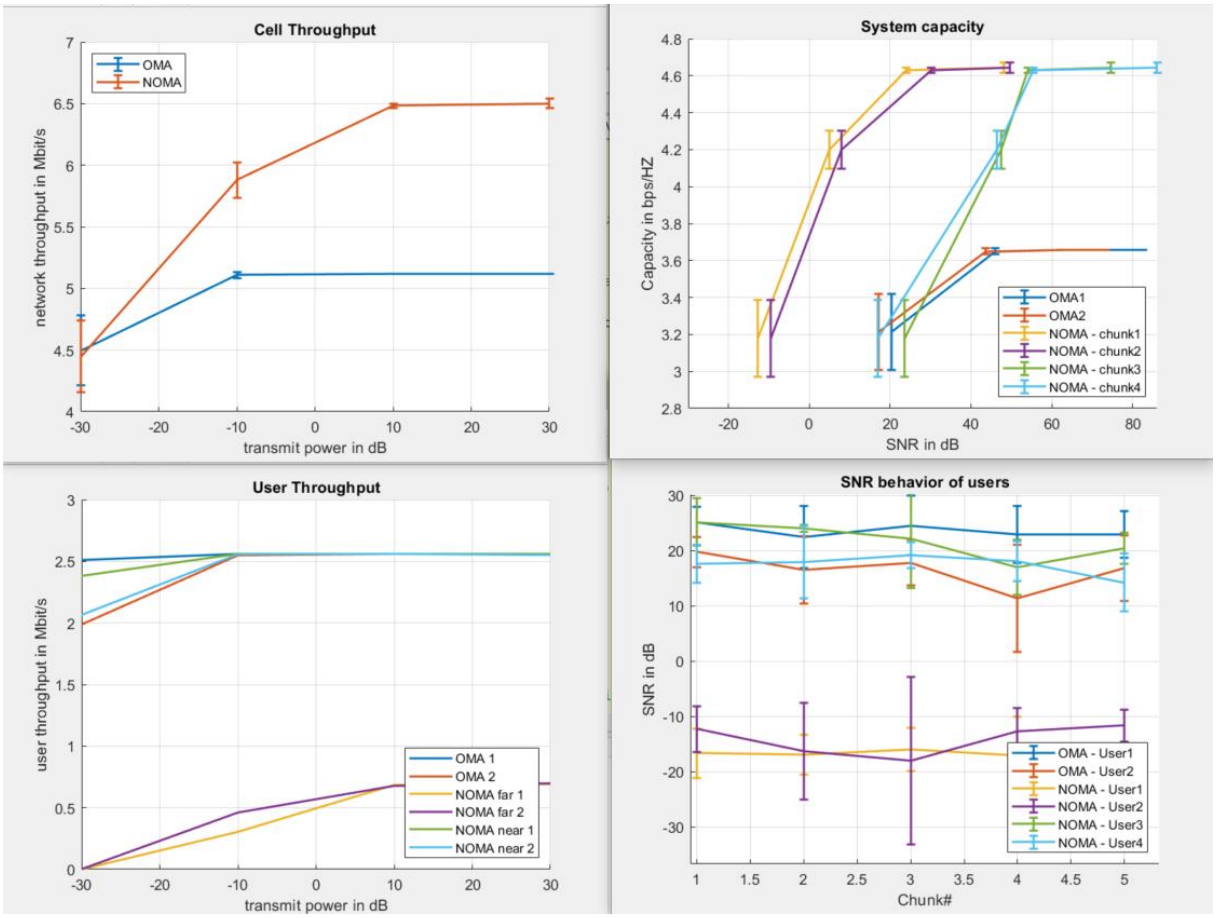
SISO; Scheduler: **RoundRobin**

nPower = 4

Chunks=5

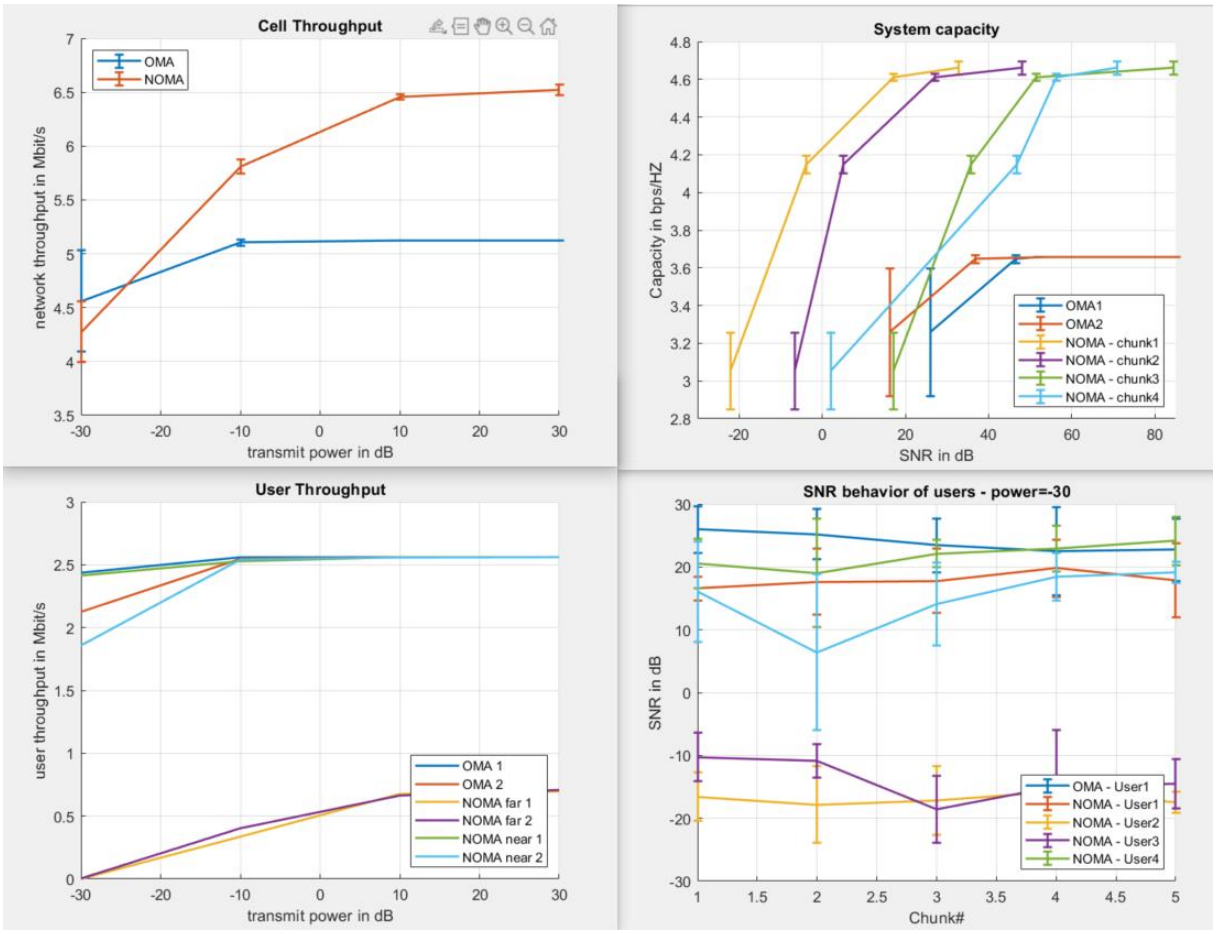
nRepetitions =5 – of course too low!

slots = 10



Powyżej:

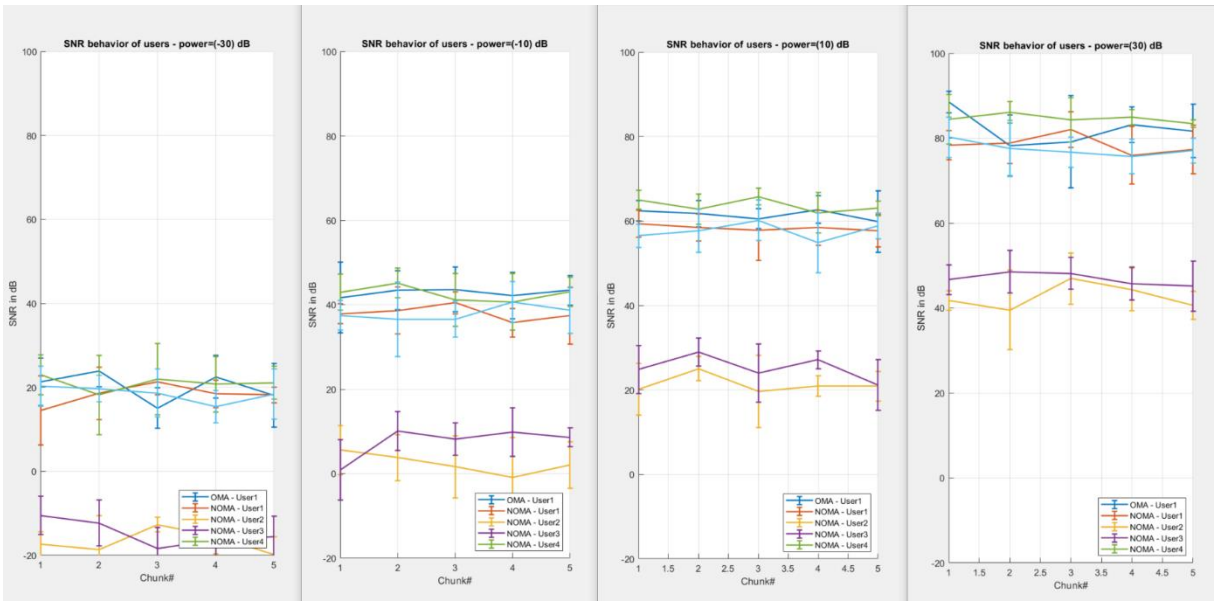
- System capacity
- SNR behaviour of users ---> dla jednego poziomu mocy BS



A teraz testy mocy „per chunk/per user”

SISO (nPower = 4, Chunks=5, nRepetitions =5, slots = 10)

`Scheduler: RoundRobin`



`A rysowane jest to za pomocą takiego kodu:`

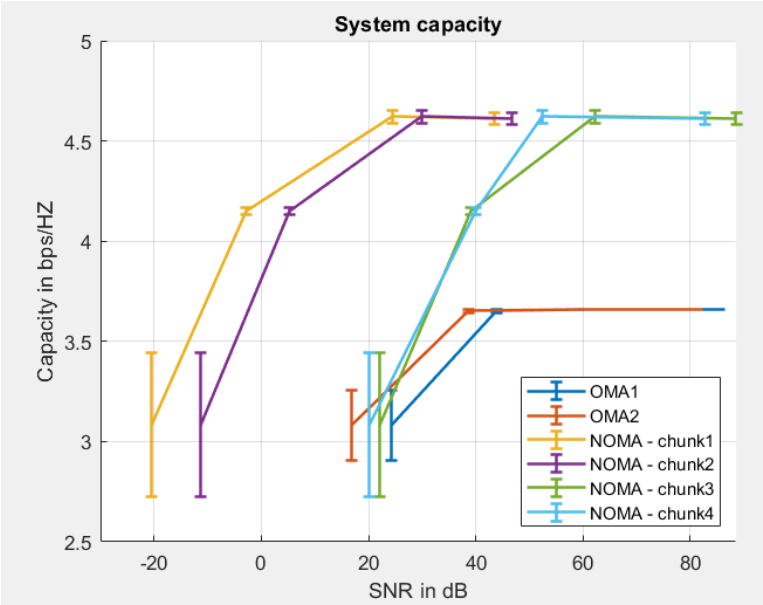
```
function plotSNR(powerIdx, snrOMA, snrNOMA, res, bsPwr)
% plot users SNR statistics (mean, std) per chunk
counter = 1010+powerIdx;
figure(counter)
hold on;
grid on;

xAxis = [1:res.params.time.numberOfChunks]; % chunks number
[bsO,repO,usersO,chunksO] = size(snrOMA(:,:,:,:)); % OMA users
[bsN,repN,usersN,chunksN] = size(snrNOMA(:,:,:,:)); % NOMA users

for i=1:usersO
    % SNRforOMA(:,:,:,:) => [BS x Repetition x User x Chunk]
    % stdSNR_U2OMA  = squeeze(std( SNRforOMA(1,:,2,:),0,2));
    % meanSNR_U4OMA = squeeze(mean(SNRforNOMA(1,:,4,:),2));
    % errorbar(xAxis', meanSNR_U2OMA,  stdSNR_U2OMA, 'LineWidth', 1.5);
    errorbar(xAxis', squeeze(mean(snrOMA(powerIdx,:,i,:),2)), squeeze(std( snrOMA(powerIdx,:,i,:),0,2)), 'LineWidth', 1.5);
end
for i=1:usersN
    %stdSNR_U2OMA  = squeeze(std( SNRforOMA(1,:,2,:),0,2));
    %meanSNR_U4NOMA = squeeze(mean(SNRforNOMA(1,:,4,:),2));
    %errorbar(xAxis', meanSNR_U2NOMA,  stdSNR_U2NOMA, 'LineWidth', 1.5);
    errorbar(xAxis', squeeze(mean(snrNOMA(powerIdx,:,i,:),2)), squeeze(std( snrNOMA(powerIdx,:,i,:),0,2)), 'LineWidth', 1.5);
end
```

launcherNomaMODIFIED2.m

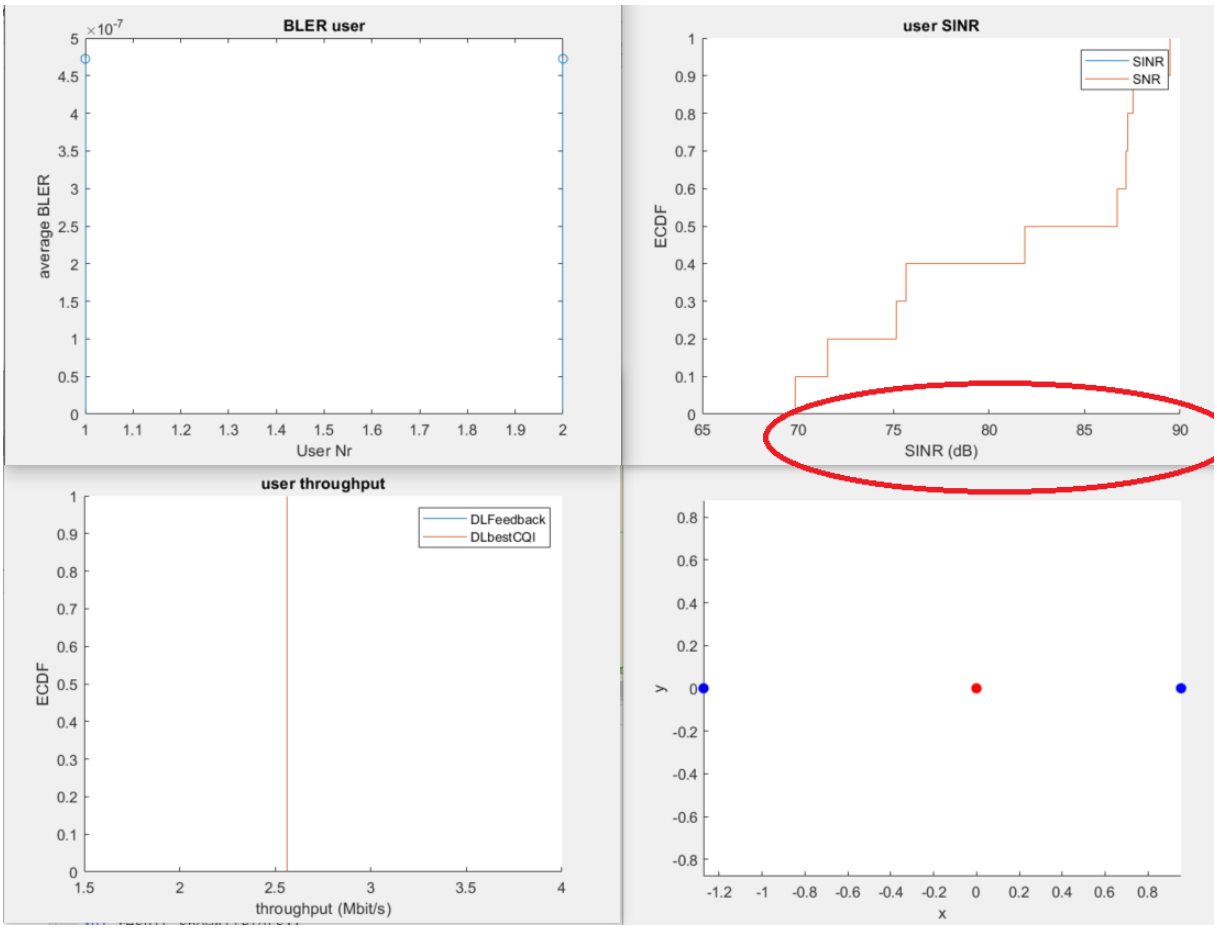
OMA

NOMA

oraz do tego CAPACITY:



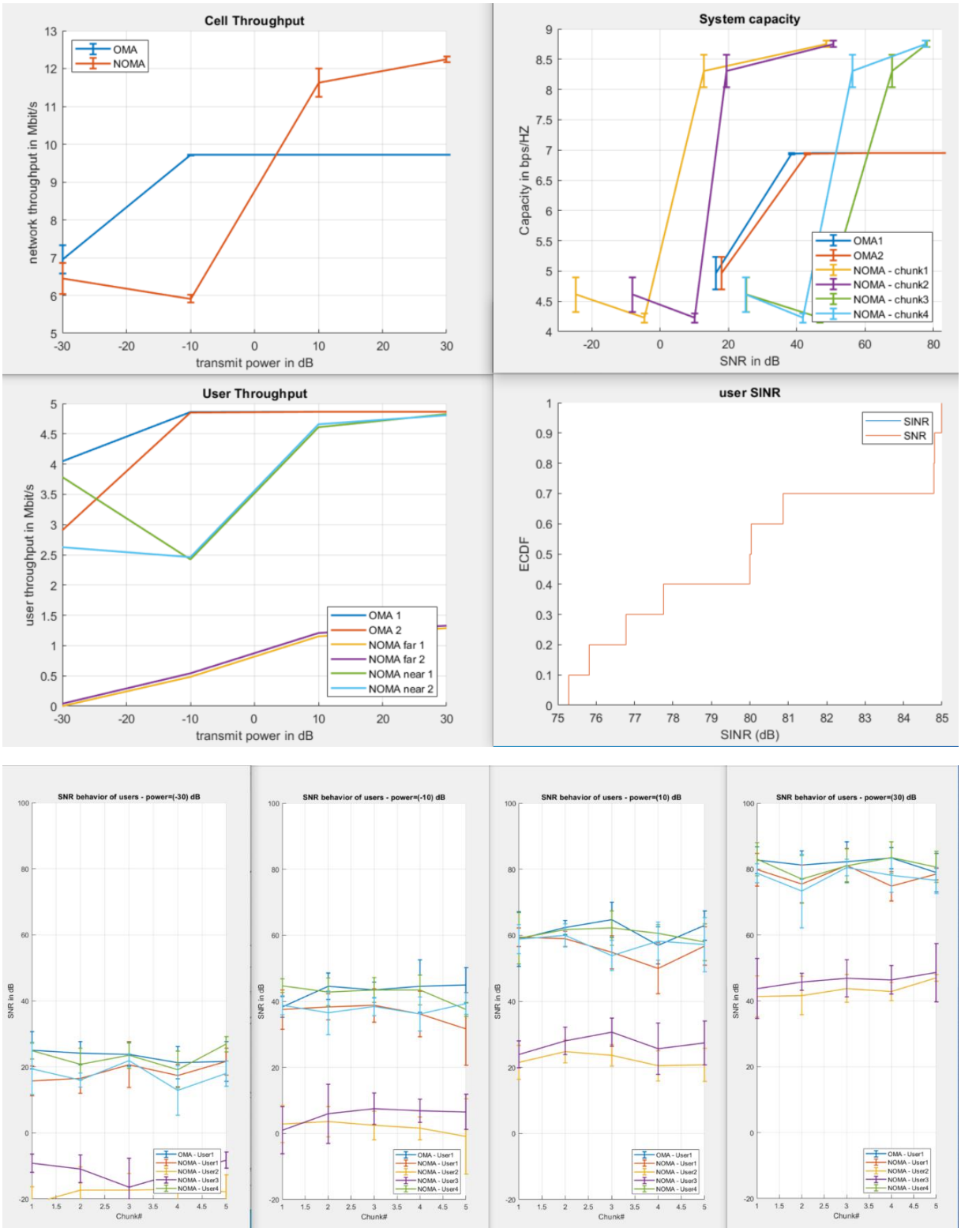
No i zbiorcze wykresy podsumowujące scenariusz (warto zwrócić uwagę na SINR)…


BLER user
×10^-7
average BLER
User Nr
user SINR
SINR
SNR
ECDF
SINR (dB)
user throughput
DLFeedback
DLbestCQI
ECDF
throughput (Mbit/s)
y
x

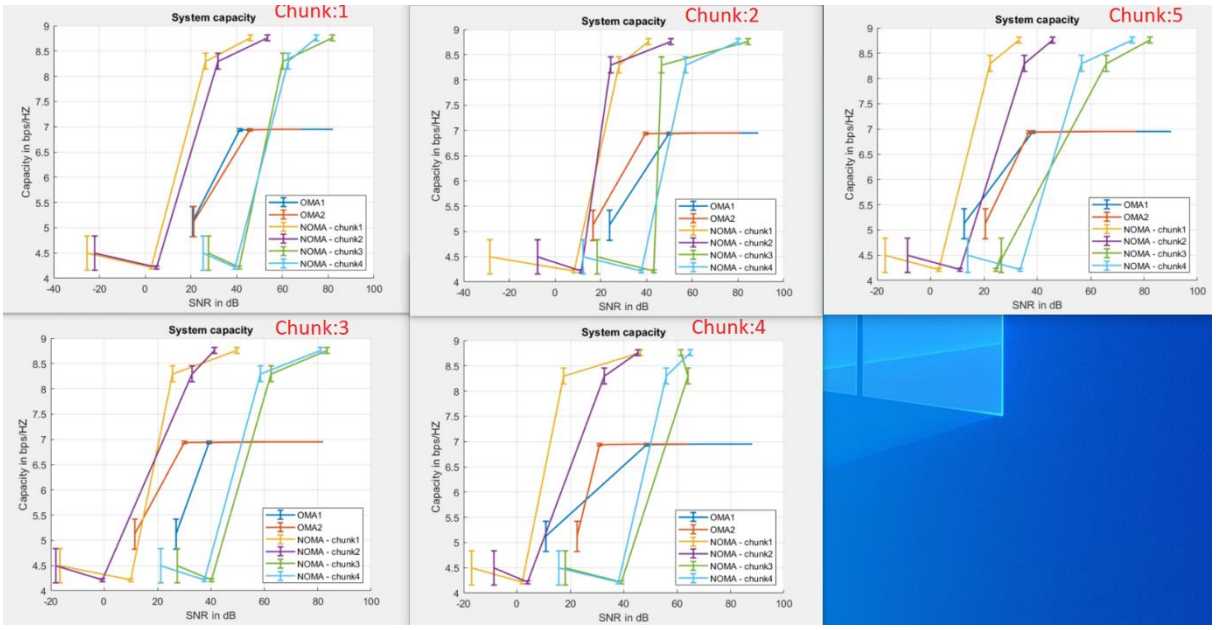
Poniższe wykresy wynikają z testowania przebiegu SNR na potrzeby walidacji pomiaru system capacity – **MIMO oraz niskie wartości repetition/slots/chunks** żeby było szybciej

MIMO (nPower = 4; Chunks=5; nRepetitions =5; slots = 10)

`Scheduler: RoundRobin`



MIMO (nPower = 4; Chunks=5; nRepetitions =5; slots = 10)

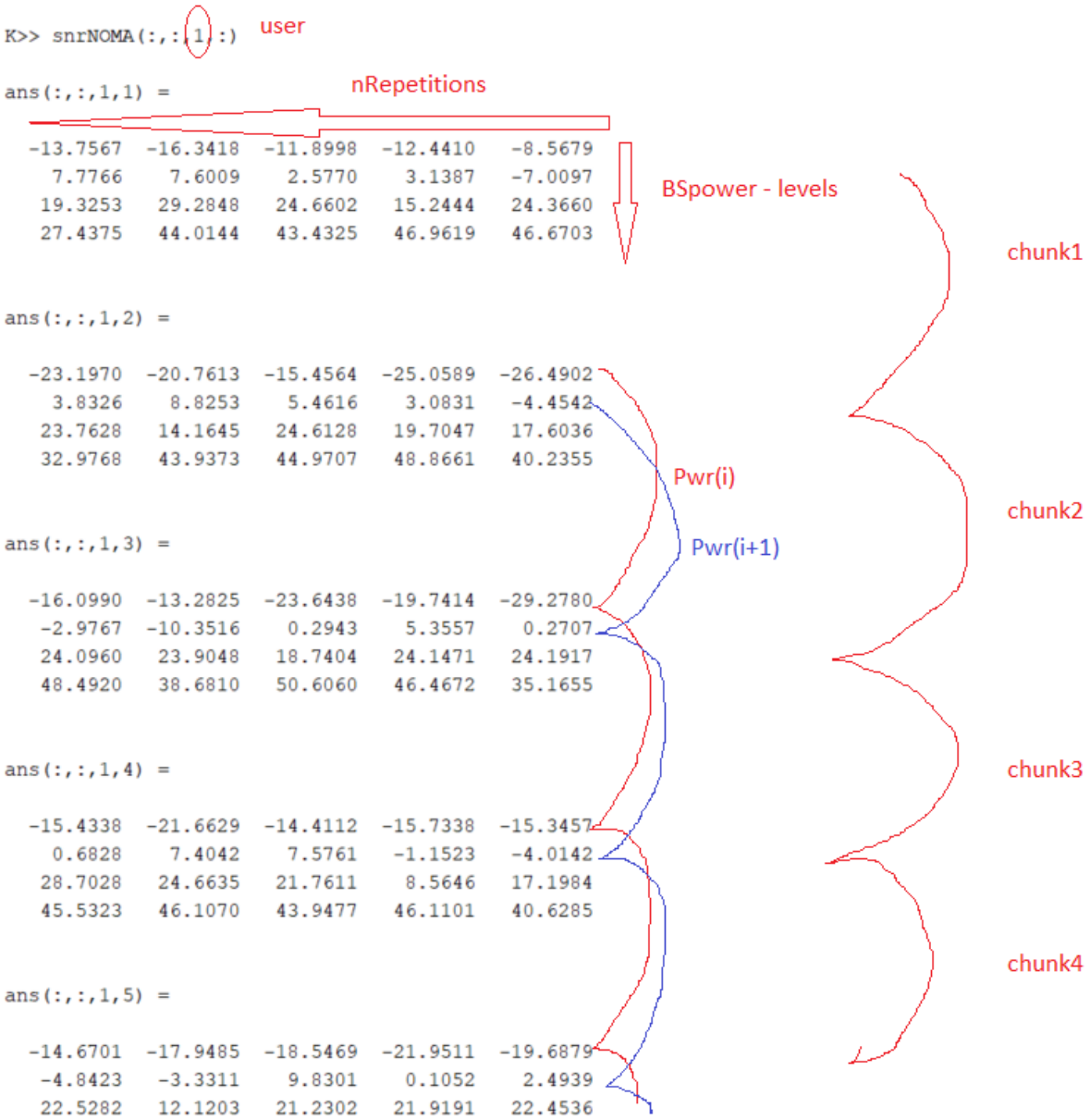


WNIOSEK: celem było sprawdzenie wyników dla różnych chunk'ów. Wygląda na to że wyniki per chunk są zbliżone.

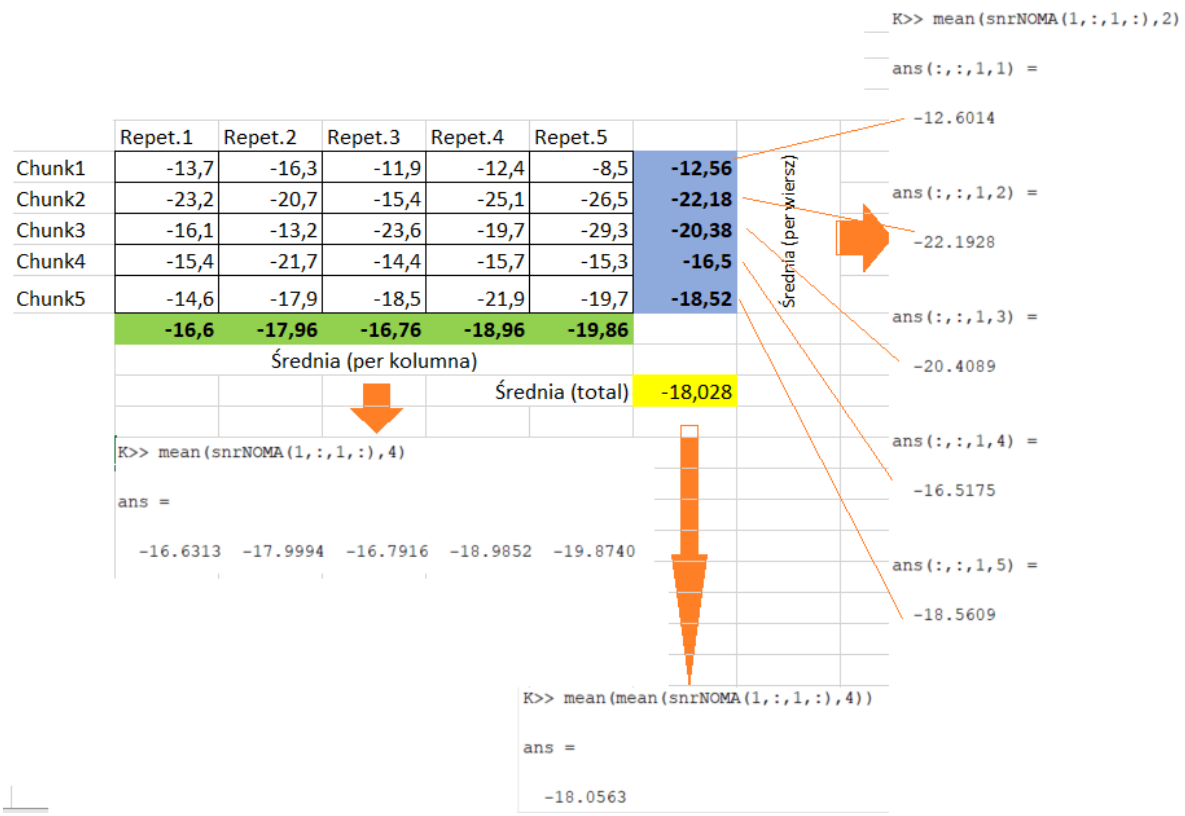
UWAGA teraz działamy na pliku: **launcherNomaMODIFIED2.m**

Analiza danych odnośnie SNR dla NOMA (nSlots =10; nRep=5; BxPower=4, chunk=5;)

Scheduler: RoundRobin



Poniżej wyjaśnienie odnośnie jak należy wyznaczać średnią per user:



Czyli:

1) mean(snr(NOMA(1,:,1,:),4) = średnia po kolumnach (per repetit.)
2) mean(snr(NOMA(1,:,1,:),2) = średnia po wierszach (per chunk)
3) mean(mean(snr(NOMA(1,:,1,:),4)) = całkowita średnia per user

Uwzględniłem Ad.3 w pokazywaniu wyników. Są nadal „per user" ale tym razem uśrednianie jest po wszystkich wymiarach. **Utworzone zostały dwie funkcje:**

- **plotCapacity_vs_SNR_withPower**(bsTxPower, SNRforOMA, SNRforNOMA, mTOB_N, sTOB_N, mTNB_N, sTNB_N, chunk_ii); -- to jest nowa funkcja gdzie uśrednia się per user poszczególne repetitions i chunks i wyznacza power levels w oparciu o poziomy mocy testowanej(muszą być multiple!)
- **plotCapacity_vs_SNR_noPower**(SNRforOMA, SNRforNOMA, mTOB_N, sTOB_N, mTNB_N, sTNB_N, chunk_ii); -- to jest funkcja dotychczasowa, wg której były robione wszystkie wykresy powyżej.

Sposób uśredniania odczytów SNR per user został pokazany poniżej, dla funkcji plotCapacity_vs_SNR_withPower():

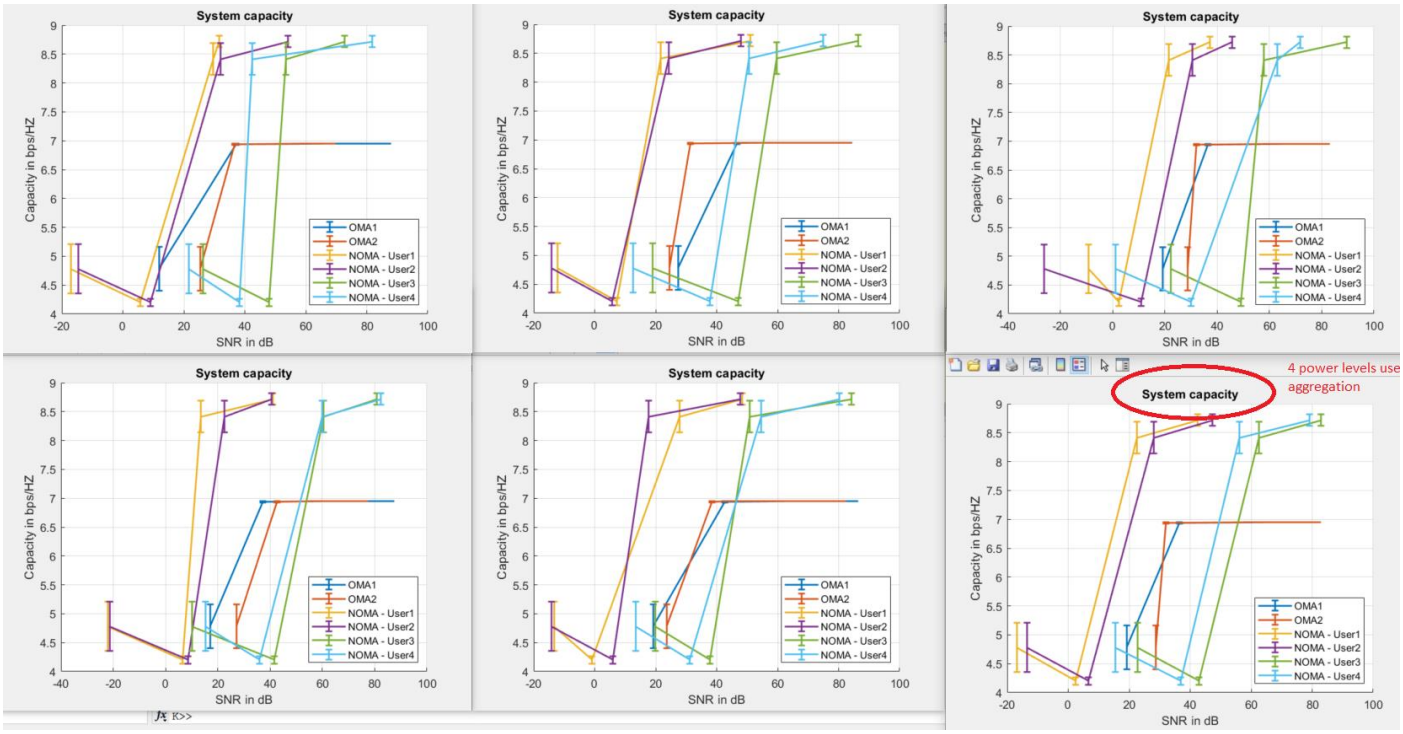
launcherNomaMODIFIED2.m

```
356      %end
357
358      %% TESTING with power
359 -    for u=1:usersN
360 -        avgPerUE = []; % czyść wartości SNR dla usera
361 -        for p=1:numel(powerLevels)
362              % SNRforOMA(:,:,:,:) => [BS x Repetition x User x Chunk]
363              % errorbar(SNRforNOMA(:,1,1,chunk), meanThroughputNOMAbest_NORMALIZED, stdThroughputNOMAbest_NORMALIZED, 'LineWidth', 1.5);
364              % WERSJA wcześniejsza --> errorbar(snrNOMA(:,1,j,chunkToPlot), meanThpNOMA, stdThpNOMA, 'LineWidth', 1.5);
365 -                avgPerUE = [avgPerUE, mean(mean(snrNOMA(p,:,u,:),4))]; % średnia per ALL-repetitions, ALL-chunks
366 -        end
367 -        errorbar(avgPerUE', meanThpNOMA, stdThpNOMA, 'LineWidth', 1.5); % krzywa per jeden user!
368 -    end
369      %%
370 -    legend('OMA1', 'OMA2', 'NOMA - User1', 'NOMA - User2', 'NOMA - User3', 'NOMA - User4', 'Location', 'southeast')
371 -    xlabel('SNR in dB');
372 -    ylabel('Capacity in bps/HZ');
373 -    title('System capacity');
```

W zrzucie kodu powyżej, pokazano linię 365 w której odbywa się sumowanie oraz linię 367 gdzie odbywa się wyrysowanie krzywej.

Analiza danych odnośnie SNR dla NOMA (nSlots =10; nRep=5; BxPower=4, chunk=5;)

Scheduler: RoundRobin

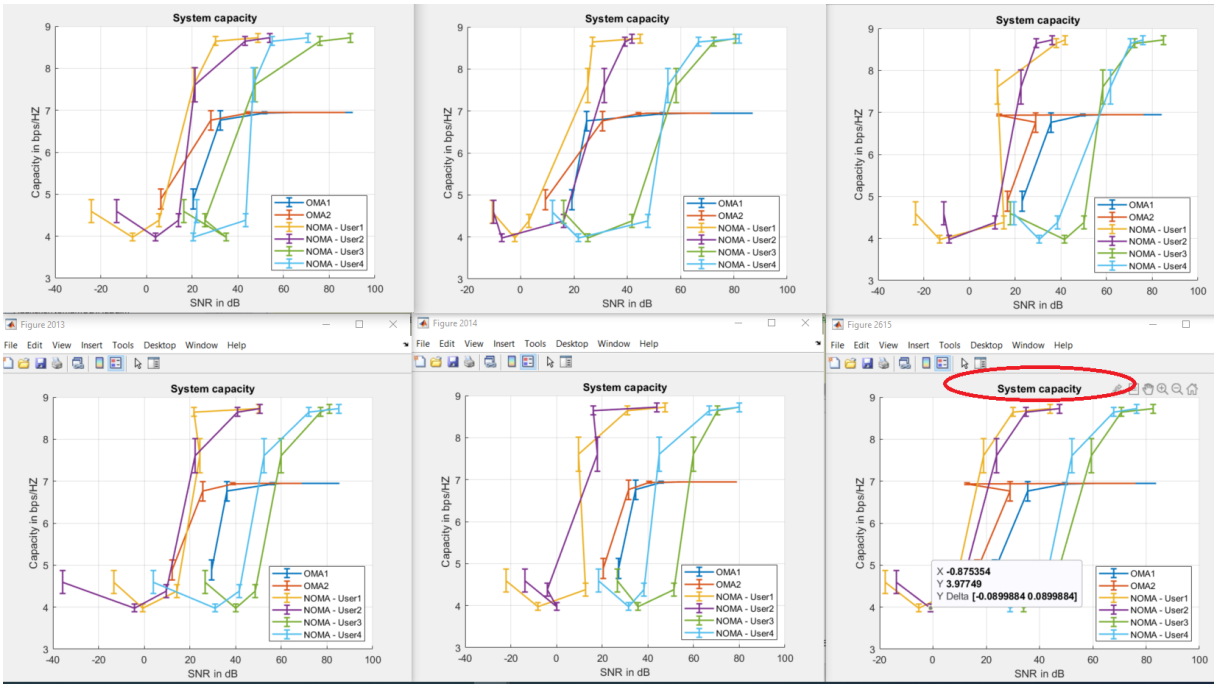


WNIOSEK: wyniki dla ostatniego wykresu (z czerwoną elipsą) są zsumowane. Wykresy poprzedzające (tj wszystkie pozostałe) korzystają z nieprecyzyjnych średnich.

Sprawdzamy dla większej wartości nPower =6 (powyższe są dla „4”)

Analiza danych odnośnie SNR dla NOMA (nSlots =10; nRep=5; BxPower=6, chunk=5;)

Scheduler: RoundRobin



WNIOSKI: jako że wykres dla „ładnego uśredniania” ma dziwne wartości dla OMA (czerwony plot), **ponownie modyfikujemy kod do uśredniania wartości OMA,** na taki sam jak dla NOMA. I puszczamy symulacje TAK SAMO jeszcze raz dla sprawdzenia efektu.

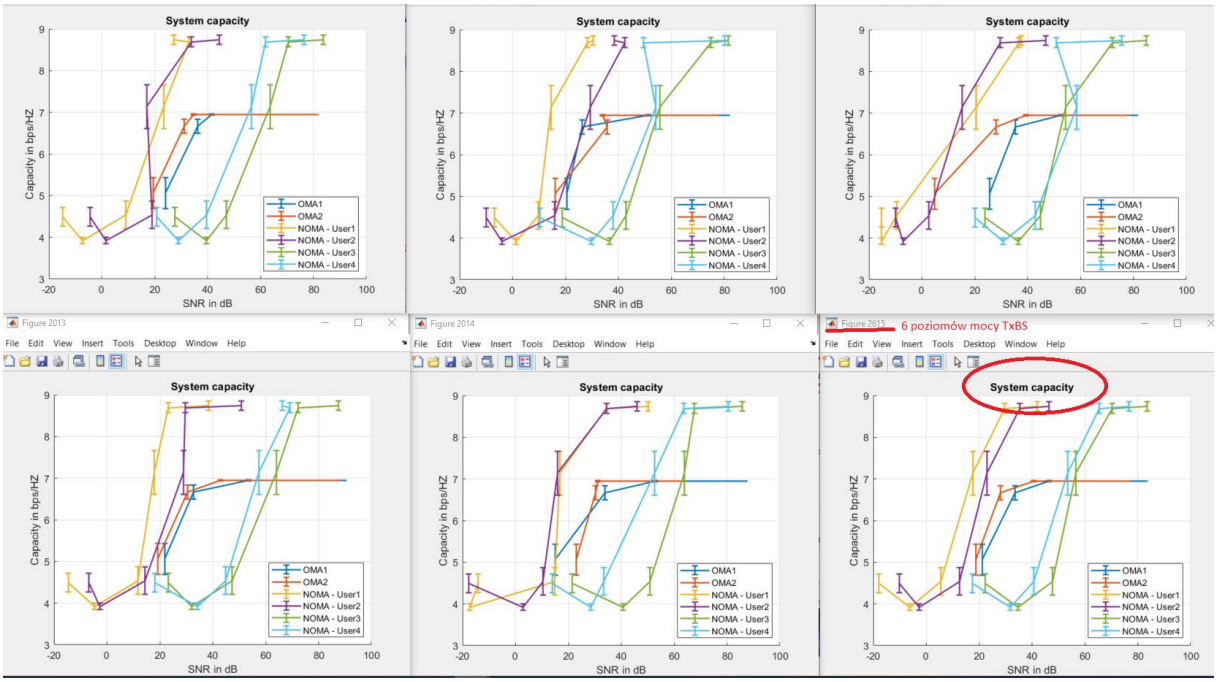
```
335
336     %for i=1:usersO
337         % SNRforOMA(:,:,:,:) => [BS x Repetition x User x Chunk]
338         % errorbar(SNRforOMA(:,1,1,1),  meanThroughputOMAbest_NORMALIZED,  stdThroughputOMAbest_NORMALIZED,  'LineWidth', 1.5);
339         errorbar(snrOMA(:,1,i,chunkToPlot),  meanThpOMA,  stdThpOMA,  'LineWidth', 1.5);
340     %end
341
342     for u=1:usersO
343         avgPerUE = []; % czyść wartości SNR dla usera
344         for p=1:numel(powerLevels)
345         % SNRforOMA(:,:,:,:) => [BS x Repetition x User x Chunk]
346         % errorbar(SNRforOMA(:,1,1,1),  meanThroughputOMAbest_NORMALIZED,  stdThroughputOMAbest_NORMALIZED,  'LineWidth', 1.5);
347             avgPerUE = [avgPerUE, mean(mean(snrOMA(p,:,u,:),4))]; % średnia per ALL-repetitions, ALL-chunks
348         end
349         errorbar(avgPerUE', meanThpOMA, stdThpOMA, 'LineWidth', 1.5); % krzywa per jeden user!
350     end
351
```

launcherNomaMODIFIED2.m

stary

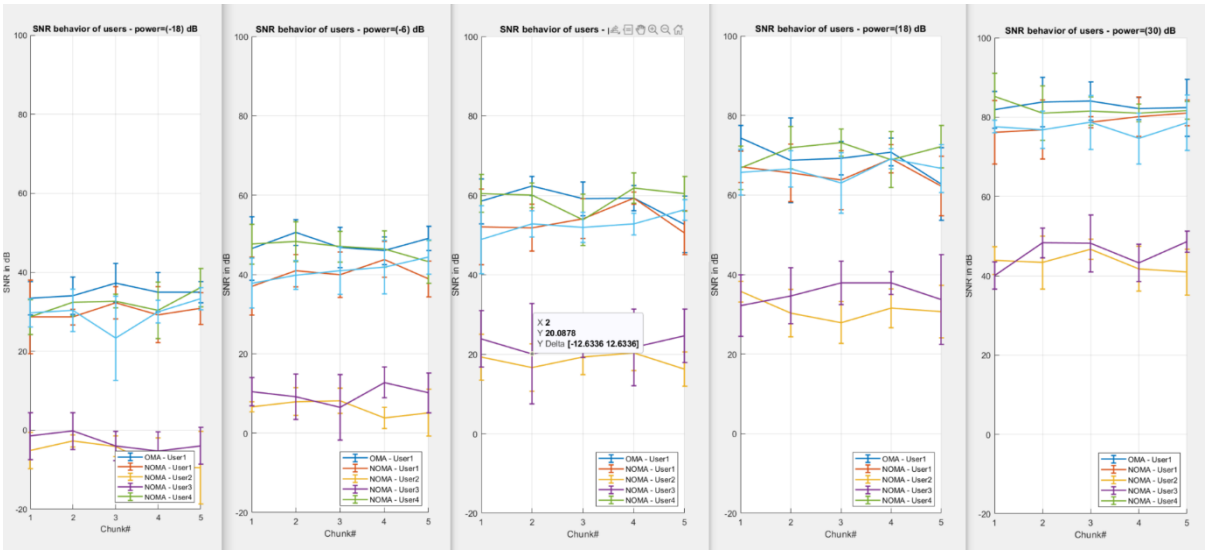
NOWY!!!

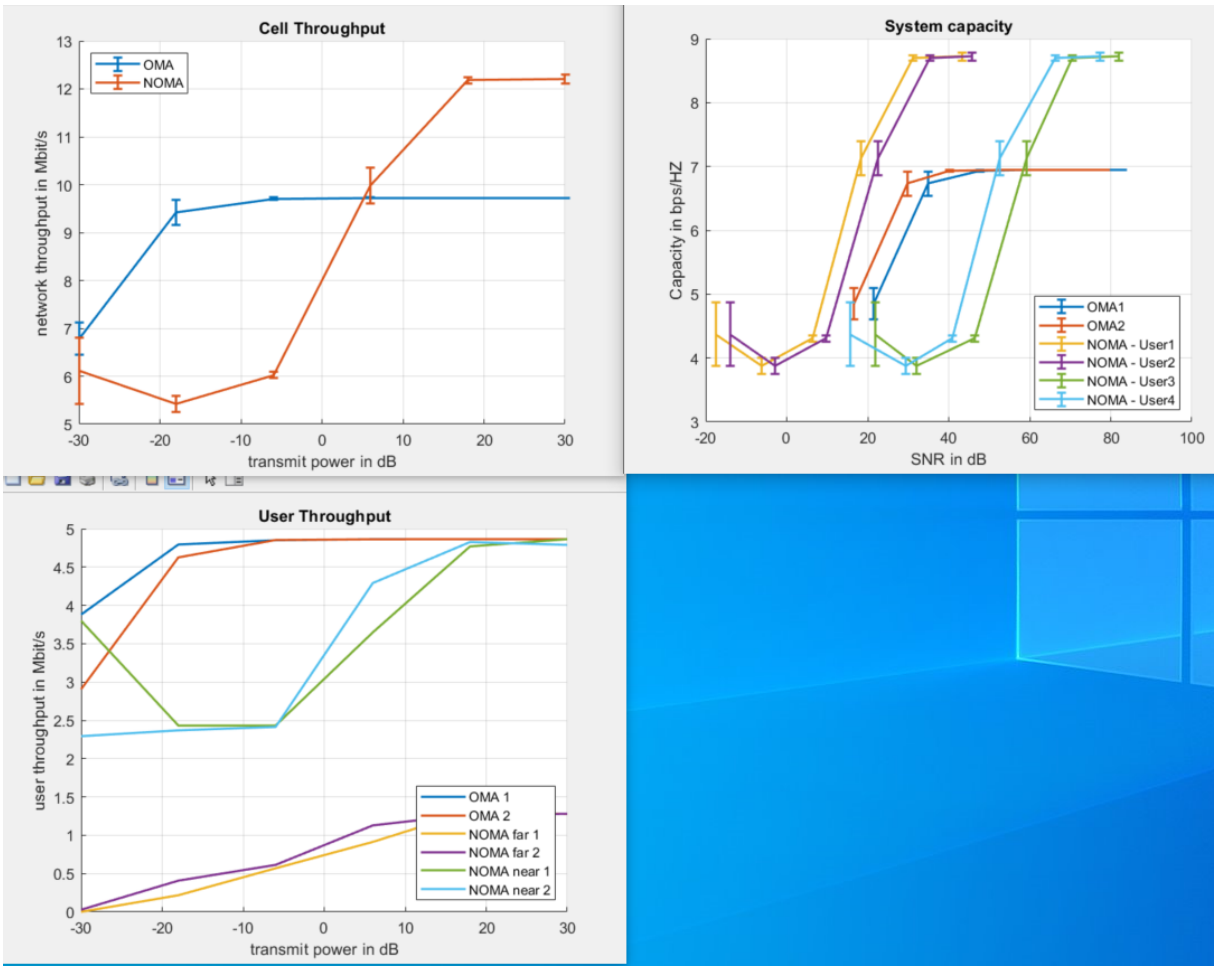
Fig. Środek funkcji **plotCapacity_vs_SNR_withPower()** dla OMA userów



WNIOSEK: wygląda na to, że „dziwne” zachowania wykresów się udało naprawić.

Kolejne powtórzenie powyższej symulacji żeby sprawdzić wykresy „SNR per chunk”

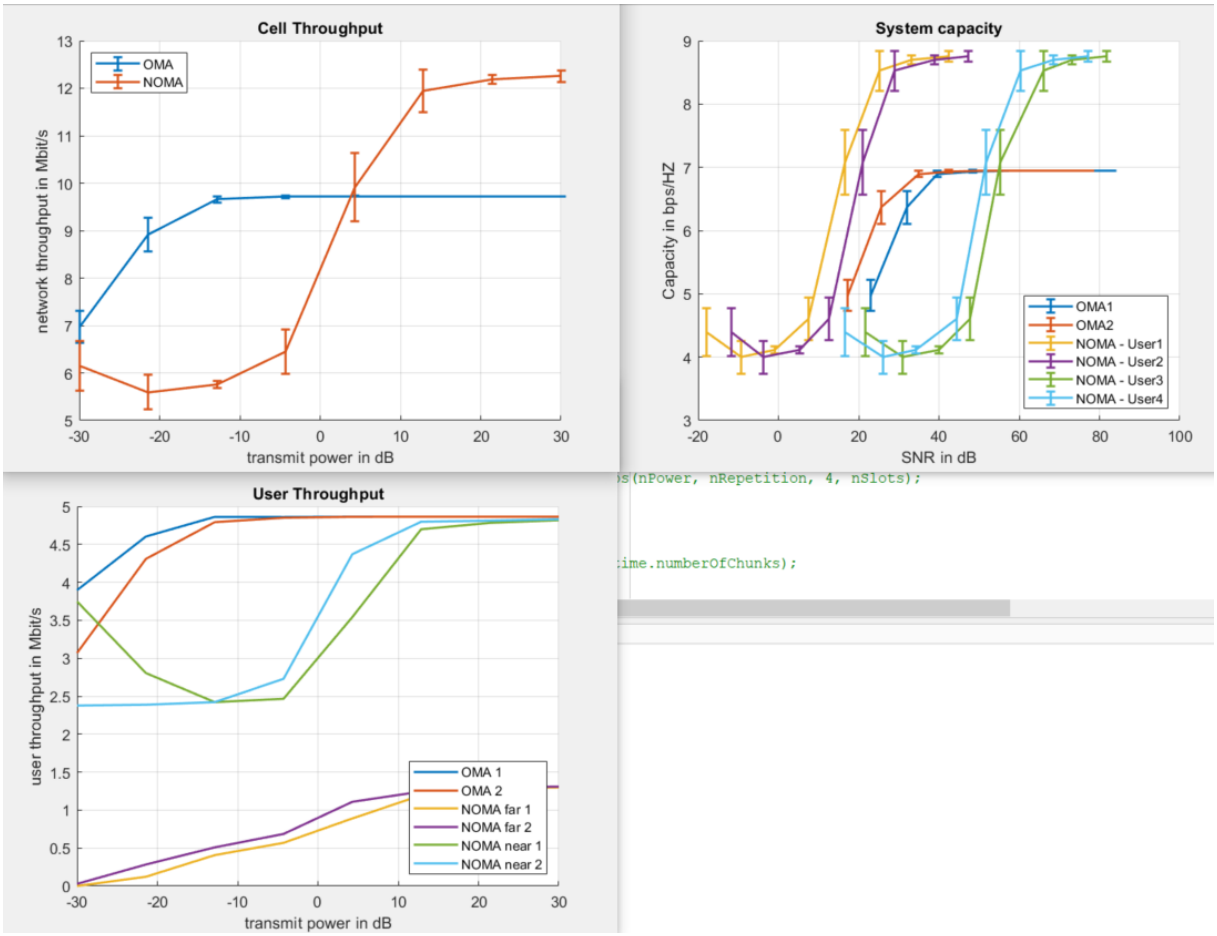




A teraz nPower = 8, nRepetition = 15;

Analiza danych odnośnie SNR dla NOMA (**MIMO**, nSlots =10; nRep=15; BxPower=8, chunk=5;)

Scheduler: RoundRobin



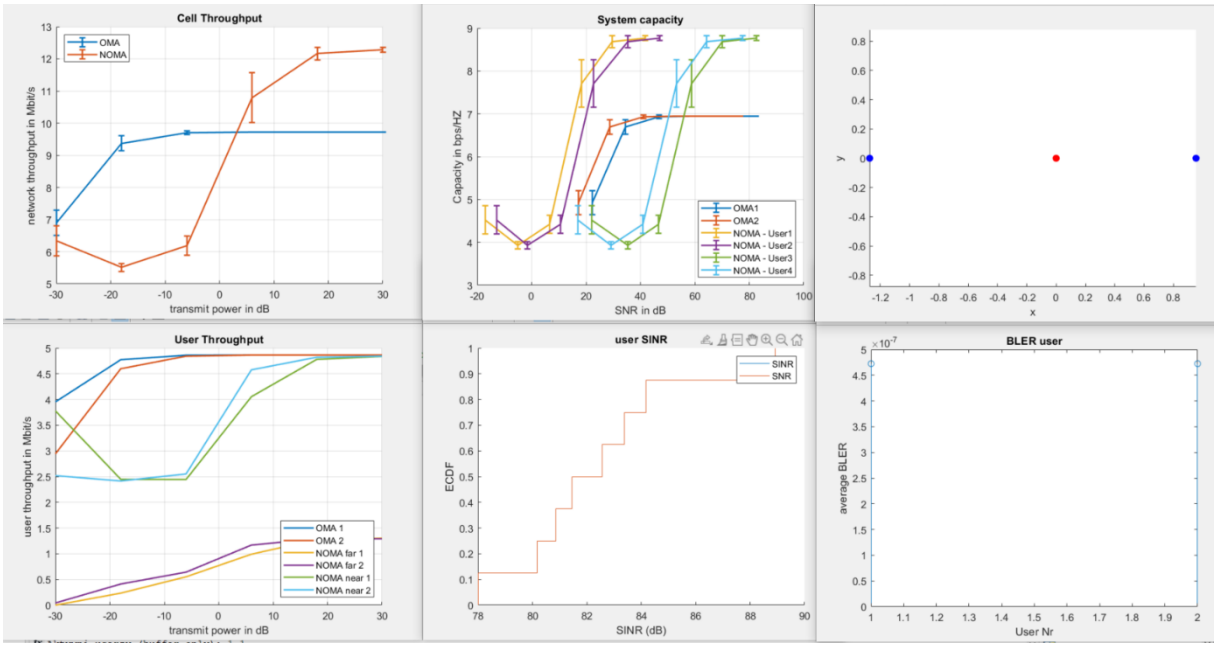
Wnioski: wykres capacity staje się coraz „ładniejszy”. Natomiast trzeba jeszcze podzielić wykresy „userów” tak żeby dostać ich „capacity share”.

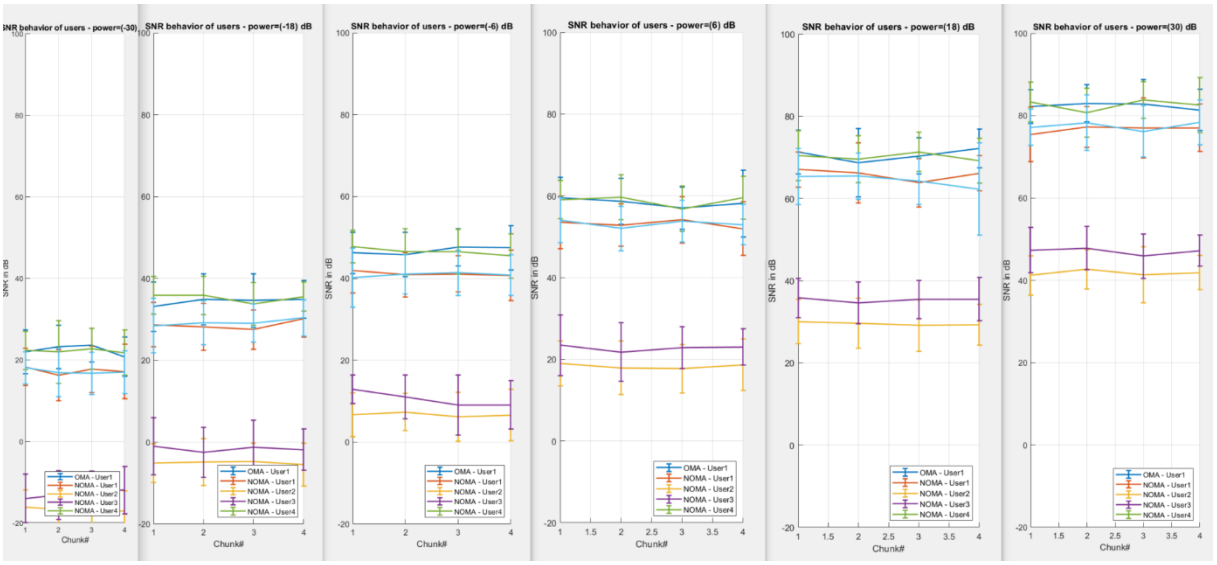
Analiza danych odnośnie SNR dla NOMA (**MIMO**, nSlots =10; nRep=32; BxPower=6, chunk=4;)

Symulacja przebiega poprawnie pomimo długiej realizacji (900sek).

Pewnie dlatego że mało slotów.

Scheduler: RoundRobin





Aby naprawić sposób uśredniania, poniżej informacja o poczynionych poprawkach:

Poprawa sposobu liczenia średniej SNR. Teraz liniowo uśredniamy, a pokazujemy w skali logarytmicznej.

wewnątrz pętli dla symulacji NOMA:

launcherNomaMODIFIED2.m

```
[a1 b1] = size(result.SNR_DL); % sprawdzamy ile jest danych: a-users, b-chunks
SNRforNOMA(:,:,:,b1+1:end) = []; % przycinamy do właściwego rozmiaru
SNRforNOMA(:,:,a1+1:end,:) = [];

%SNRforNOMA(iBsPower, iRep, :, :) = tools.todB(result.SNR_DL);
SNRforNOMA(iBsPower, iRep, :, :) = result.SNR_DL; %NIE ZAMIENIAM na dB bo będę chciał uśredniać
```

wewnątrz pętli dla symulacji OMA:

launcherNomaMODIFIED2.m

```
[a3 b3] = size(result.SNR_DL); % sprawdzamy ile jest danych: a-users, b-chunks
SNRforOMA(:,:,:,b3+1:end) = []; % przycinamy do właściwego rozmiaru
SNRforOMA(:,:,a3+1:end,:) = [];

%SNRforOMA(iBsPower, iRep, :, :) = tools.todB(result.SNR_DL);
SNRforOMA(iBsPower, iRep, :, :) = result.SNR_DL; %NIE ZAMIENIAM na dB bo będę chciał uśredniać
```

A następnie przygotowanie danych po nowemu dla "per user capacity"

launcherNomaMODIFIED2.m*

per cell capacity

```
% get average CELL CAPACITY and statistical values for error bars
sumThroughputOMAbest_NORMALIZED     = squeeze(sum(throughputBestOMA_NORMALIZED, 1));
sumThroughputNOMAbest_NORMALIZED    = squeeze(sum(throughputBestNOMA_NORMALIZED, 1));
meanThroughputOMAbest_NORMALIZED    = mean(sumThroughputOMAbest_NORMALIZED,2);
meanThroughputNOMAbest_NORMALIZED   = mean(sumThroughputNOMAbest_NORMALIZED,2);
stdThroughputOMAbest_NORMALIZED     = std(sumThroughputOMAbest_NORMALIZED, 0, 2);
stdThroughputNOMAbest_NORMALIZED    = std(sumThroughputNOMAbest_NORMALIZED, 0, 2);

%% new
% get average USER CAPACITY and statistical values for error bars
% [nUser x nPower x nRepetition x nSlots]
UserCapacityOMAbest_NORMALIZED  = squeeze(throughputBestOMA_NORMALIZED);
UserCapacityNOMAbest_NORMALIZED = squeeze(throughputBestNOMA_NORMALIZED);
meanCapacityOMAbest_NORMALIZED  = mean(UserCapacityOMAbest_NORMALIZED,3); % było "..,2)" Eksperyment!
meanCapacityNOMAbest_NORMALIZED = mean(UserCapacityNOMAbest_NORMALIZED,3); % było "..,2)" Eksperyment!
stdCapacityOMAbest_NORMALIZED   = std(UserCapacityOMAbest_NORMALIZED, 0, 3); % było "..,2)" Eksperyment!
stdCapacityNOMAbest_NORMALIZED  = std(UserCapacityNOMAbest_NORMALIZED, 0, 3); % było "..,2)" Eksperyment!
```

new! (per user)

UWAGA: trzeba zwrócić uwagę na ostatni wymiar w mean() oraz std(). Ustawiłem na "3".

Modyfikacja funkcji do wykresów poniżej:

skalę logarytmiczna robimy dopiero PO uśrednieniu

launcherNomaMODIFIED2.m

```
%end

for u=1:usersO   % OMA
    avgPerUE = []; % czyść wartości SNR dla usera
    for p=1:numel(powerLevels)
    % SNRforOMA(:,:,:,:) => [BS x Repetition x User x Chunk]
    % errorbar(SNRforOMA(:,1,1,1),  meanThroughputOMAbest_NORMALIZED,  stdThroughputOMAbest_NORMALIZED,  'LineWidth', 1.5);
        avgPerUE = [avgPerUE, tools.todB(mean(mean(snrOMA(p,:,u,:),4)))]; % średnia per ALL-repetitions, ALL-chunks
    end
    %errorbar(avgPerUE', meanThpOMA, stdThpOMA, 'LineWidth', 1.5); % krzywa per jeden user!
    errorbar(avgPerUE', meanThpOMA(u,:), stdThpOMA(u,:), 'LineWidth', 1.5); % krzywa per jeden user!
end
```

wersja poprzednia

nowe zmienne mean/std (inaczej liczone)

Fig. plotCapacity_vs_SNR_withPower() → OMA

launcherNomaMODIFIED2.m

```
%% TESTING with power   NOMA
for u=1:usersN
    avgPerUE = []; % czyść wartości SNR dla usera
    for p=1:numel(powerLevels)
    % SNRforOMA(:,:,:,:) => [BS x Repetition x User x Chunk]
    % errorbar(SNRforNOMA(:,1,1,chunk), meanThroughputNOMAbest_NORMALIZED, stdThroughputNOMAbest_NORMALIZED, 'LineWidth', 1.5);
    % WERSJA wcześniejsza --> errorbar(snrNOMA(:,1,j,chunkToPlot), meanThpNOMA, stdThpNOMA, 'LineWidth', 1.5);
        avgPerUE = [avgPerUE, tools.todB(mean(mean(snrNOMA(p,:,u,:),4)))]; % średnia per ALL-repetitions, ALL-chunks
    end
    %errorbar(avgPerUE', meanThpNOMA, stdThpNOMA, 'LineWidth', 1.5); % krzywa per jeden user!
    errorbar(avgPerUE', meanThpNOMA(u,:), stdThpNOMA(u,:), 'LineWidth', 1.5); % krzywa per jeden user!
end
```

poprzedni sposób rysowania

najpierw uśrednij potem log.

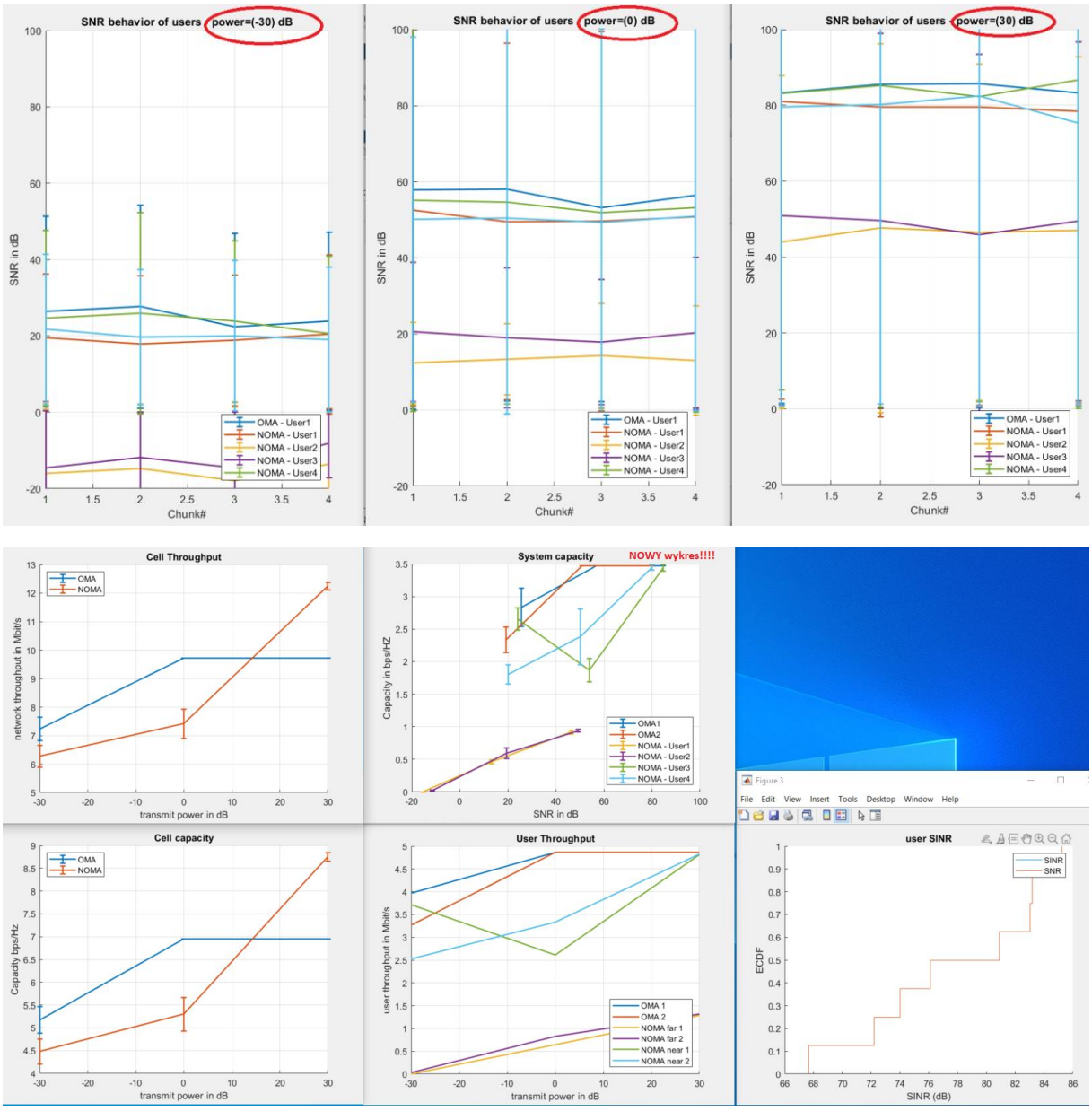
nowe wartości zmiennych mean/std

Fig. plotCapacity_vs_SNR_withPower() → NOMA

Analiza danych odnośnie SNR dla NOMA (**MIMO**, nSlots =10; nRep=5; BxPower=3, chunk=5;)

User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 80dB PL; User_OMA2 = 85dB PL
Scheduler: RoundRobin

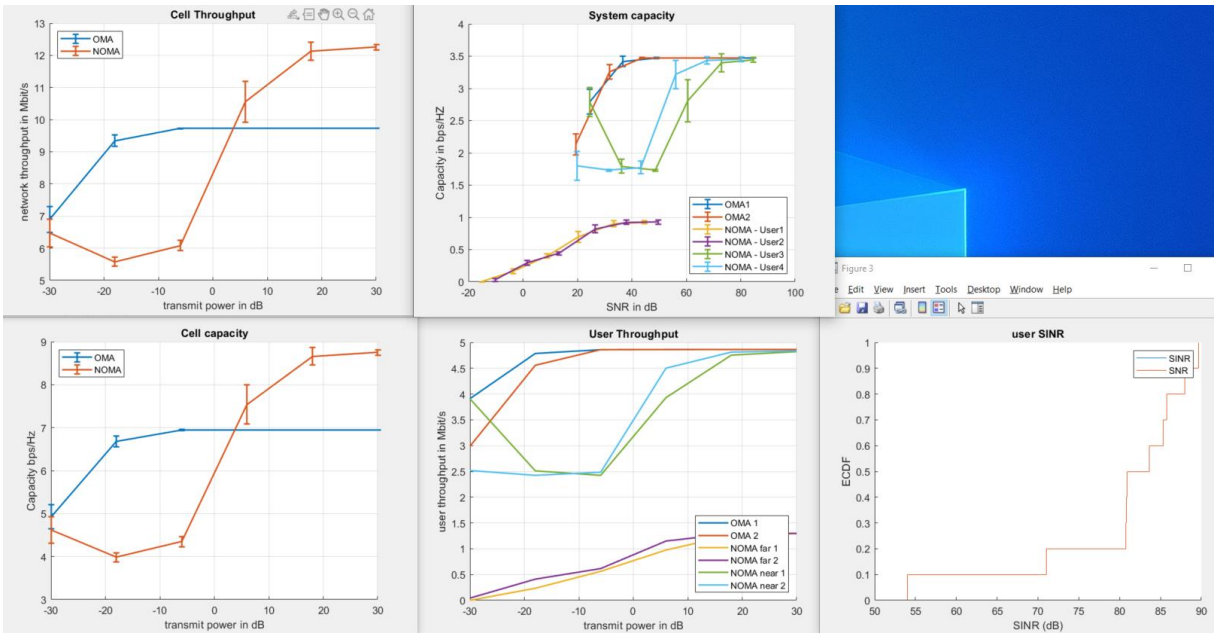
UWAGA: poprawiony sposób wyznaczania średniej, teraz średnia jest robiona dla skali liniowej a nie logarytmicznej, dopiero PO uśrednieniu wynik jest zamieniany na dB.



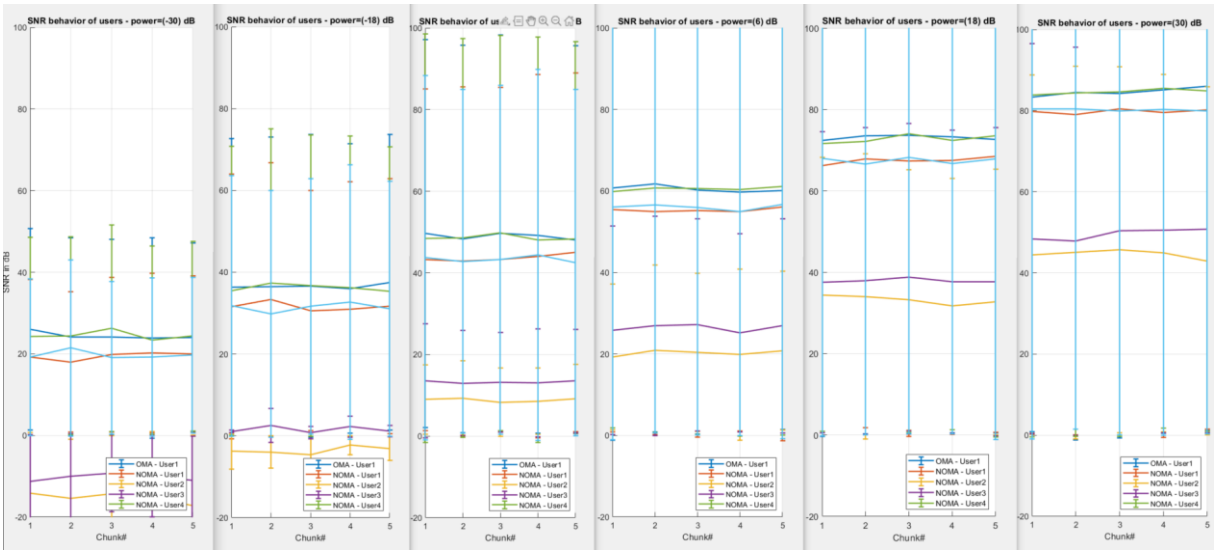
WNIOSKI: teraz widać na środkowym, górnym wykresie "capacities per user". A w lewym dolnym rogu, capacity per cell.

Analiza danych odnośnie SNR dla NOMA (**MIMO**, nSlots =10; nRep=32; BxPower=6, chunk=5;) -> około 25min

User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 80dB PL; User_OMA2 = 85dB PL
Scheduler: RoundRobin

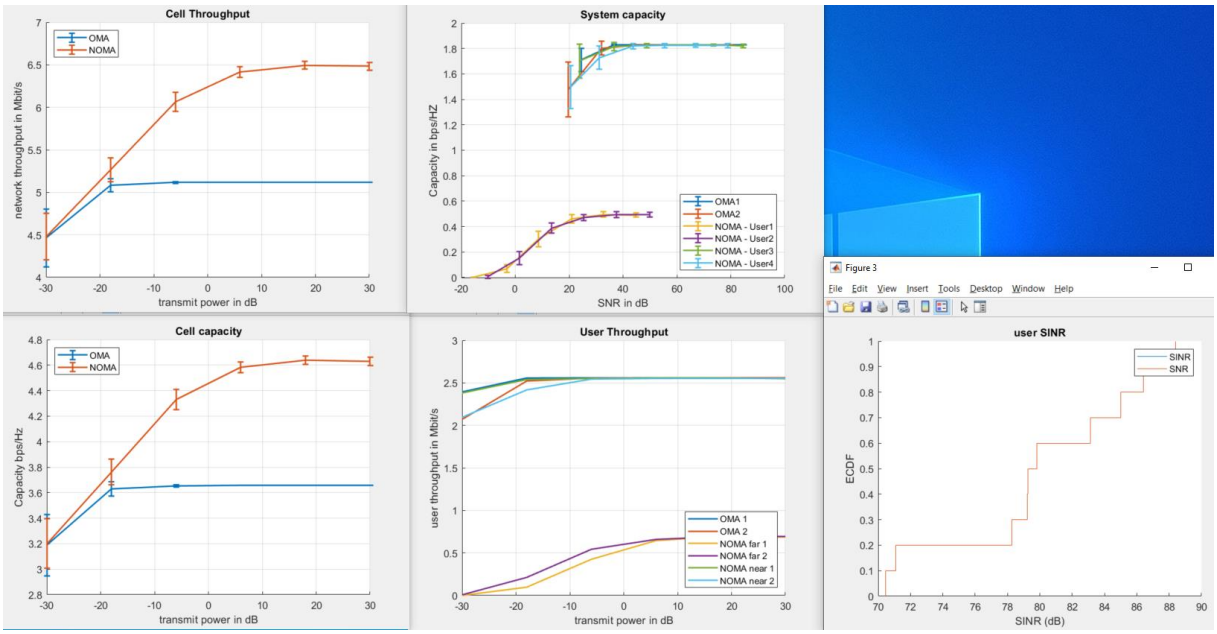


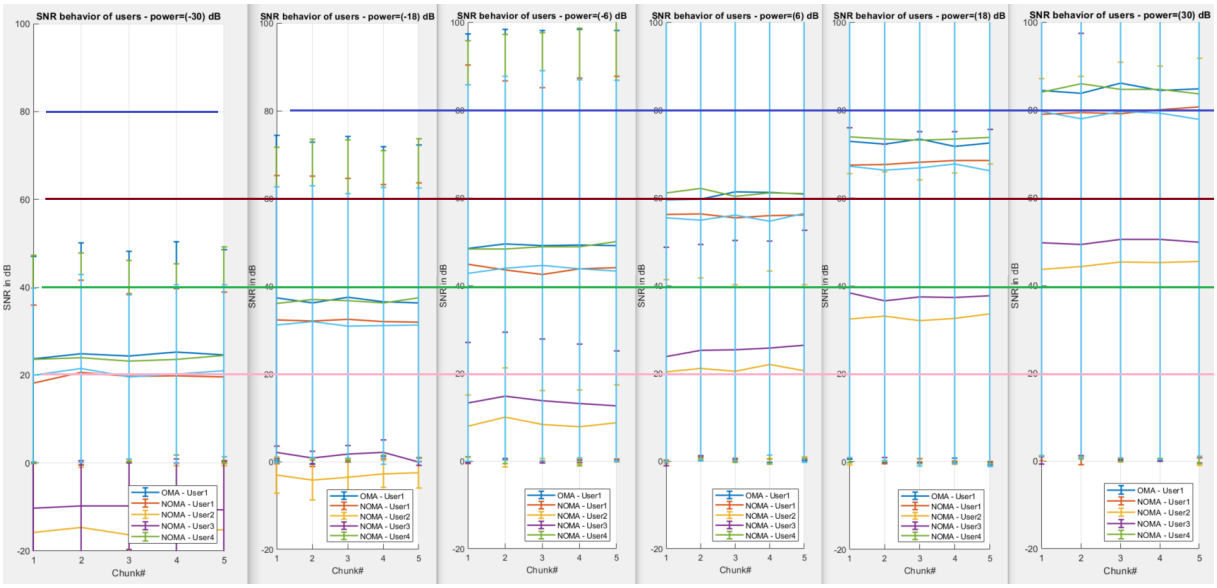
WNIOSKI: kluczowy wykres - środek, góra.



Analiza danych odnośnie SNR dla NOMA (**SISO**, nSlots =10; nRep=32; BxPower=6, chunk=5;)--> czas=900sek

User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 80dB PL; User_OMA2 = 85dB PL
Scheduler: RoundRobin

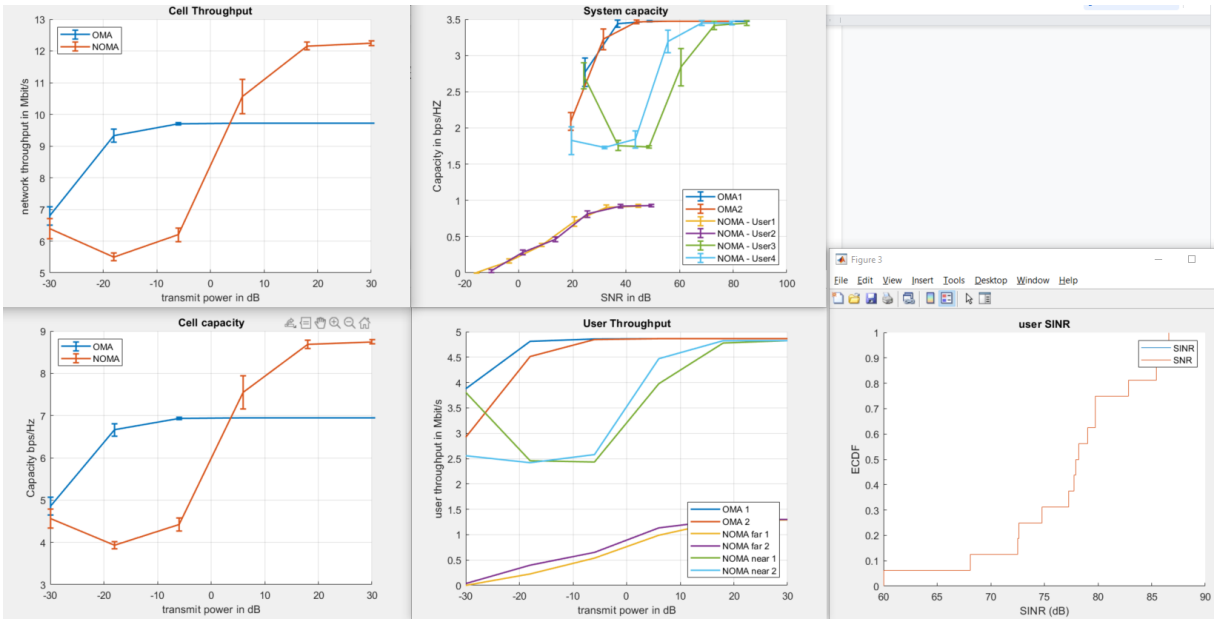




Analiza danych odnośnie SNR dla NOMA (**MIMO**, nSlots =10; nRep=32; BxPower=6, chunk=8;) -> czas = 25min

User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 80dB PL; User_OMA2 = 85dB PL

Scheduler: RoundRobin



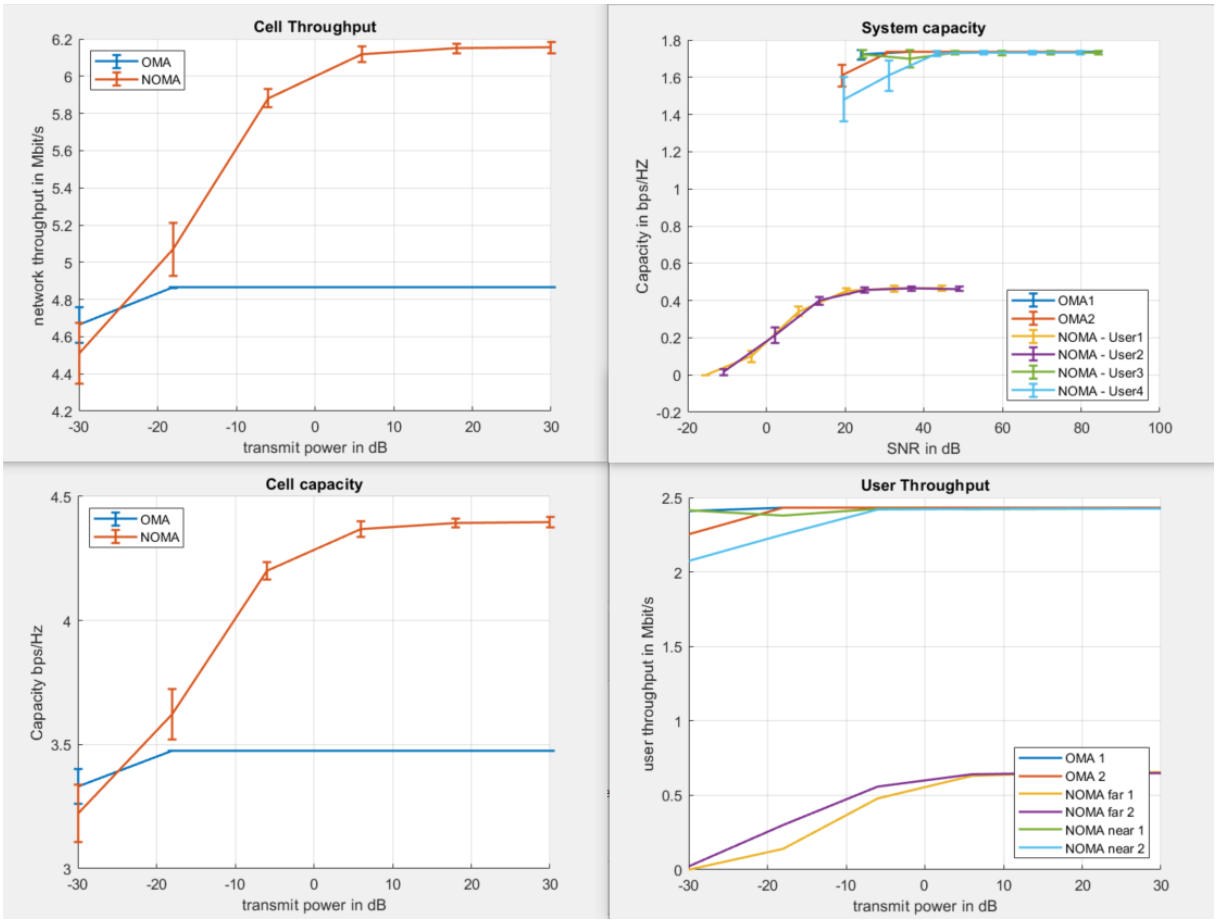
Analiza danych odnośnie SNR dla NOMA (**MISO**, nSlots =10; nRep=32; BxPower=6, chunk=8;) -> czas = 25min

User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 80dB PL; User_OMA2 = 85dB PL

Scheduler: RoundRobin

UWAGA: zmieniamy parametr "dim" na końcu funkcji mean() i std() wewnątrz plotUserSNRperChunk() na "3" bo na wykresach powyżej paski "std" są bardzo długie.

```
end
for i=1:usersN
    %stdSNR_U2OMA  = squeeze(std( SNRforOMA(1,:,2,:),0,2));
    %meanSNR_U4NOMA = squeeze(mean(SNRforNOMA(1,:,4,:),2));
    %errorbar(xAxis', meanSNR_U2NOMA,  stdSNR_U2NOMA, 'LineWidth', 1.5);
    % TEST --- errorbar(xAxis', tools.todB(squeeze(mean(snrNOMA(powerIdx,:,i,:),2))), tools.todB(squeeze(std( snrNOMA(powerIdx,:,i,:),0,2))), 'LineWidth', 1.5);
    errorbar(xAxis', tools.todB(squeeze(mean(snrNOMA(powerIdx,:,i,:),3))), tools.todB(squeeze(std( snrNOMA(powerIdx,:,i,:),0,3))), 'LineWidth', 1.5);
end

%%
```



pojawia się taki błąd, najprawdopodobniej dlatego że zmieniliśmy parametry mean() i std() i na końcu użyłem wymiaru "3" zamiast "2", jak to było do tej pory.

```
352      %errorbar(xAxis', meanSNR_U2NOMA,  stdSNR_U2NOMA, 'LineWidth', 1.5);
353      % TEST --- errorbar(xAxis', tools.todB(squeeze(mean(snrNOMA(powerIdx,:,i,:),2))), tools.todB(squeeze(std( snrNOMA(powe
354 -    errorbar(xAxis', tools.todB(squeeze(mean(snrNOMA(powerIdx,:,i,:),3))), tools.todB(squeeze(std( snrNOMA(powerIdx,:,i,:)
355 -  end
356
357      %%
358 -    legend('OMA - User1', 'NOMA - User1', 'NOMA - User2', 'NOMA - User3', 'NOMA - User4', 'Location', 'southeast')
359 -    xlabel('Chunk#');
360 -    ylabel('SNR in dB');
```

```
Command Window
Sesja usera 2, slot=11 AKTYWNA!
nBitsQueue=Inf, sentBits=2112
Simulation done. (elapsed time: 2.728825e+00 s)
Error using errorbar (line 105)
X-data must be the same size as Y-data.

Error in launcherFiles.launcherNomaMODIFIED2>plotUserSNRperChunk (line 354)
    errorbar(xAxis', tools.todB(squeeze(mean(snrNOMA(powerIdx,:,i,:),3))), tools.todB(squeeze(std(
    snrNOMA(powerIdx,:,i,:),0,3))), 'LineWidth', 1.5);

Error in launcherFiles.launcherNomaMODIFIED2 (line 313)
    plotUserSNRperChunk(p,SNRforOMA,SNRforNOMA, result, bsTxPower);

fx >>
```

	A	B	C	D	E	F	G	H	I	J	K
1	**Repetition**	**Chunk1**	**Chunk2**	**Chunk3**	**Chunk4**						
2	#1	0,387641	1,155285	0,828144	0,01886						
3	#2	0,490901	0,940486	1,152898	0,064969						
4	#3	1,861899	2,899417	1,191463	0,259019						
5	#4	0,691608	1,241088	1,583931	2,591888						
6	#5	0,176116	0,605614	3,002995	0,69453						
7	#6	6,201015	0,442729	1,061259	1,104083						
8	#7	0,115479	2,475464	0,868065	2,207493						
9	#8	1,893195	1,457517	0,476624	2,499501						
10	#9	0,132764	3,709347	1,142236	3,221368						
11	#10	1,33513	1,301469	1,35137	7,423501						
12	#11	2,347296	1,415807	1,074457	0,121131						
13											
14							**Total std.**				
15		1,775103	1,00683	0,648661	2,19075	->	0,703555				
16		**odch standardowe próbki**									
17											
18		`tools.todB(squeeze(std(snrNOMA(powerIdx,:,i,:),0,2))),`									
19											
20											

`squeeze(std(std(snrNOMA(2,:,2,:),0,2),0,4))`

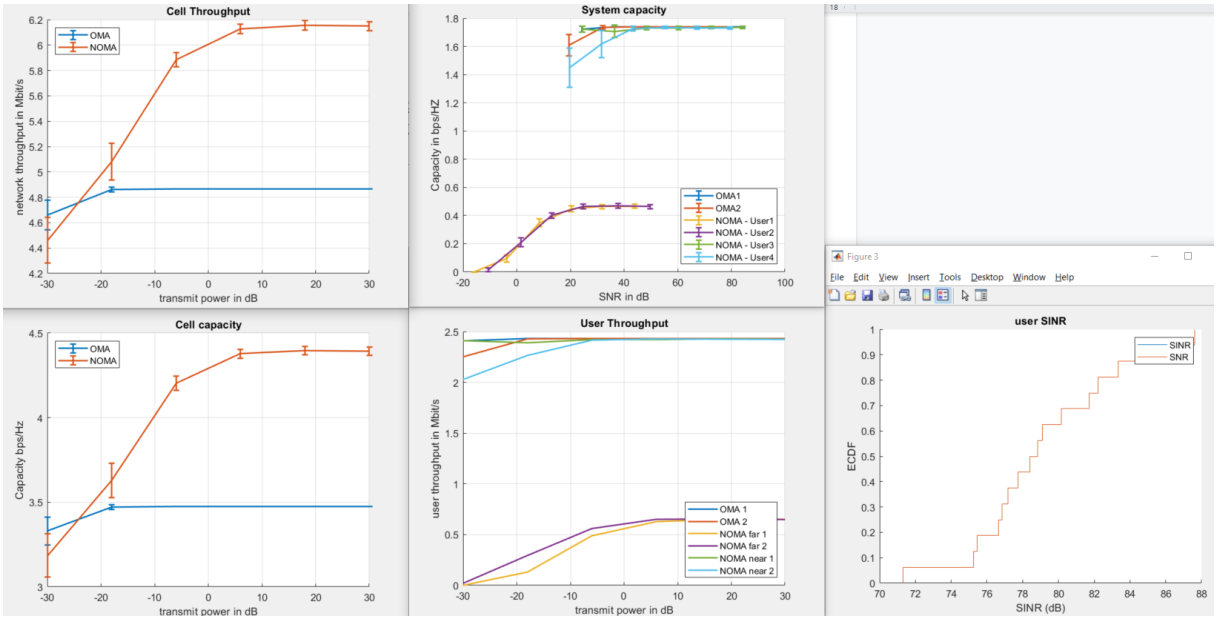
Fig. Tak wygląda wyliczanie odchylenia std dla SNR…

Analiza danych odnośnie SNR dla NOMA (**MISO**, nSlots =10; nRep=32; BxPower=6, chunk=8;) -> czas = 25min

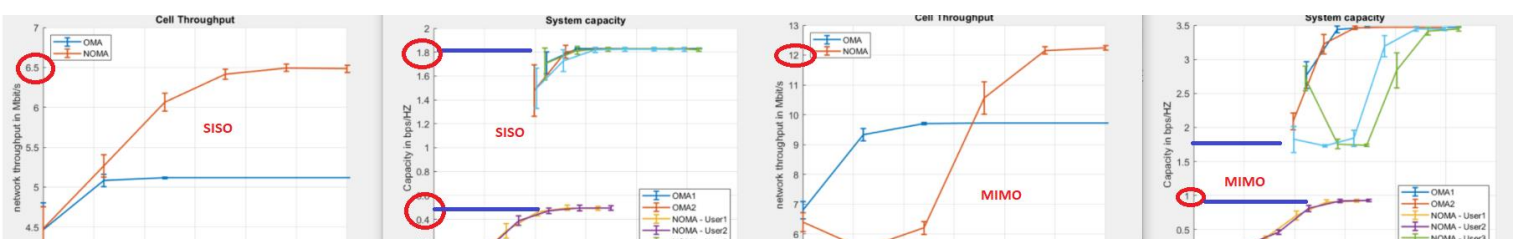
UWAGA: poprawiamy parametr mean() i std() z powrotem na "2" zamiast "3".

User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 80dB PL; User_OMA2 = 85dB PL

Scheduler: RoundRobin



Poniżej znajduje się obecne (13.02.2022) podsumowanie pojemności dla różnej liczby anten: **SISO, MISO, MIMO.**



WNIOSEK: różnicy pomiędzy SISO i MISO dużej nie ma. Różnica pomiędzy nimi a MIMO jest spora! Na wykresie powyżej jest PRZEPŁYWNOŚĆ oraz CAPACITY!

Analiza danych odnośnie SNR dla NOMA (**MISO**, nSlots =10; nRep=11; BxPower=6, chunk=4;) -> czas = 25min

User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 80dB PL; User_OMA2 = 85dB PL

Scheduler: RoundRobin
Sprawdzenie – wykonujemy taką samą symulację jak "jedno wyżej" ale tym razem skorzystam z innej przepływności tj.
result.userThroughputBit.DL.

Zmiana w 3-ech miejscach jak pokazane poniżej.

UWAGA: Zmiany tylko dla pętli "NOMA".

```
39    %RATfor5G
40        % 16.01 -- wyłączam to bo nie pasują wymiary throughputBestNOMA = zeros(nPower, nRepetition, 4, nSlots);
41 -      %result.userThroughputBit.DLBestCQI=0;
42        result.userThroughputBit.DL=0;
```

Czerwony - JEST; Niebieski - było dotychczas

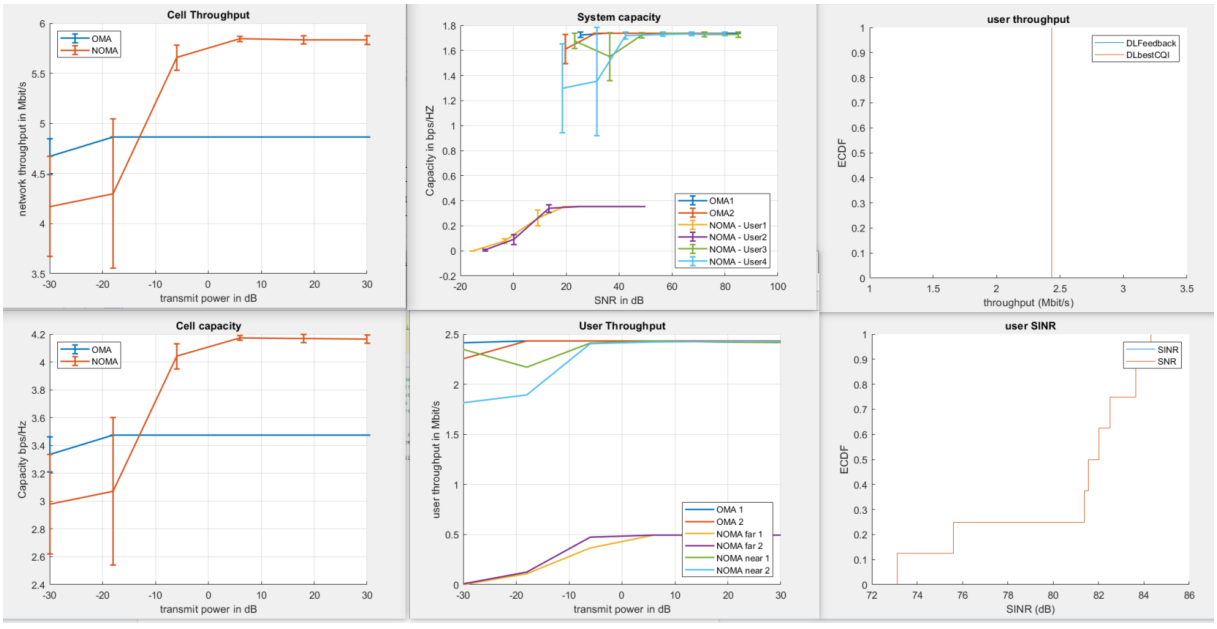
launcherNomaMODIFIED2.m

```
64        % a - liczba userów, b - liczba slotów
65        % UWAGA: liczba userów w przypadku "sessionGenerator" będzie mogła
66        % być RÓŻNA pomiędzy wywołaniami symulacji!!! (tak???)
67        %[a b] = size(result.userThroughputBit.DLBestCQI); % sprawdzamy ile jest danych
68 -      [a b] = size(result.userThroughputBit.DL); % sprawdzamy ile jest danych
69 -      throughputBestNOMA(:,:,:,b+1:end) = []; % przycinamy do właściwego rozmiaru
70 -      throughputBestNOMA(:,:,a+1:end,:) = [];
71
72        %throughputBestNOMA(iBsPower, iRep, :, :) = result.userThroughputBit.DLBestCQI;
73 -      throughputBestNOMA(iBsPower, iRep, :, :) = result.userThroughputBit.DL;
74
75 -      if(iBsPower == 1 || iBsPower == nPower) && (iRep == 1) && showPlots % tylko dla jednego powtórzenia
76 -          a= (10+iBsPower)*iRep;
```

..było → JEST

..było → JEST

BLER = 5*10-7

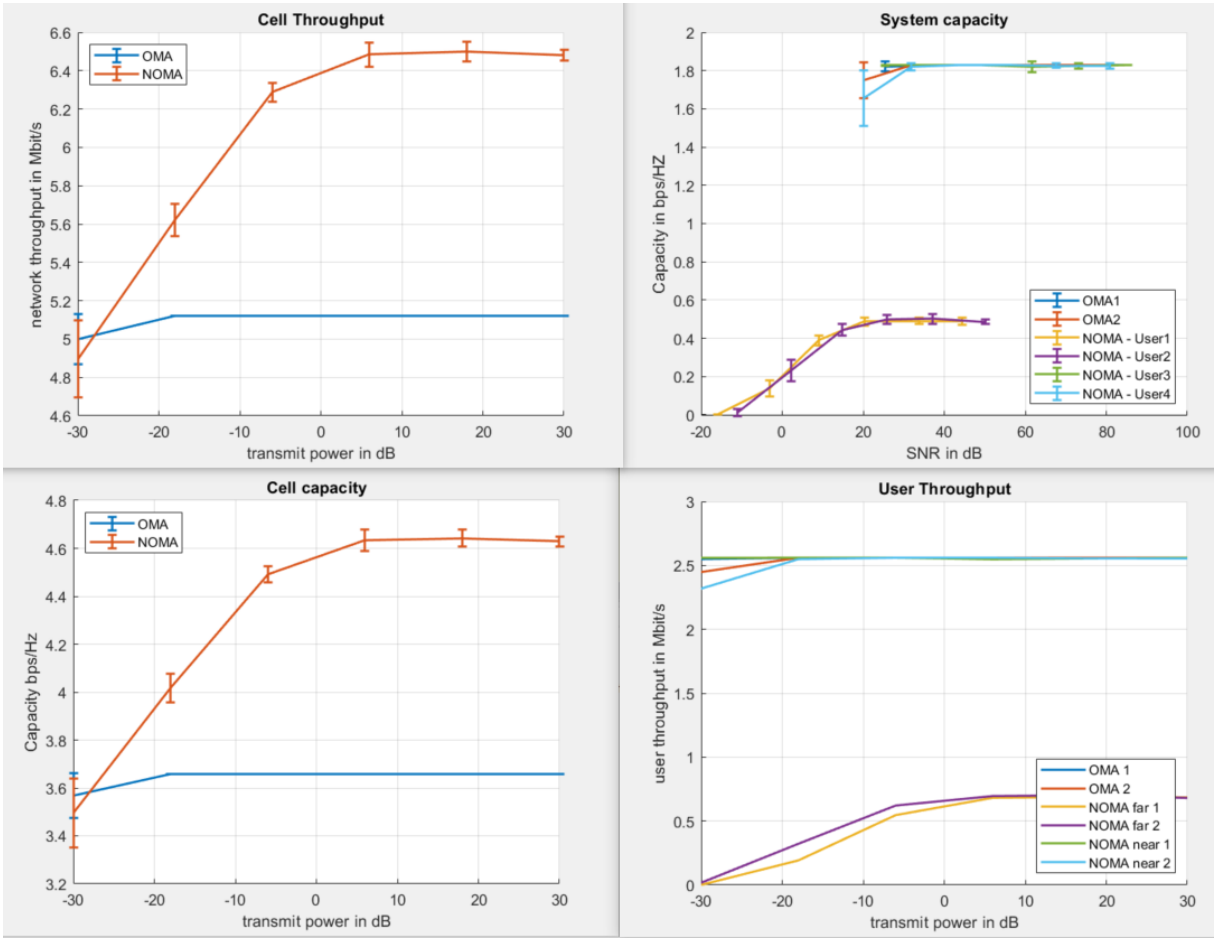


UWAGA: po zakończeniu przywrócono z powrotem *result.userThroughputBit*.**DLBestCQI**

Analiza danych odnośnie SNR dla NOMA (**SIMO**, nSlots =10; nRep=11; BxPower=6, chunk=4;) -> czas = 25min

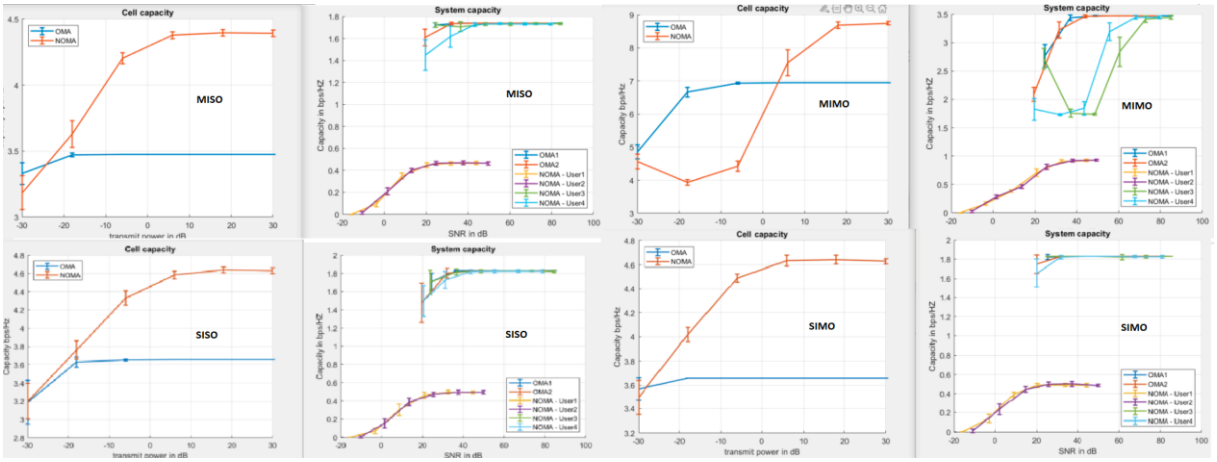
User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 80dB PL; User_OMA2 = 85dB PL

Scheduler: RoundRobin



Podsumowanie wyników dotychczasowych:

Plik: launcherNomaModified2.m (link do tej wersji)



Scheduler: RoundRobin
User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 80dB PL; User_OMA2 = 85dB PL

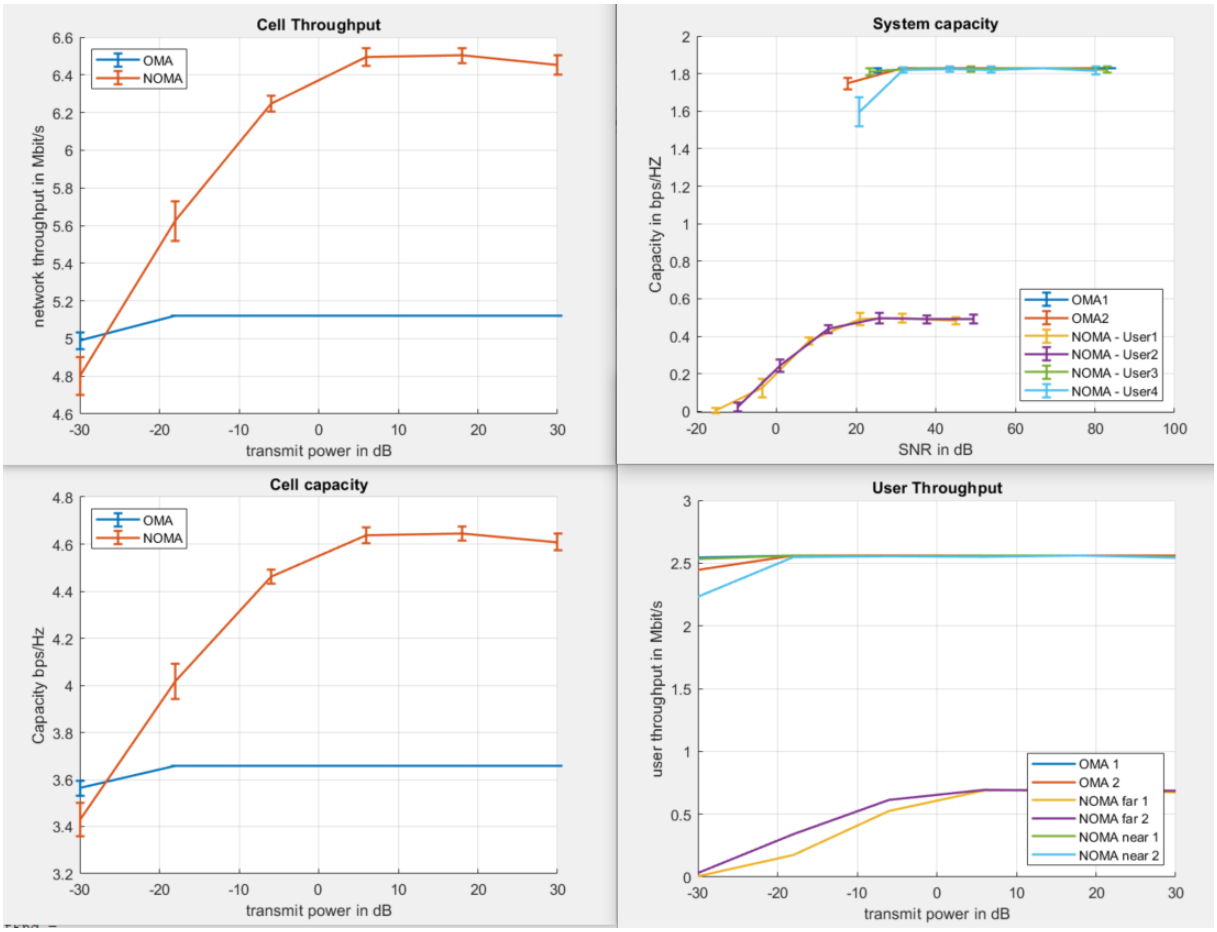
Kanał: Rayleigh

nSlots =10; nRep=11-32; BxPower=6, chunk=4

Analiza danych odnośnie SNR dla NOMA (**SIMO**, nSlots =10; nRep=11; BxPower=6, chunk=4;) -> czas = 25min

params.smallScaleParameters.userSpeed = 20;

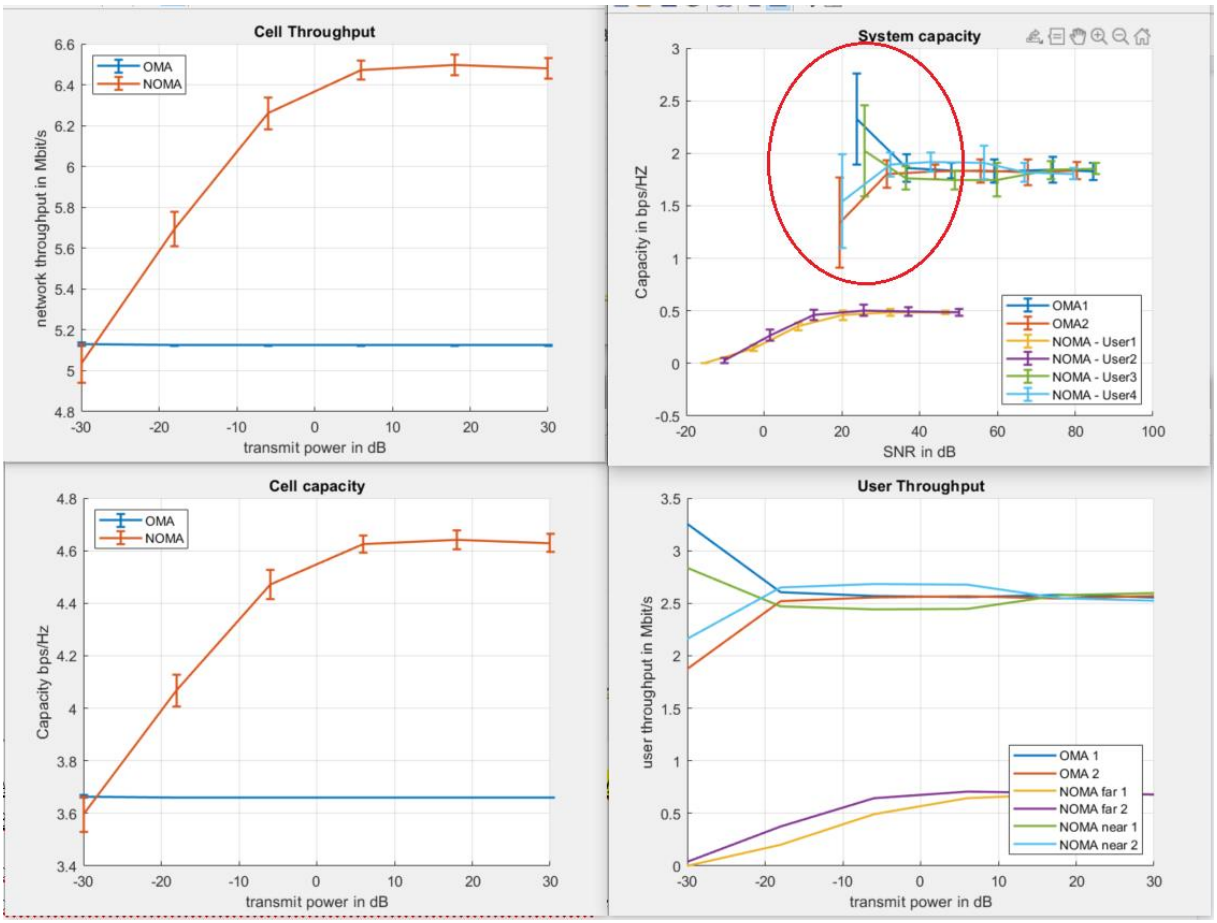
scheduler: RoundRobin



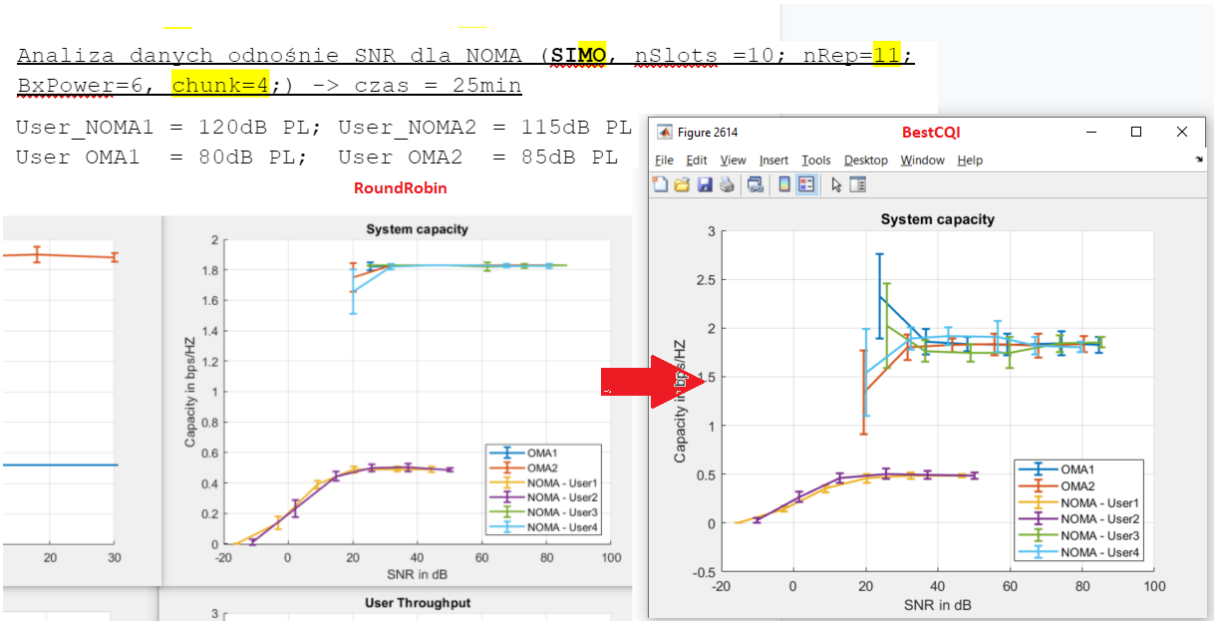
WNIOSEK: nie widać żeby mobility o wartości 20 (m/s??) miało zauważalny wpływ na wartości.

Analiza danych odnośnie SNR dla NOMA (**SIMO**, nSlots =10; nRep=11; BxPower=6, chunk=4;) -> czas = 25min

params.schedulerParameters.type = parameters.setting.SchedulerType.**bestCqi**; (ZAMIAST domyślnego roundrobin)



I jeszcze porównanie z SIMO/RoundRobin. Widać że userzy z lepszym SNR mają nieco wyższe capacity.

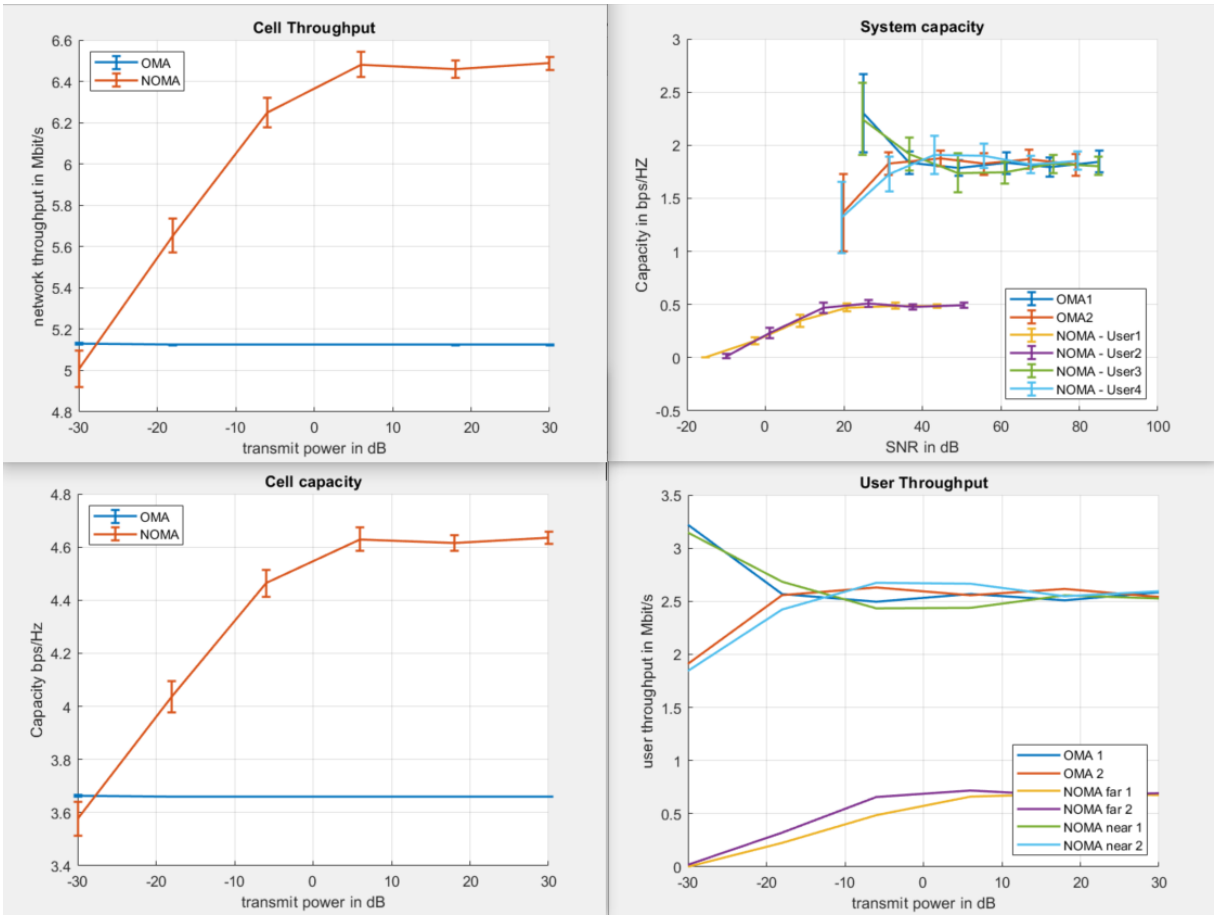


Zmieńmy jeszcze strategię dołączania się do stacji BS. **Ale tutaj więcej stacji będzie potrzebnych..**

```
81
82  % cell association strategy
83  % the cell association metric is used for NOMA user pairing
84  %params.downlinkAssociationStrategy = parameters.setting.CellAssociationStrategy.maxReceivePower;
85  params.downlinkAssociationStrategy  = parameters.setting.CellAssociationStrategy.maxSINR;
86
87  % scheduler:
88  % assign the same amount of resources to all users
89  %params.schedulerParameters.type     = parameters.setting.SchedulerType.roundRobin;
90  params.schedulerParameters.type     = parameters.setting.SchedulerType.bestCqi;
```

noma.m

KOMENTARZ(eksperyment): warto by zrobić pomiary gdzie sprawdzi się: a) wpływ strategii asocjacji usera z BS, b) wpływ schedulera, …

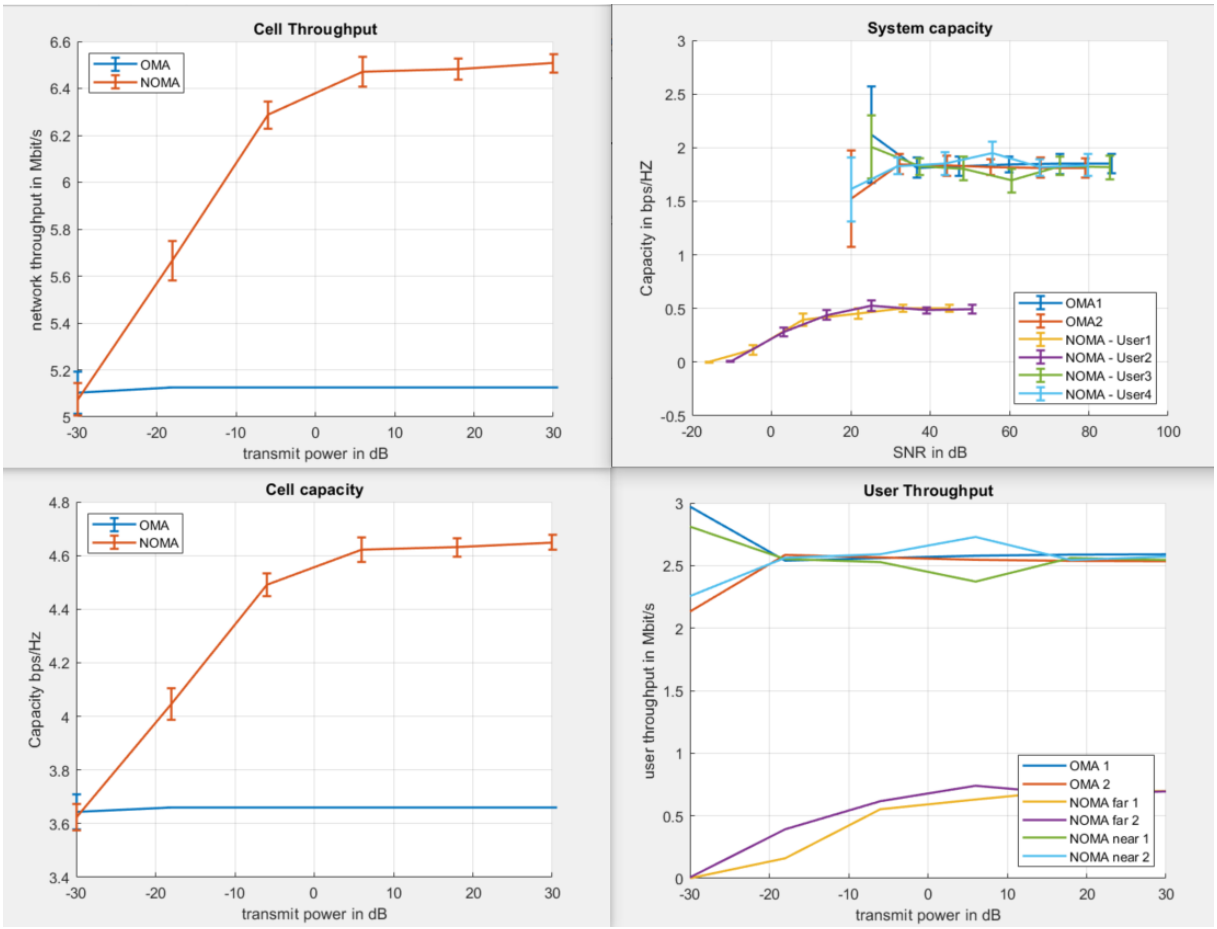


WNIOSEK: jedna stacja bazowa to za mało żeby sprawdzić działanie parametru **cellAssociationStrategy**.Więc przywracam wartość domyślną.

Analiza danych odnośnie SNR dla NOMA (**SIMO**, nSlots =10; nRep=11; BxPower=6, chunk=4;) -> czas = 25min

Alpha ustawiam na 2 (zamiast "4" jak cały czas było w testach powyżej)

Scheduler: **bestCQI**



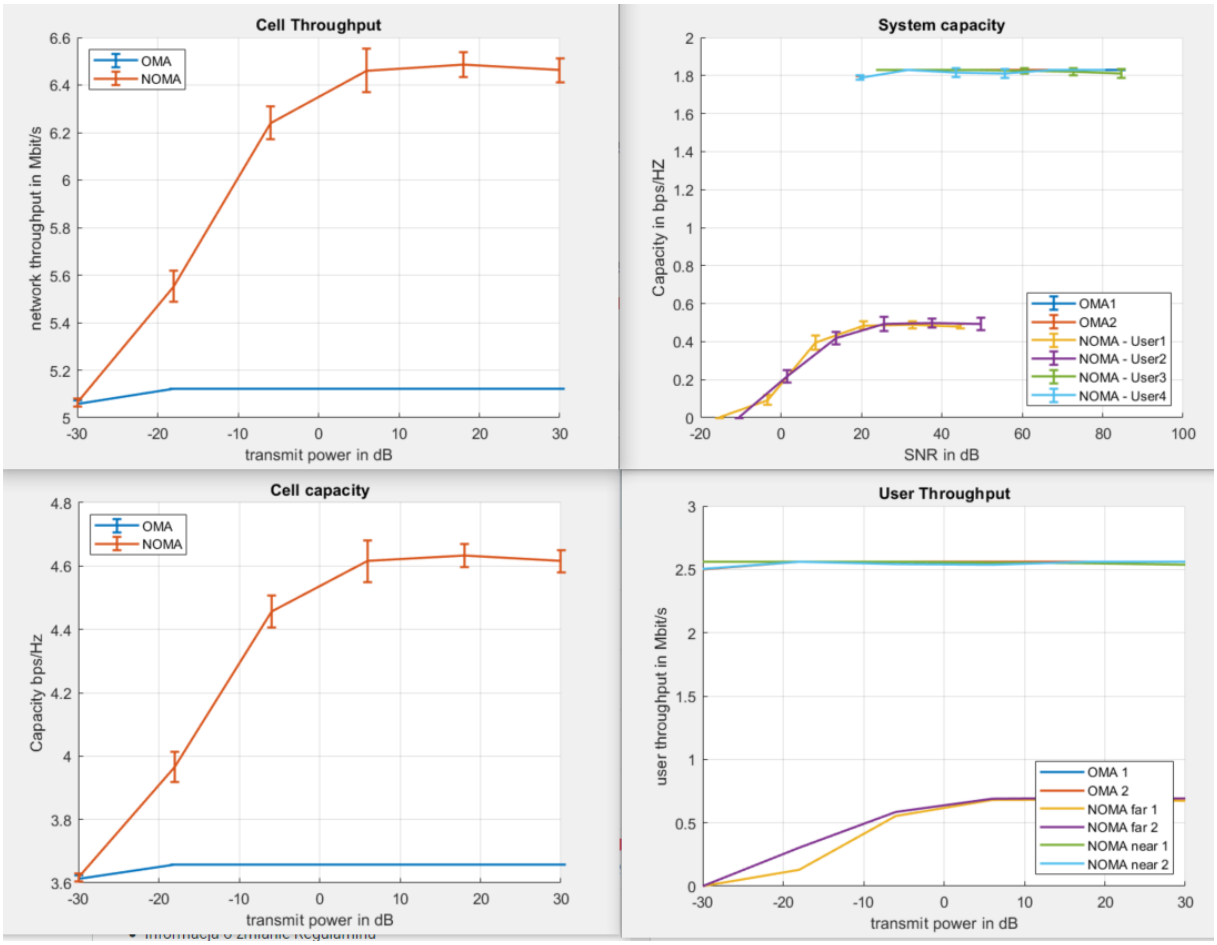
WNIOSEK: nie widać szałowej zmiany w porównaniu do sytuacji gdy alpha=4 (tj. wszystkie wcześniejsze przypadki).

Analiza danych odnośnie SNR dla NOMA (**SISO**, nSlots =10; nRep=11; BxPower=6, chunk=4;) -> czas = 25min

Alpha z powrotem na "4"

Scheduler: **RoundRobin**

Kanał: AWGN!



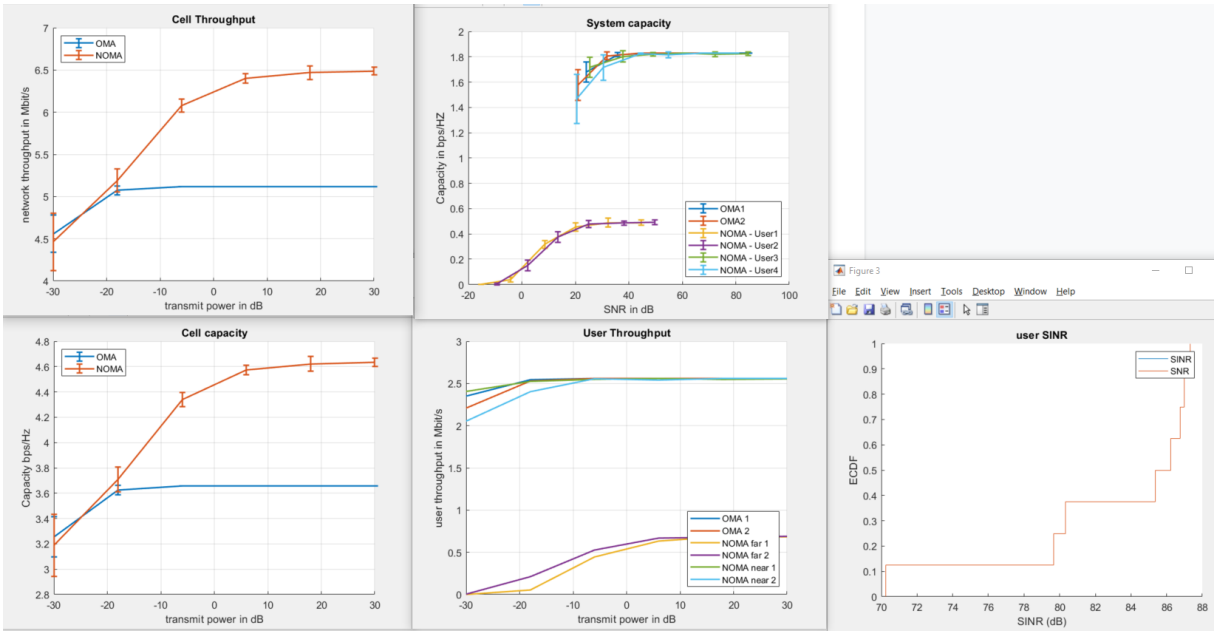
Wniosek: porównanie z kanałem Rayleigh pokazuje że wyniki nie są mocno różne dla AWGN…

Analiza danych odnośnie SNR dla NOMA (**SISO**, nSlots =10; nRep=11; BxPower=6, chunk=4;) -> czas = 25min

Alpha z powrotem na "4"

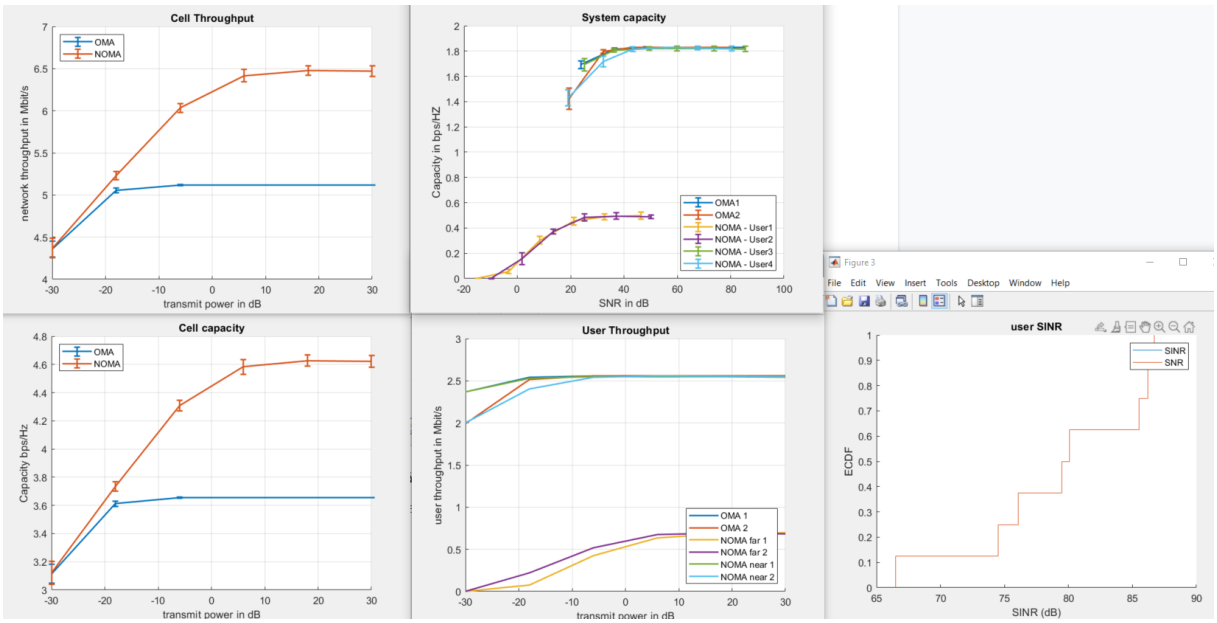
Scheduler: **RoundRobin**

Kanał: VehA!



teraz jeszcze sprawdzenie wpływu prędkości usera (*reszta ustawień taka jak powyżej)

```
params.smallScaleParameters.userSpeed = 20;
```



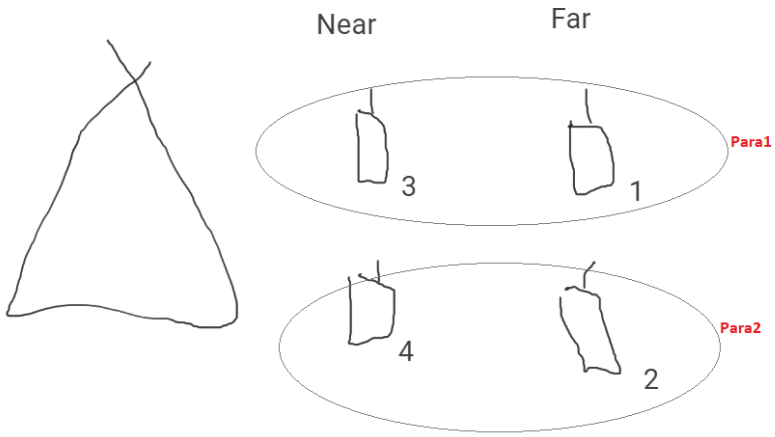
WNIOSKI: na tym poziomie **nie widać** specjalnie wpływu mobility…

Dotychczasowe wnioski

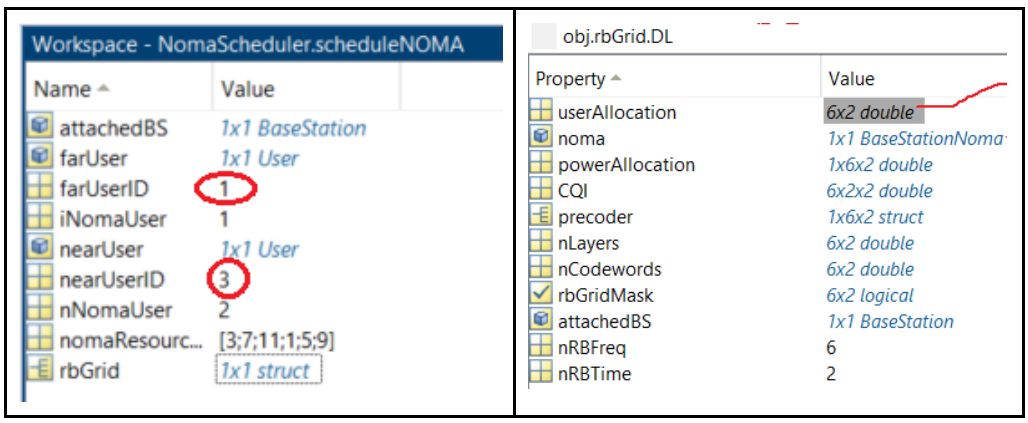
1. Wydajność widmowa NOMA (w symulatorze jest wersja MUST) dla kierunku DL, jest nieco niższa niż to pokazuje literatura zarówno dla SISO jak i dla MIMO
2. Zachowanie wykresów "system throughput" (a co za tym idzie także i capacity bps/Hz) dla NOMA w przypadku MIMO wymagają dalszej analizy (jest spory skok przepływności/capacity w pewnym przedziale Tx power)
3. Zwiększenie wartościowości modulacji dla "far user" NOMA powyżej ustawienia domyślnego dla takiego usera (NOMA), prowadzi do znaczącego zwiększenia problemów z detekcją w odbiorniku (SIC).

Analiza zasobów schedulowanych per slot dla userów OMA/NOMA.

1. Schemat



   a.
2. Parowanie userów
   a.



      i. Liczba podnośnych = 6
      ii. Liczba slotów czasowych = 2
   b. FAR user = 1
   c. NEAR user = 3
3. najpierw userzy są alokowani do zasobów OMA przez RR

a. rbGrid.DL.userAllocation

	1	2
1	1	3
2	2	4
3	3	1
4	4	2
5	1	3
6	2	4

4. Następnie na bazie tego schedulowane są zasoby dla userów NOMA

a.

```
            if nNomaUser
                for iNomaUser = 1:nNomaUser

                    % get NOMA users
                    farUser     = attachedBS.attachedUsers.DL(attachedBS.nomaPairs(1,iNomaUser));
                    nearUser    = attachedBS.attachedUsers.DL(attachedBS.nomaPairs(2,iNomaUser));
                    % get NOMA user IDs
                    farUserID   = farUser.id;
                    nearUserID  = nearUser.id;

                    % Oryginalnie było
                    % if farUser.isActive && nearUser.isActive
                    % RATfor5G --- modyfikacja
                    if (farUser.isActive && farUser.isActiveTraffic(farUser.getSlotRATfor5G()))&&(nearUser.isActive && nearUser.isActiveTraffic(

                        % find resources allocated to near user
                        nomaResources = [find(rbGrid.DL.userAllocation == nearUserID); find(rbGrid.DL.userAllocation == farUserID)];
                        % set near user as NOMA user
                        rbGrid.DL.noma.userAllocation(nomaResources) = nearUserID;
                        % overwrite scheduling to set far user in 'normal' scheduling
                        rbGrid.DL.userAllocation(nomaResources) = farUserID;
                    end % if both users have data to transmit
                end
            end % if there are NOMA user pairs
```

NomaScheduler.m

NOMA resources

OMA resources

b. rbGrid.DL.noma.userAllocation

	1	2	3	4	5
1	3	3			
2	-1	1			
3	3	3			
4	-1	-1			
5	3	3			
6	-1	-1			

NOMA user

5. Potem zasoby w domenie OMA są nadpisywane dla usera “1” z pary NOMA, w ten sposób że miejsca “po userze 3” są zapisywane przez usera “1”

a. rbGrid.DL.userAllocation

	1	2	3	4
1	1	1		
2	2	4		
3	1	1		
4	4	2		
5	1	1		
6	2	4		
7				
8				
9				
10				
11				

6. Teraz to samo jest robione dla drugiej pary userów
   a. Workspace - NomaScheduler.scheduleNOMA

Name	Value
attachedBS	*1x1 BaseStation*
farUser	*1x1 User*
farUserID	2
iNomaUser	2
nearUser	*1x1 User*
nearUserID	4
nNomaUser	2
nomaResourc...	[3;7;11;1;5;9]
rbGrid	*1x1 struct*

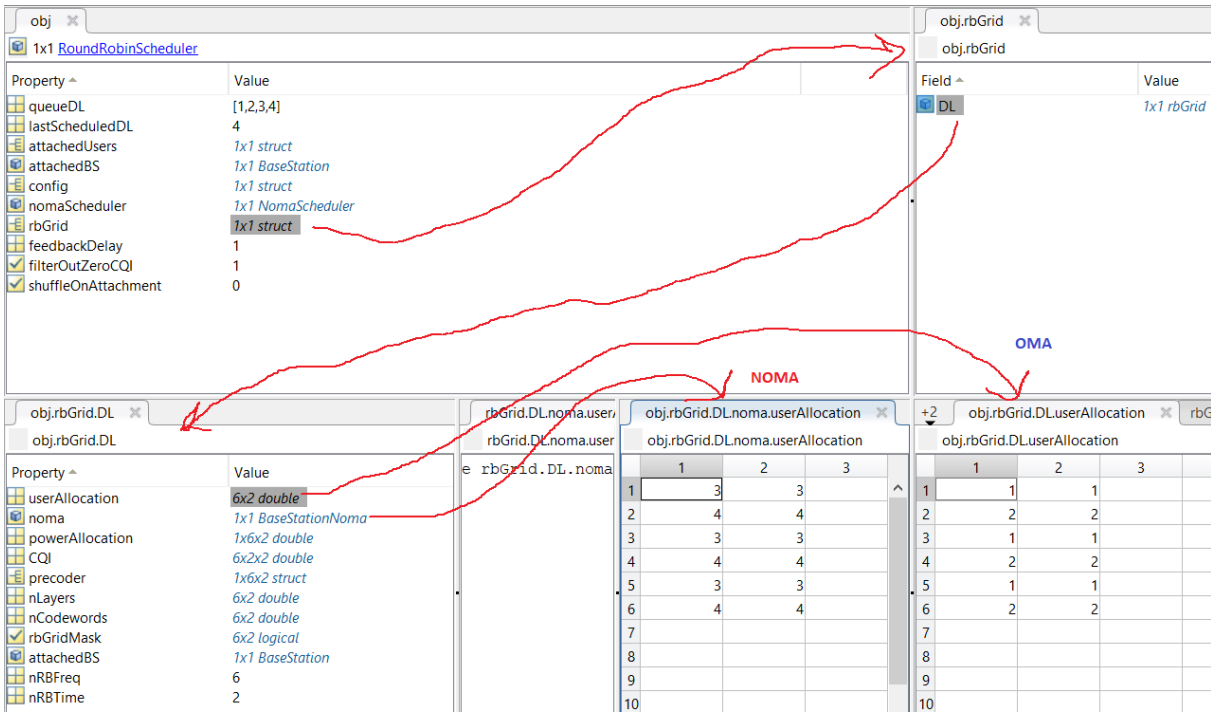
7. Po zrealizowaniu kroków schedulera z punktu “4.a” przypisanie zasobów (linie101, 103) wygląda tak
   a. rbGrid.DL.noma.userAllocation

	1	2	3	4	5
1	3	3			
2	4	4			
3	3	3			
4	4	4			
5	3	3			
6	4	4			
7					

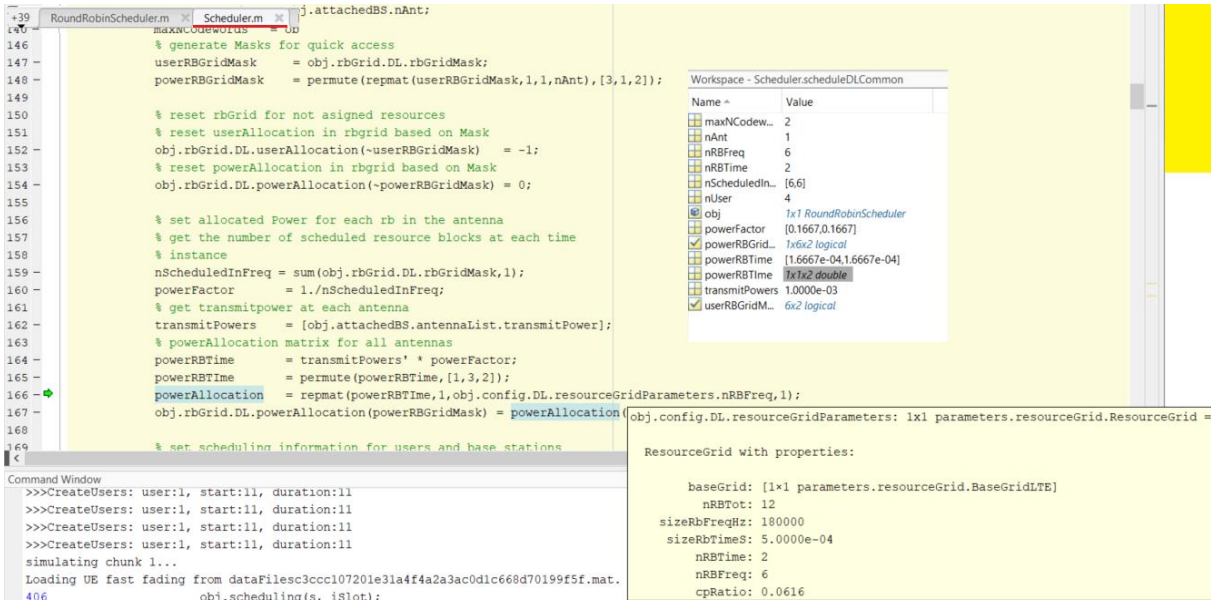
rbGrid.DL.userAllocation

	1	2	3
1	1	1	
2	2	2	
3	1	1	
4	2	2	
5	1	1	
6	2	2	
7			

8. No i po wyjściu z funkcji NomaScheduler… wracamy do RoundRobinScheduler i tam mamy te alokacje już gotowe

a. 

9. Now we go to “Scheduler.m” → **scheduleDLCommon**()



a.
b. widać:
   i. wszystkie istotne parametry ramki OFDMA
c. następnie realizujemy powerAllocation zgodnie z ww. parametrami

obj | powerRBGridMask | powerAllocation

1x6x2 double

```
val(:,:,1) =

   1.0e-03 *

    0.1667    0.1667    0.1667    0.1667    0.1667    0.1667


val(:,:,2) =

   1.0e-03 *

    0.1667    0.1667    0.1667    0.1667    0.1667    0.1667
```

d.

10. Później już alokowane są zasoby per user

a.

```
179                 % get indices of RBs assigned to this user
180 -               obj.attachedUsers.DL(iUser).scheduling.setUserAllocation(obj.rbGrid.DL.userAllocation, obj.attachedUsers.DL(iUser).id);
181 -               assignedRBs = obj.attachedUsers.DL(iUser).scheduling.assignedRBs;
182
183 -               if obj.attachedUsers.DL(iUser).scheduling.nRBscheduled %% here skip "near_NOMA" users , but schedule "OMA" + "far_NOMA"
184
```

i. w linii 180 są wyciągane alokacje usera **iUser = 1**

+3 | obj.attachedUsers | obj.attachedUsers.DL | obj.attachedUsers.DL(1, 1) | obj.attachedUsers.DL(1, 1).scheduling

obj.attachedUsers.DL(1, 1).scheduling

Property	Value
assignedRBs	[1;3;5;7;9;11]
iRBFreq	[1;3;5;1;3;5]
iRBTime	[1;1;1;2;2;2]
nRBscheduled	6
nCodeword	0
nLayer	1
CQI	[ ]
nomaPowerShare	1
noma	1x1 UserNoma

b.

c. **WAŻNE:** jak widać w linii 183 powyżej najpierw są alokowani userzy "OMA" oraz "far NOMA"

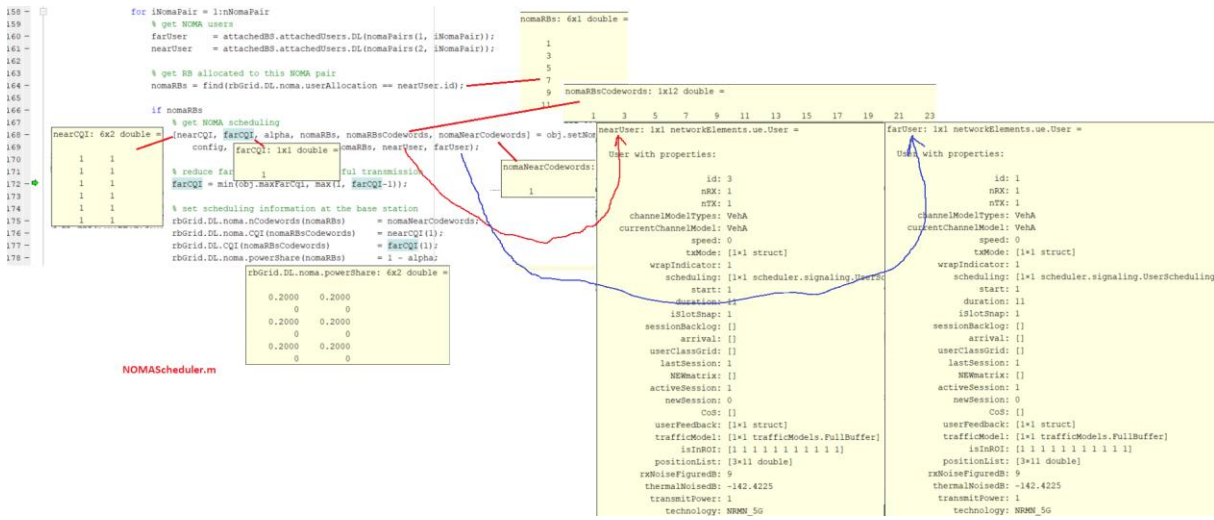
d. dla userów "near NOMA" **assignedRBs** są puste

Workspace - Scheduler.scheduleDLCommon

Name	Value
ans	0
assignedRBs	[ ]
assignedRBsCodewords	1x12 double
feedback	1x1 FeedbackLTE
iAntenna	1
iUser	4
maxNCodewords	2
nAnt	1
nRBFreq	6
nRBTime	2
nScheduledInFreq	[6,6]
nTX	1
nUser	4
obj	1x1 RoundRobinScheduler

e. dla tych "near NOMA" alokacja następuje już PO WYJŚCIU z pętli patrz "Ad.10a"

```
225
226 -             % set user signaling
                  obj.attachedUsers.DL(iUser).scheduling.setUserScheduling(obj.rbGrid.DL, maxNCodewords);
227
228 -         end %for all users attached to this base station
229
230         % set the NOMA scheduling - this can overwrite the scheduling
231 -       obj.nomaScheduler.schedulingNOMAcommon(obj.attachedBS, obj.config, obj.rbGrid);
232
```

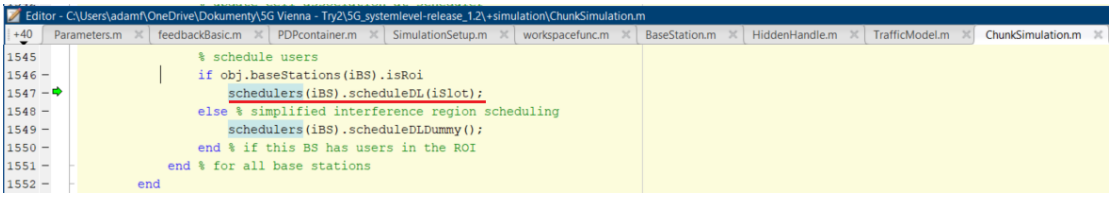


f.

g. teraz już wychodzimy z linii obj.nomaScheduler.scheduleNOMA()... (która jest w **RoundRobinScheduler**.m)

h. idziemy do kolejnej linijki w tej samej klasie tj. obj.scheduleDLCommon(currentTime) – tu mi się nic nie złapało

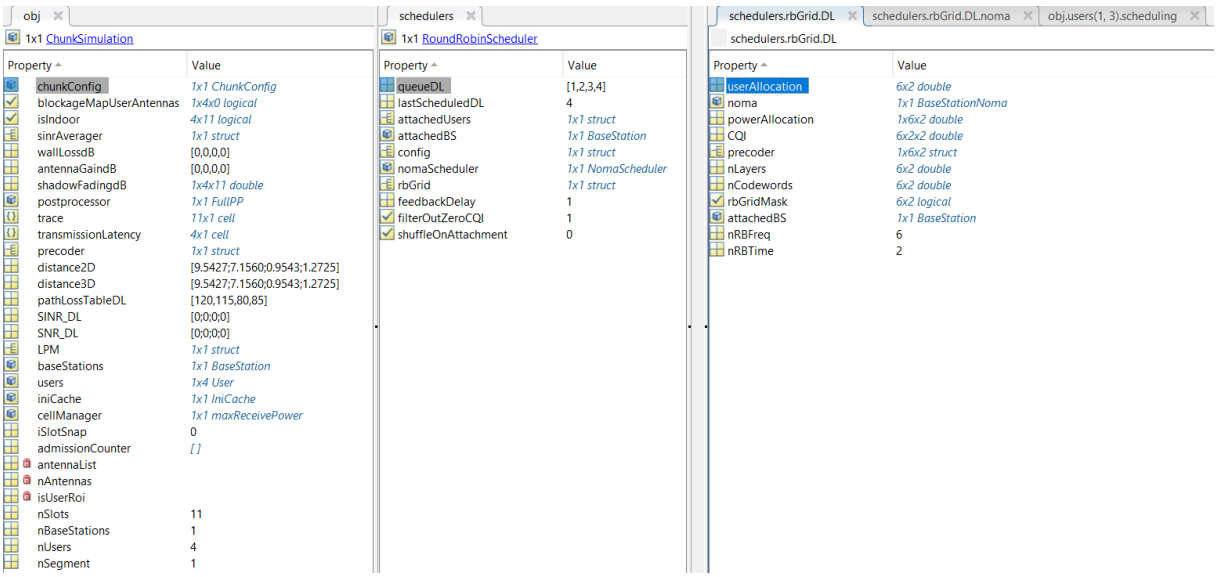
i. no i wyjście z powrotem do **ChunkSimulation**…



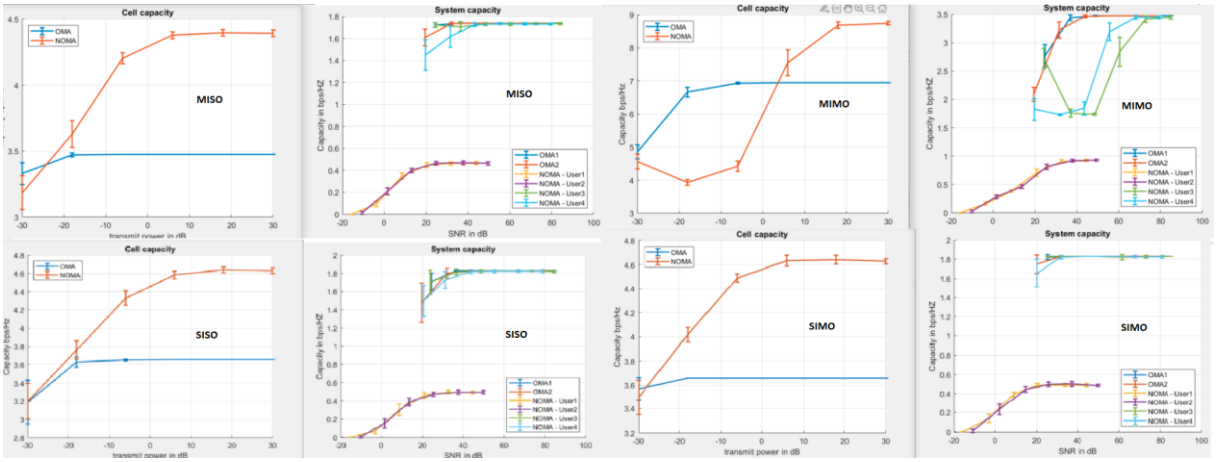
i.

j.

11. PO powrocie do ChunkSimulation.m mamy



Wcześniejsze wyniki



Scheduler: RoundRobin
User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 80dB PL; User_OMA2 = 85dB PL

Kanał: Rayleigh

Porównanie z materiałem [Paper17]

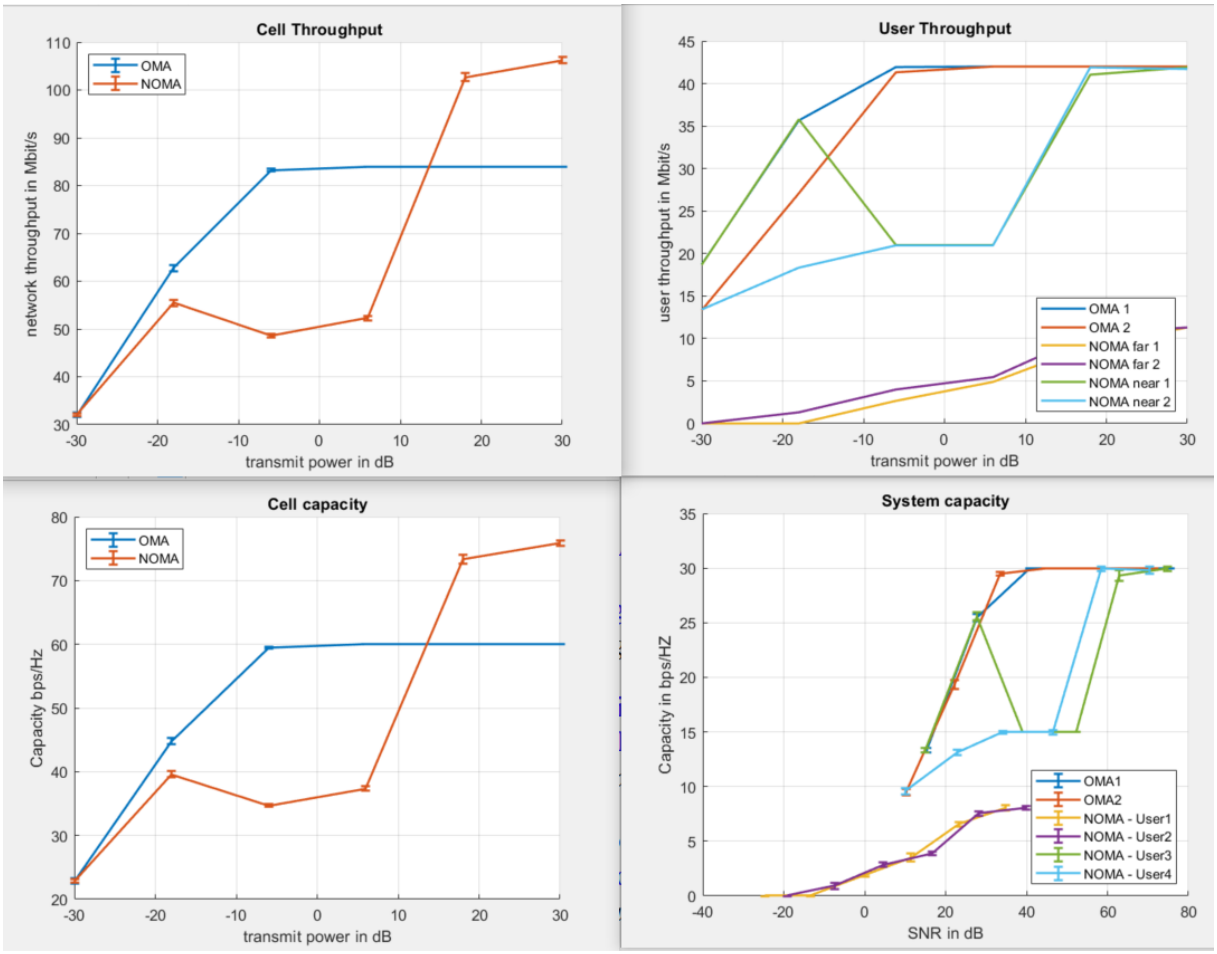
MIMO: 3x2 → UWAGA! dla "3" nie działa, jest błąd ze względu na ograniczenie symulatora! **Zmieniłem na 2x2.**

BW: 10MHz

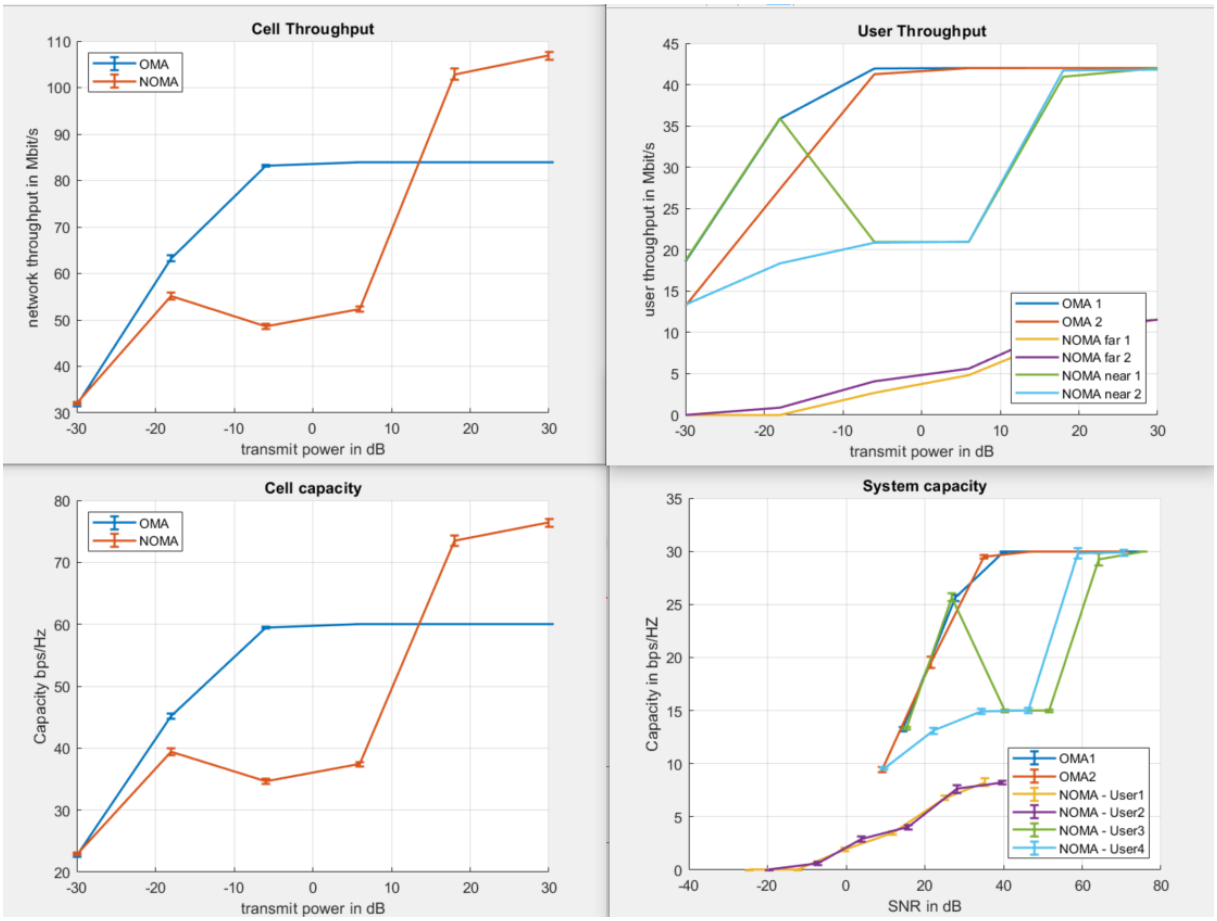
Thermal noise: -174 dBm

PowerShare (NomaScheduler.m) = **0,8** (dla MustIDx = 1)

Czas 760sek (~14min)

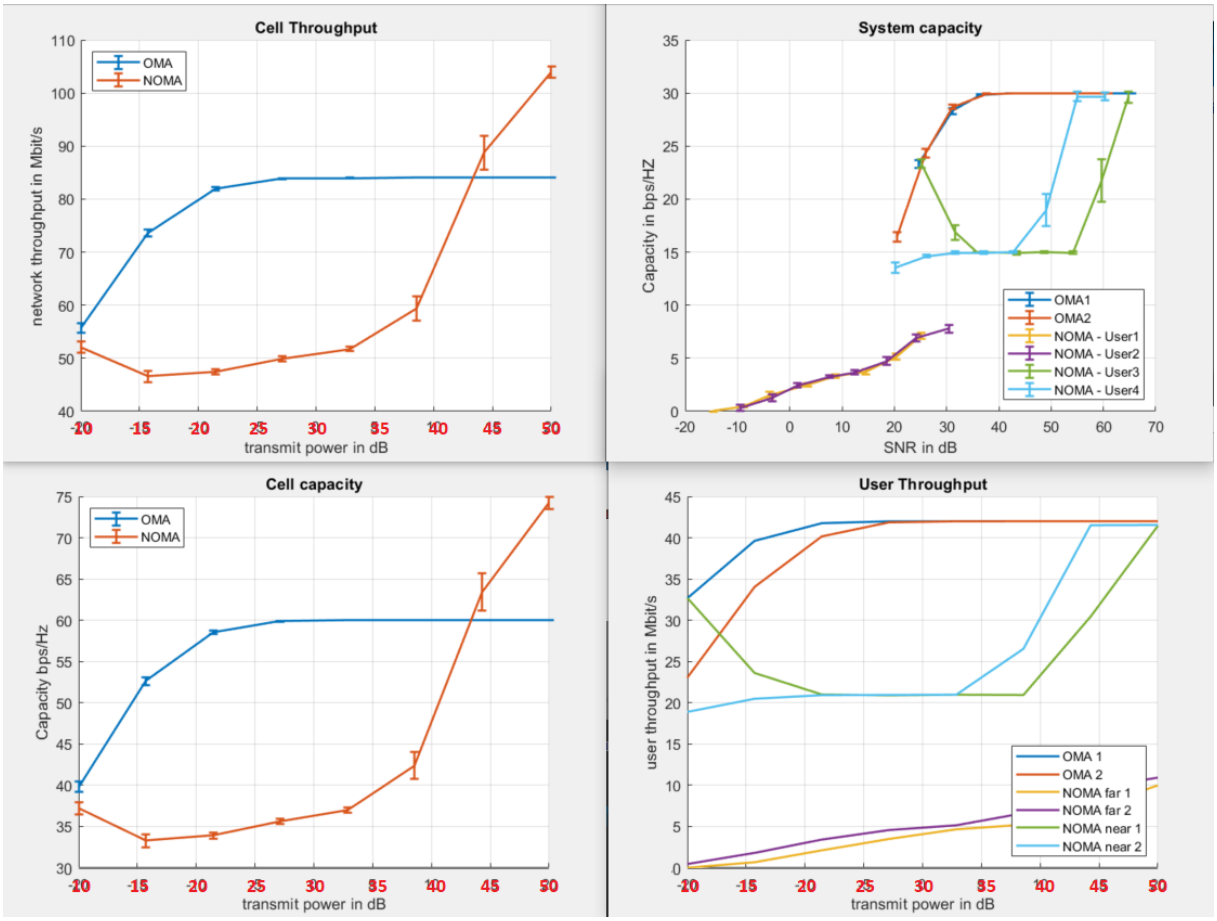


ZMIANA! --> NomaScheduler.m/linia 126: zmieniłem alpha dla MustIDx na **128/138**

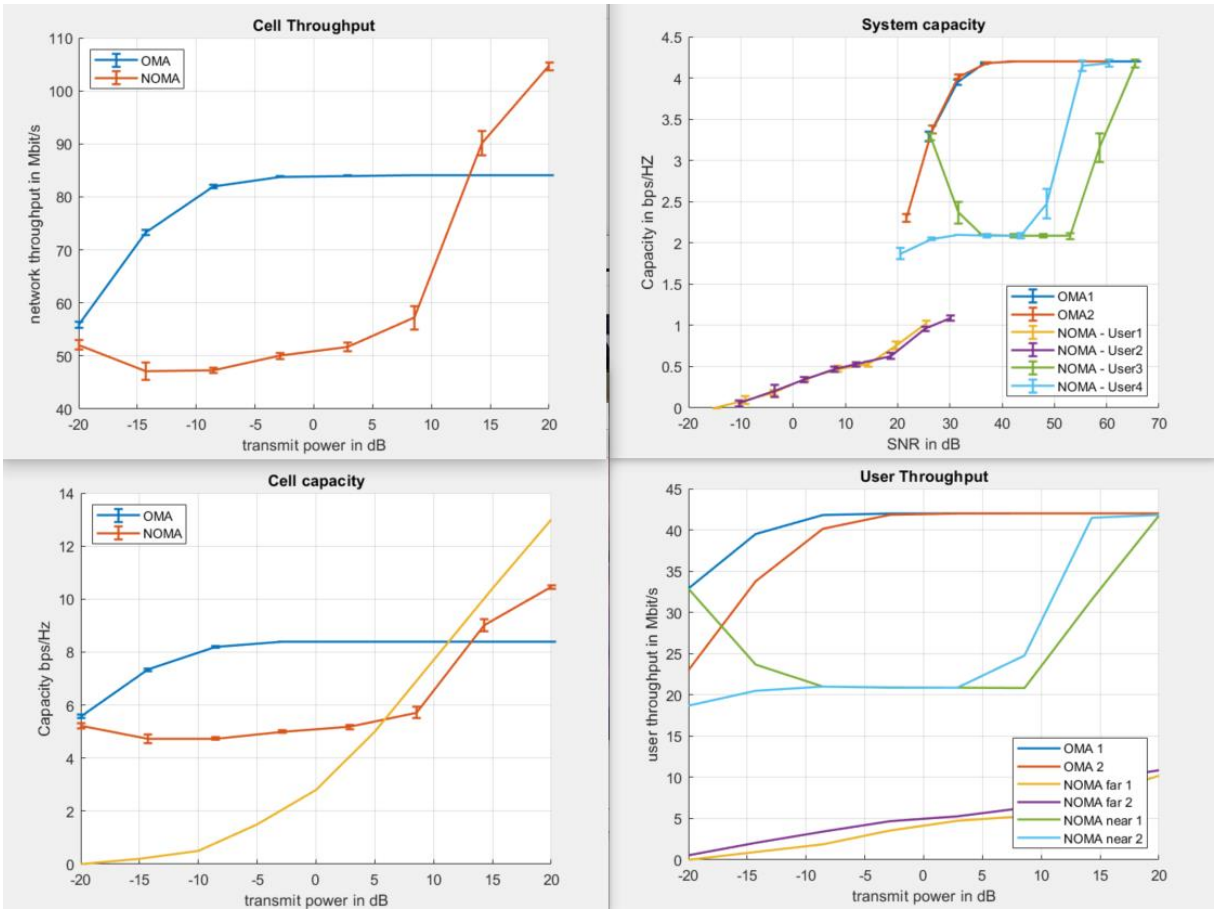


WNIOSKI: nie widać różnicy

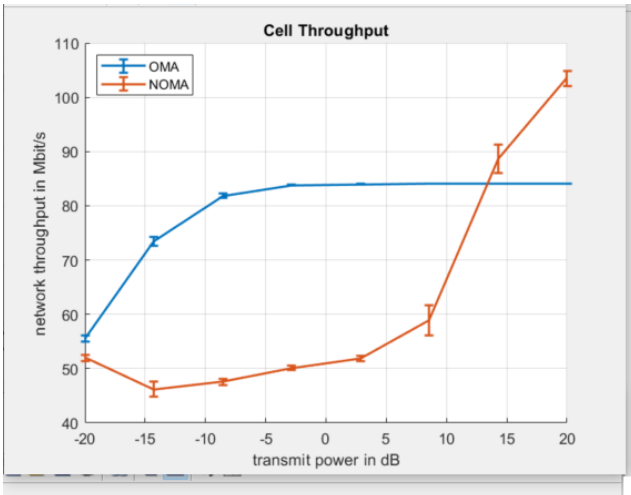
Wszystko jak wyżej tylko dopasowana skala (dBm - dB do Paper17)

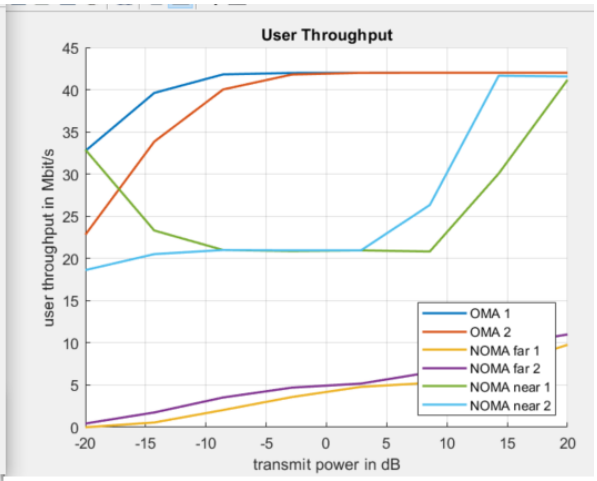


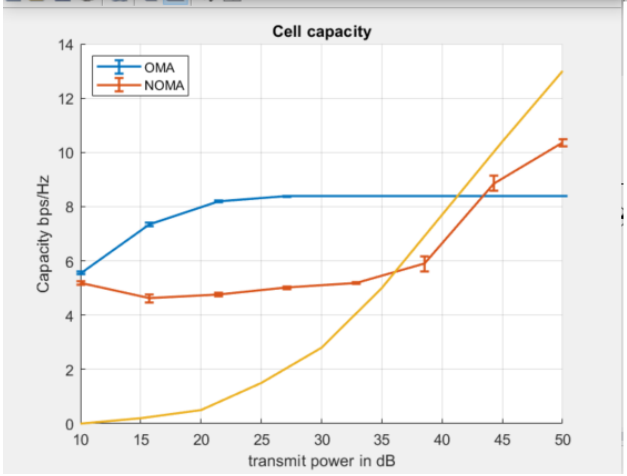
**Zmiana: a) wykres “cell capacity” oś X przesunięta o “-30”; b) alpha = 0.95 (NomaScheduler/powerShare) żeby być bliżej Paper17



UWAGA: skala na rysunku “Cell capacity” jest błąd!



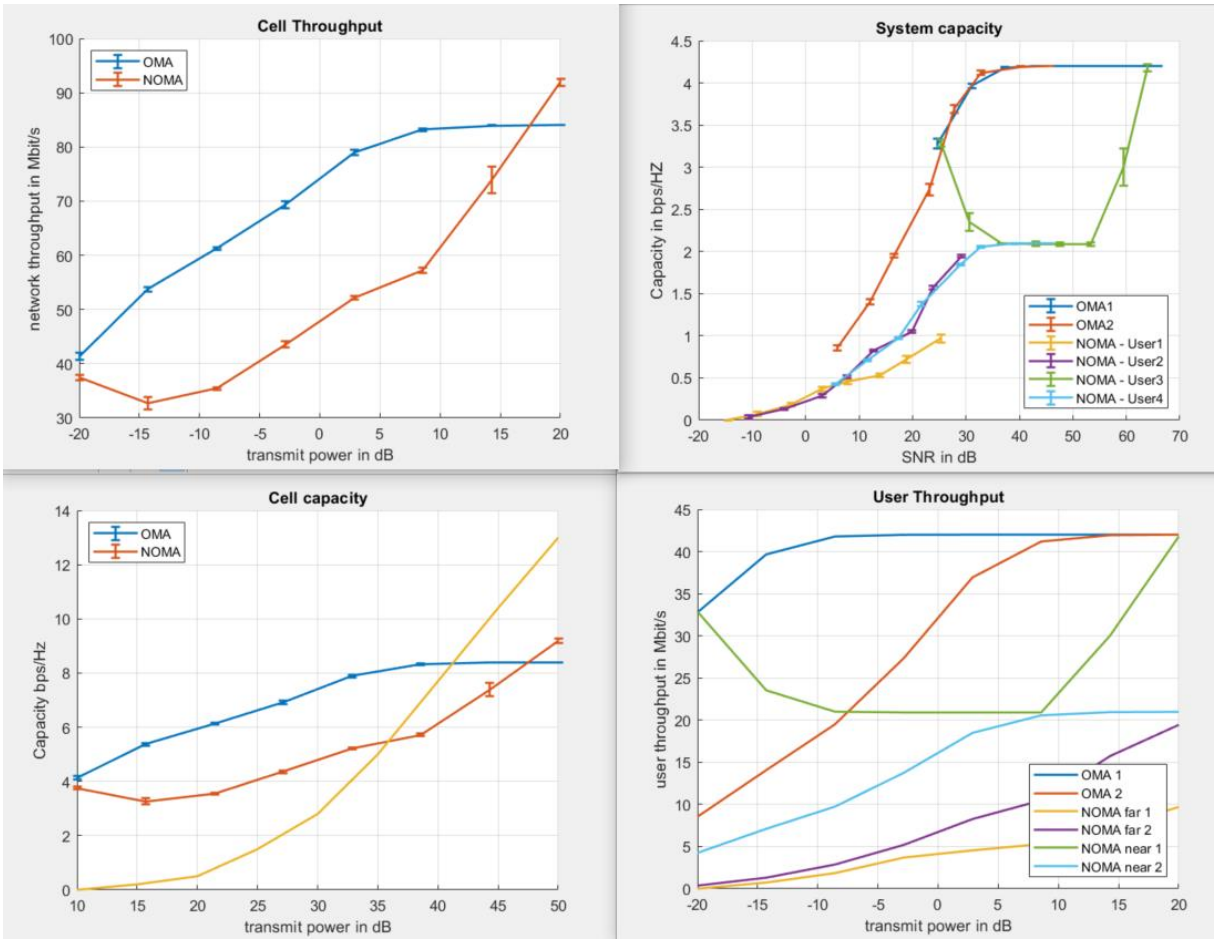




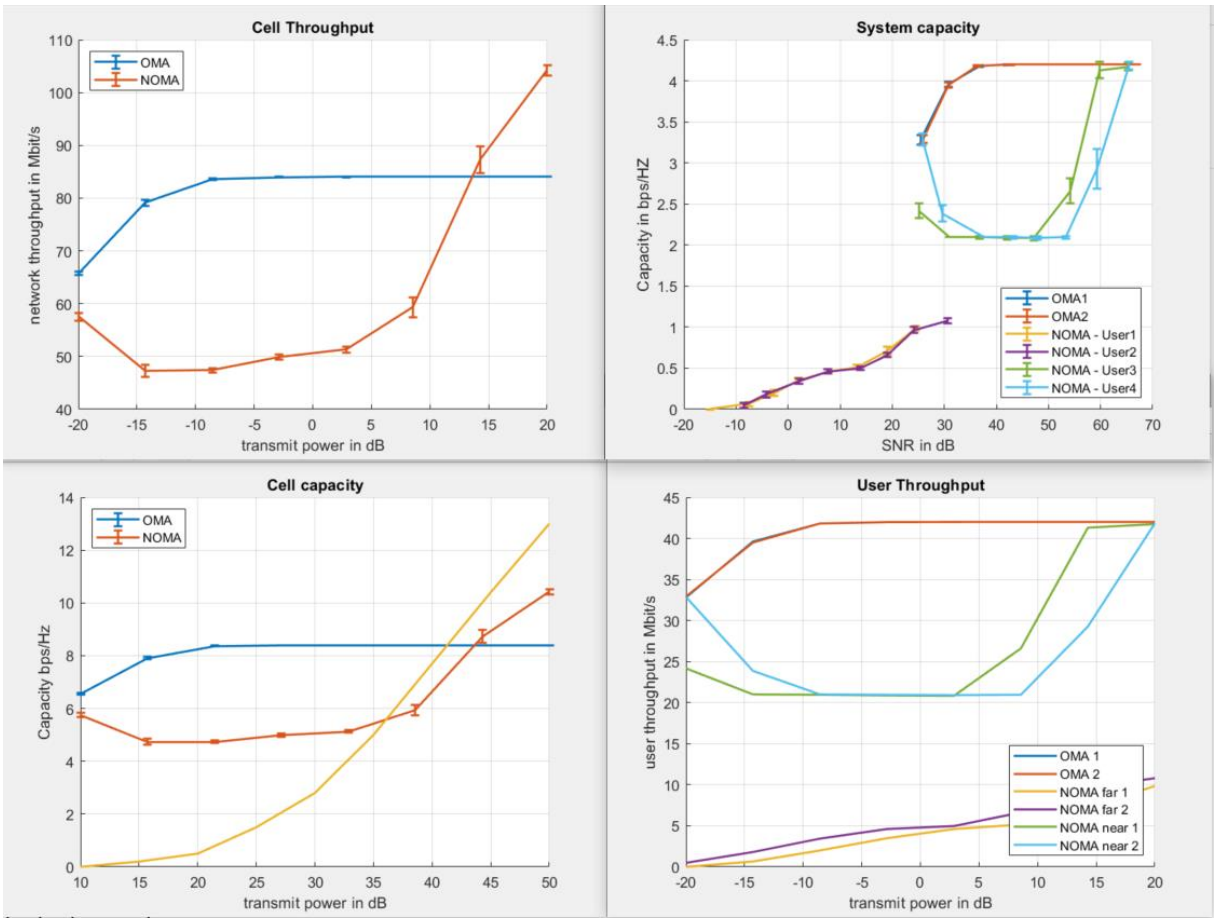
Powyżej obrazek “cell capacity” już z dobrą skalą.

Teraz rozpoczynamy śledzenie "co się dzieje" z userem "zielonym" w zakresie "skoków" przepływności.

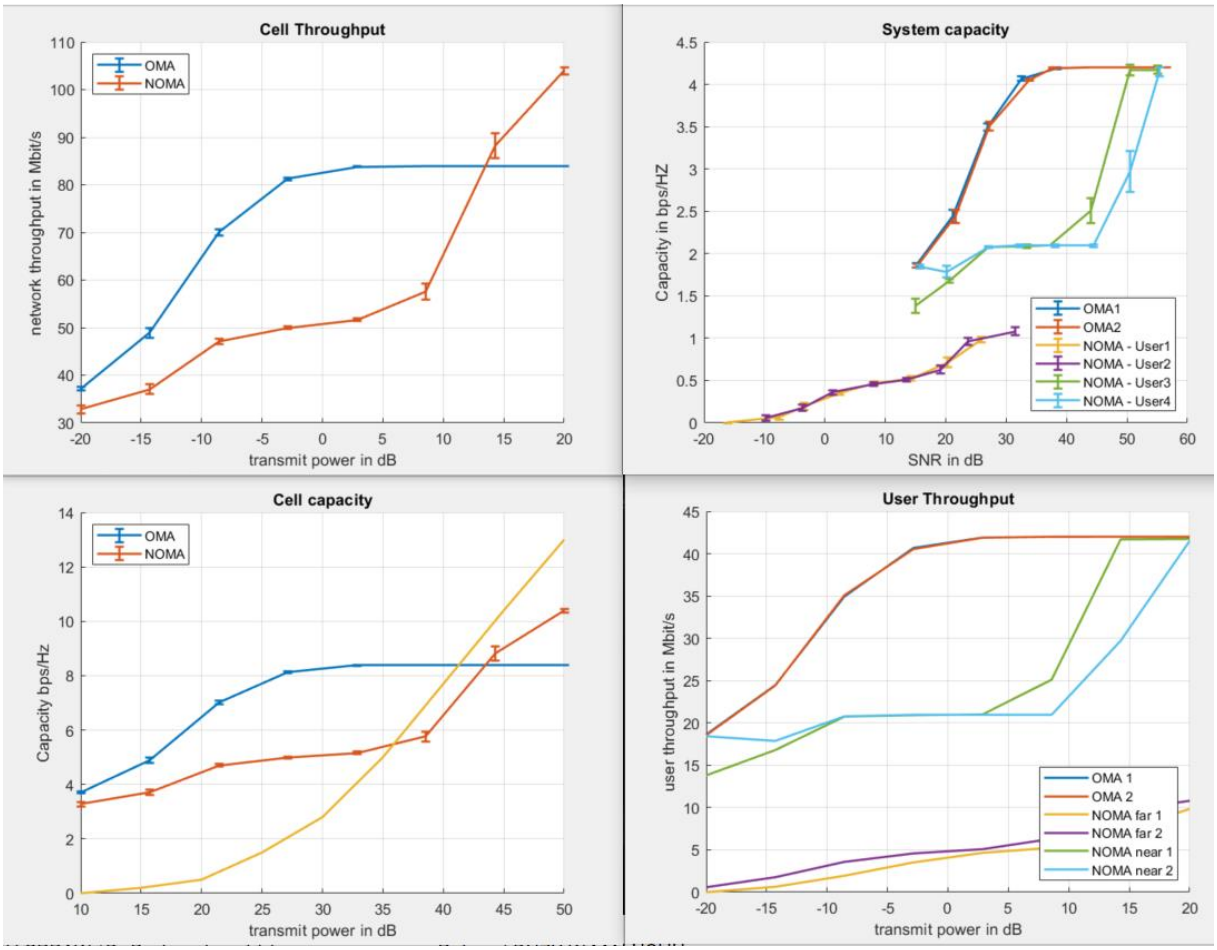
Scheduler: RoundRobin
User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 80dB PL; User_OMA2 = ~~85dB~~ PL 100dB



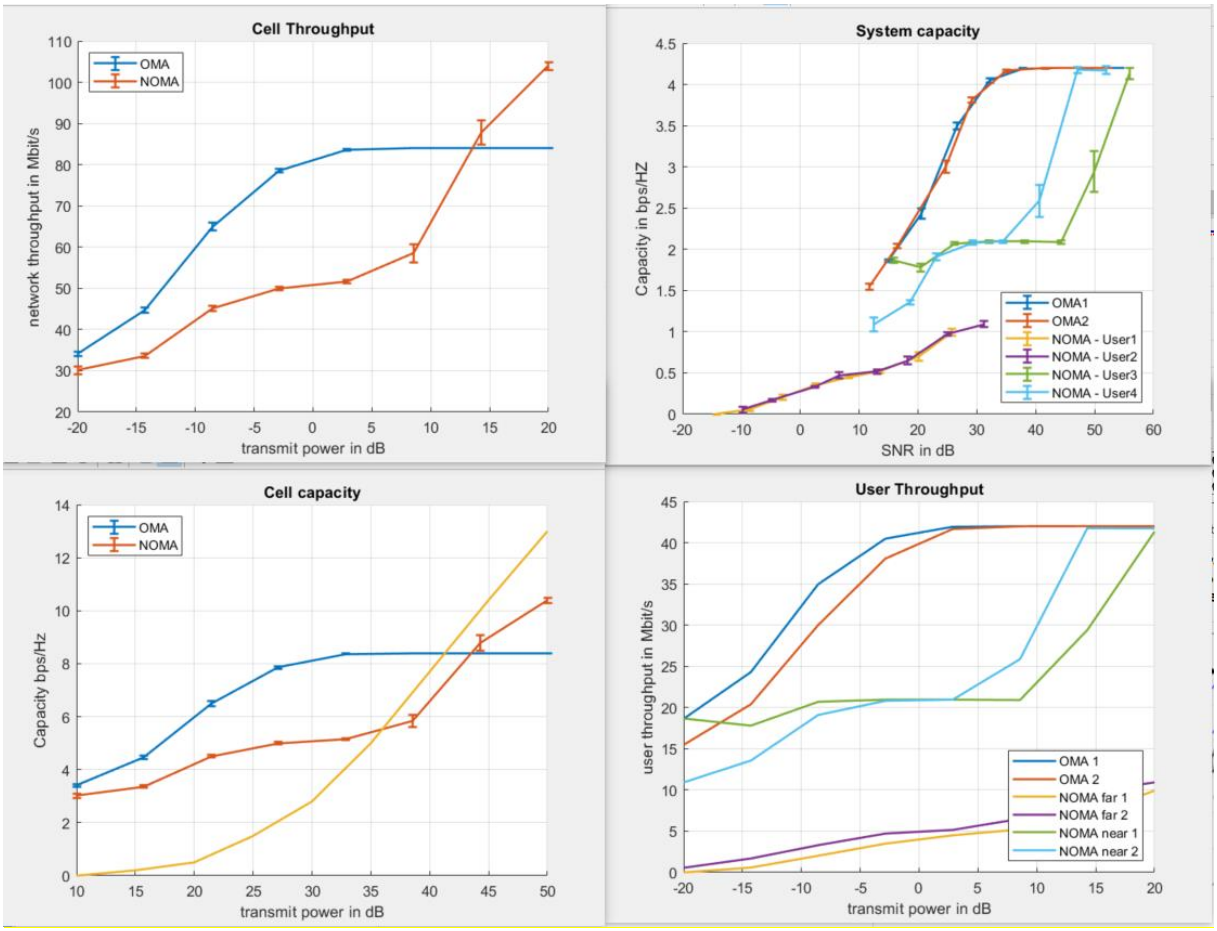
Scheduler: RoundRobin
User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 80dB PL; User_OMA2 = ~~85dB~~ PL ~~100dB~~ 80dB



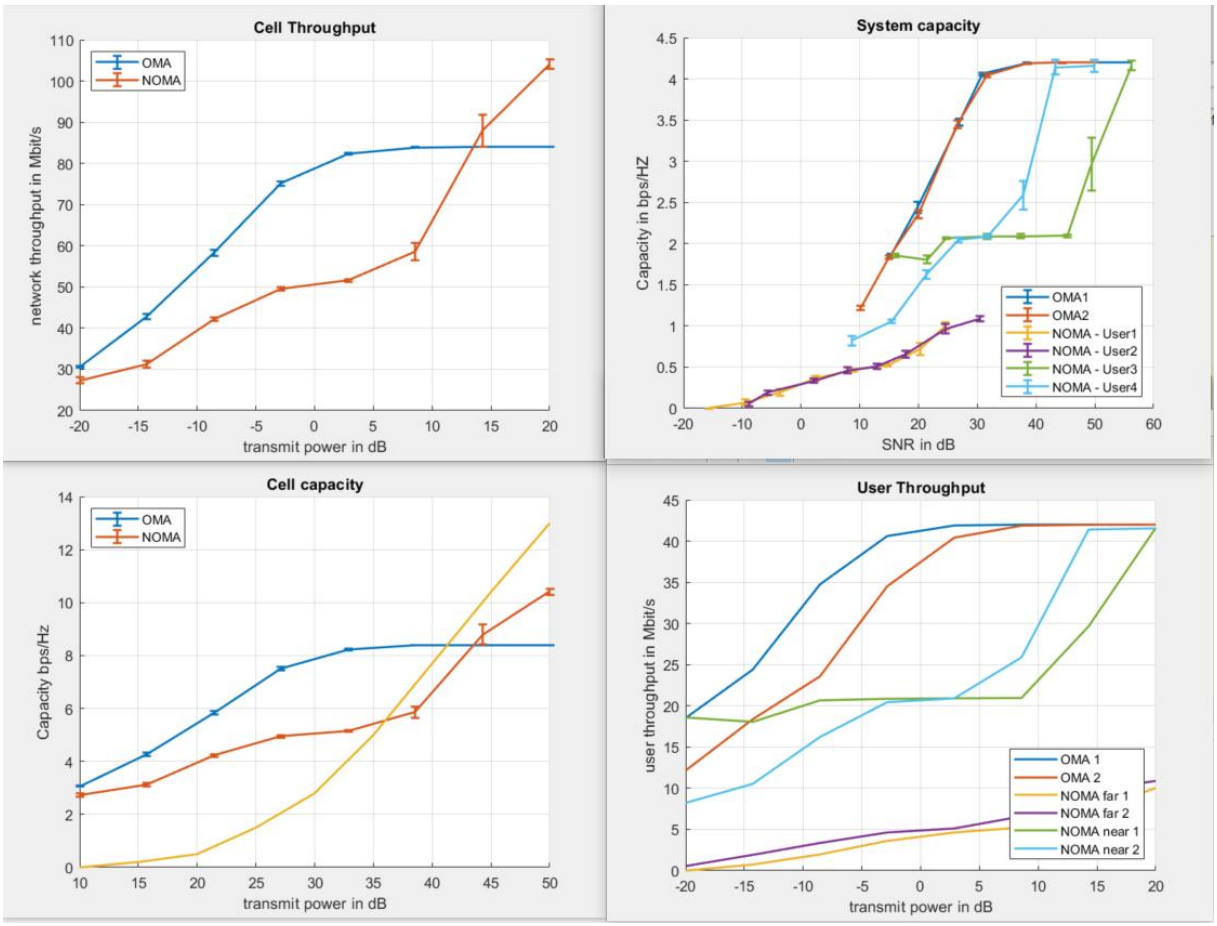
Scheduler: RoundRobin
User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 90dB PL; User_OMA2 = ~~85dB~~ PL ~~100dB~~ 90dB



Scheduler: RoundRobin
ChannelModel - TU
User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 90dB PL; User_OMA2 = ~~85dB~~ PL ~~100dB~~ 93dB



Scheduler: RoundRobin
ChannelModel - TU
User_NOMA1 = 120dB PL; User_NOMA2 = 115dB PL
User_OMA1 = 90dB PL; User_OMA2 = ~~85dB~~ PL ~~100dB~~ 96dB



Istotne obserwacje.

"For the two-user scenario, it has been demonstrated that SC is capable of achieving the capacity region of degraded Gaussian channels [50, 51, 52]. Furthermore, when the number of symbols becomes sufficiently large, i.e., when the number of symbols approaches infinity, SC ideally approaches the Shannon capacity [49]." **[4]**

"in [54], it was stated that SIC can achieve considerable better performance and lower hardware complexity than other interference cancellation techniques. In addition, it has been demonstrated in [55, 23] that SIC can approach the Shannon capacity in both single-antenna and MIMO systems." **[4]**

"In massive MIMO scenarios the non-ideal antenna placement at BS can lead to correlated paths between antennas – so the assumtion of i.i.d Rayleigh channels is too unrealistic". **[4]**

First, let $\mathbf{h}_u \in \mathbb{C}^{M\times 1}$ be the correlated complex channel response between the BS and the user u. Under this definition, the transmit spatial channel covariance matrix of user u can be obtained as $\mathbf{R}_u = \mathbb{E}[\mathbf{h}_u \mathbf{h}_u^H] \in \mathbb{C}^{M\times M}$, which has rank denoted by r. In simple terms, $\mathbf{R}_u$ is a positive semi-definite Hermitian matrix that informs how the transmit antennas are correlated with each other. In uncorrelated scenarios, $\mathbf{R}_u$ is equal to the identity matrix $\mathbf{I}_M$, which is full-rank, i.e., $r = M$. However, in highly correlated environments, due to redundant propagation paths, the resulting covariance matrices can exhibit a low-rank behavior, in which only a small portion of eigenvalues is nonzero. This implies that the radiated energy will, most likely, propagate through a reduced number of directions, corresponding to the dominant nonzero eigenmodes. In [68], it has been demonstrated that rank-deficient covariance matrices can impact both channel hardening and favorable propagation properties. In particular, more antennas are necessary to reach (asymptotically) channel hardening in rank-deficient channels, making correlation detrimental for this property. On the other hand, depending on the spatial

[4] - czyli że dla massive-MIMO, może performance bardzo zależeć od lokalizacji userów względem anten!

Literatura

[1] http://www.ijiee.org/papers/201-X2020.pdf
[2] https://www.researchgate.net/publication/335395546_An_Accurate_Approximation_of_Resource_Request_Distributions_in_Millimeter_Wave_3GPP_New_Radio_Systems
[3] https://arxiv.org/pdf/2001.10309.pdf
[4] https://drive.google.com/file/d/1aabzc27jmu3p9nbKzXFvffr3SZW_eP7q/view?usp=sharing

Aneks1 Wybrane modyfikacje kodu

Funkcja chukSimulation.m

W tej funkcji realizowane są wszystkie operacje związane z przewarzaniem poszczególnych ramek OFDMA. W ramach projektu zostały dopisane do oryginalnej wersji symulatora rozszerzenia które umożliwiają m.in. zarządzanie sesjami użytkowników.

[…]

```
                %RATfor5G --- konfiguracja userów
                %obj.users(2).setSessionTiming(1,obj.nSlots);

                % poniżej sprawdzamy czy ma być generowanie ruchu wg
                % rozkładu poisson'a czy nie.
                if(isempty(networkElements.ue.User.persistentPoisson(0,0))
|| networkElements.ue.User.persistentPoisson(0,0)==0) % zapisz info czy
symulacja z generowaniem ruchu poisson czy nie
                    poissonGenerator=0; % set to "0" if poisson is not
needed=0
                else
                    poissonGenerator =
networkElements.ue.User.persistentPoisson(0,0);
                    % tylk odczytaj co jest zapisane w zmiennej
                end
                sess = 1; % slot w którym dany user ma mieć aktywną sesję
(jeśli nie ma dla niego "poissonGenerator"
                result = -1;

                % RATfor5G --- tutaj ustawiamy bieżący slot DLA KAŻDEGO
                % usera.
                for zz=indexUserRoi
                    % ustawiamy nowy slot dla wszystkich userów OD RAZU =
przed wejściem do pętli
                    % bo inaczej nie będzie się zgadzało, i niektórzy
                    % userzy będą na slocie iSlot, a inni jeszcze na
                    % iSlot-1!
                    obj.users(zz).setSlotRATfor5G(iSlot); % ustaw w
obiekcie usera pamięć aktywnego slotu

                end

                % RATFor5G - admission control in ChunkSimulation
                %% UWAGA: można by zrobić optymalizację, że jeśli nie ma
sesji która zawiera ten slot, to w ogóle nie odpadalmy kodu poniżej
                for ii=indexUserRoi %
                    %obj.users(ii).setSlotRATfor5G(iSlot); % ustaw w
obiekcie usera pamięć aktywnego slotu

                    sesjaUsera_ii = ~obj.users(ii).activeSession;

                    if(poissonGenerator && sesjaUsera_ii)
```

```
                        %obj.users(ii).retrieveSessionOfUser(iSlot); %
update each user.id's session information (start, duration)

                       % CHECK if (~obj.users(ii).activeSession)
                                % WAŻNE: dzięki temu sprawdzeniu unikamy
                                % NADPISYWANIA sesji dla danego
obj.users(ii).id!

                                result =
obj.users(ii).retrieveSessionOfUser(iSlot);
                                aa = numel(result); % zapamiętaj LICZBE
SESJI która jest do utworzenia
                                usrTemp = [];       % tu zbierzemy
instancje klasy User, które mogą zostać wykorzystane do nowo-tworzonej sesji
                                if((aa==1&&result>1) || (aa>1))
                                    % if result>1 => means there was more
                                    % than one session in this slot
                                    % started, but as aa=1 one was already
                                    % created in retrieveSessionOfUser2
                                    cc=1;
                                    usrTemp = obj.findIdleUEs(aa);
                                    %% UWAGA: tutaj byłem!!!!
                                    % usrTemp = obj.findIdleUEs(aa);

                                    while(cc < aa+1)
                                        % this session number "1" is
already created inside object
                                        %if (cc==1) cc=cc+1; continue; end
                                        % OLD APPROACH if
(obj.users(cc).activeSession) cc=cc+1; continue; end

                                        % odyzskaj tablicę "arrival" dla
obiektu
                                        % obj.users(cc), po to, zeby ją
przekazać
                                        % do funkcji aktywującej sesję
danego usera
                                        % ----- tmp =
find(obj.users(cc).arrival.class == obj.users(cc).NEWmatrix(result(cc),3));
                                        % aktywuj sesję "równoległych
userów"
                                        % których numery sesji są
zapamiętane w tablicy
                                        % "result"

                                        % a =
usrTemp.retrieveSessionOfUser(xxx);

usrTemp(cc).activateUserSession(result(cc), iSlot);

%usrTemp(cc).newSessionInfo(result, result(cc),iSlot);
                                         %usrTemp(cc).newSession=1;
```

```
%usrTemp(cc).lastSession=result(cc);
                                        cc=cc+1; % następny user, if
needed
                                    end
                                end
                            elseif(isempty(obj.users(ii).lastSession))
                                    % Here we want to deal with user who has
no
                                    % poisson arrivals/departures defined.
                                    % So we set obj.START=sess for this user
session
                                    % CAUTION: default settings of
START=nSlots in
                                    % order to "inform" the functions that
"this
                                    % user is not used now" for running
session


obj.users(ii).setSessionTiming(sess,obj.nSlots); % ustaw sesję

                                if (sess == iSlot) % stworzona sesja jest
gotowa do aktywowania w tym slocie
                                    result = 1;
                                    obj.users(ii).lastSession = sess;
                                    obj.users(ii).activeSession = 1;
                                end
                                if (sess > iSlot) % slot do aktywowania tej
sesji jest ustawiony w przyszłości
                                    result = 0;
                                    % nie ustawiamy "lastSession" skoro
początek
                                    % sesji (tj. obj.START) jest w
przyszłości, to
                                end % sesja jest stworzona "na później"

                        %CHECK else
                            %CHECK obj.users(ii).newSessionInfo([],
[],iSlot);
                                    % wyświetl informację o stanie sesji dla
                                    % tego usera - jego sesja trwa!
                        %CHECK end


                    end % the above code is executed ONLY if user(ii)
session is NOT active. If it is active we need to skip to next user = ii+1

                    %jeśli ii-ty user ma aktywną sesję to sprawdźmy czy
                    %za następny slot ona sie nie musi zakończyć
                    if(obj.users(ii).activeSession)
                          sesja = obj.users(ii).activeSession;
```

```
                            obj.users(ii).cleanFinishedSession(0,sesja,
iSlot);
                        end
                end

                for iii=indexUserRoi
                    % perform admission control if the "result" is positive
                    % so we need to activate session of this user NOW.
                    sessionStartSlot = obj.users(iii).start;
                    if(sessionStartSlot == iSlot) &&
obj.users(iii).newSession
                        obj.performSessionCAC(obj.users(iii),iSlot); %
user(ii) has just now started new session
                        obj.admissionCounter = obj.admissionCounter + 1;
                        obj.users(iii).newSession = 0;
                    end

%obj.users(iii).newSessionInfo(obj.users(iii).lastSession,
obj.users(iii).activeSession,iSlot);
                end
```