Uniwersytet Warszawski
Wydział Fizyki

Raport.
Zredukowany model wzrostu kanałów i struktur dendrytycznych.
część 5.

Oleg Kmechak

Warszawa, Kwiecień 2019

Treść

Wstęp

Model wzrostu struktur dendrytycznych jest zależny od kilku parametrów, a program dla obliczeń numerycznych ma ich jeszcze więcej.
Dla parametrów modeli ważne jest ich dopasowanie do prawdziwych struktur dendrytycznych, znalezienie parametru *n* ,który odpowiada za prędkość wzrostu i ***b**crit* - próg bifurkacji [1].
Parametry numeryczne mają dwa więzy - czas trwania symulacji który trzeba zminimalizować i zbieżność wyniku numerycznego do rozwiązania analitycznego.

Wzrost odwrotny

Przy symulacji wzrostu sieci jest uwzględniane pole wokół wierzchołków, *n* i ***b**crit,* z czego znamy kierunek i długość następnego kroku wzrostu [1].
Jednak gdy mamy już wygenerowaną przez przyrodę sieć to parametry *n* i ***b**crit* nie są znane. Jednym ze sposobów ich znalezienia jest “wzrost odwrotny”. Metoda ta polega na tym, że korzystamy z tego samego wzoru na obliczanie odległości kroku, ale przemieszczamy wierzchołek w kierunku wstecznym i powtarzamy daną procedurę dopóki nie “zniknie” cała sieć.

Wzrost odwrotny był zrealizowany w programie symulacji. Żeby przetestować działanie metody można skorzystać z następnych komend:

./riversim -n 100 -o simdata

dana komenda wykona wzrost sieci w 100 krokach i wygeneruje plik simdata.json, który mieści wszystkie parametry modelu i parametry numeryczne, granice regionu i warunki na brzegach, a głównie wygenerowaną sieć.

./riversim --simulation-type=1 -n 150 -o reversesimdata simdata.json

dana komenda z pomocą opcji “simulation-type” będzie symulować wzrost odwrotny. Obowiązkowym parametrem w tym przypadku jest plik wejściowy (simdata.json który

otrzymaliśmy w poprzednim kroku). Symulcja będzie generować plik *reversesimdata.json* (można wybrać dowolną inną nazwę). Plik JSON mieści obiekt GeometryDifference, który z kolei mieści takie dane jak odległość, kąty pomiędzy wierzchołkami i inne dane.

Geometria sieci i regionu może być dowolna, a nie tylko wygenerowana przez program. Wystarczy tylko dostarczyć taki sam format pliku json.

Przykład danych w pliku json:


```
"GeometryDifference": {
        "angles": [
            -2.1073424255447017e-08,
            -2.1073424255447017e-08
        ],
        "biffurcationDifference": [],
        "biffurcationValues": [
            0.6999393003425749,
            0.710922967721284,
        ],
        "description": "this structure holds info about backward river simulation.",
        "distances": [
            0.0,
            0.0,
        ]
    }
```


Gdzie *angles* - kąt pomiędzy wierzchołkami, *biffurcationDifference* - różnica w odległościach do punktu bifurkacji, *biffurcationValues* - wartości a3/a1, distances - *odległości* pomiędzy wierzchołkam.i

Parametry numeryczne. Zbieżność.

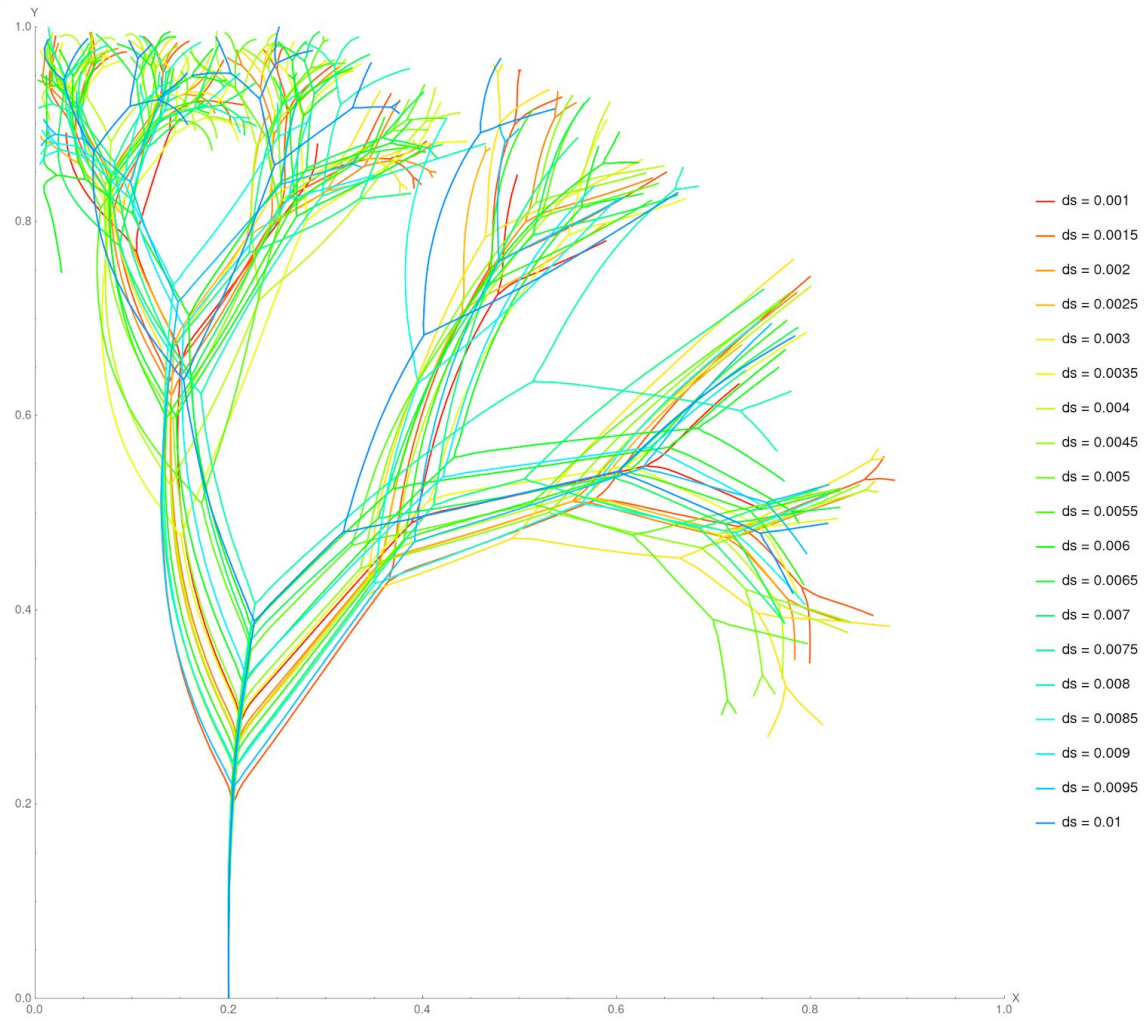
W porównaniu do parametrów modeli matematycznych, liczba parametrów numerycznych jest o wiele większa i jak się okazuje dopasowanie ich do wyników analitycznych nie jest proste.
Parametry numeryczne można podzielić na trzy rodzaje:

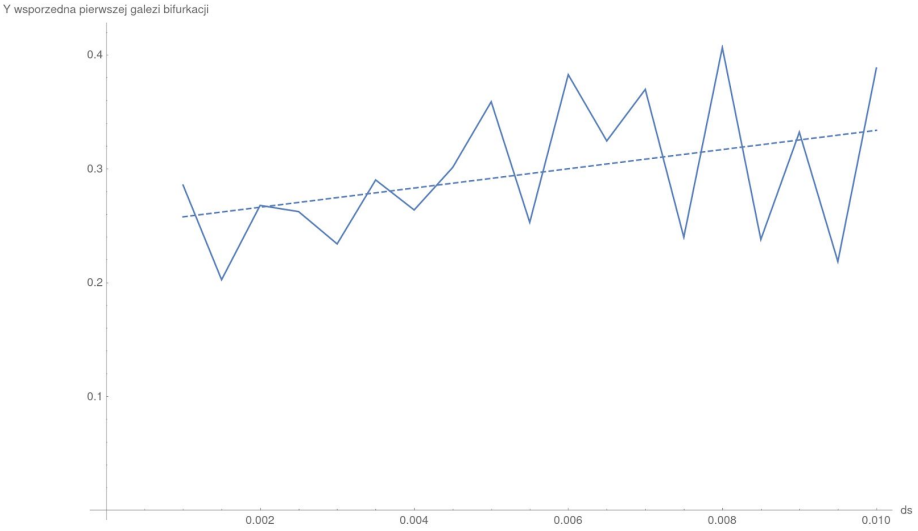
- Parametry całkowania w okolicy wierzchołka. Promień całkowania, parametry funkcji wagi.
- Parametry siatki(mesh). Min/maksymalny rozmiar siatki, parametry funkcji wagi.

- Parametry FEM(solver) podprogramów a dokładniej - parametry adaptywnej siatki.

Pierwszym ważnym warunkiem jest sprawdzenie zbieżności rozwiązania (wygenerowanej sieci) do pewnej granicy.

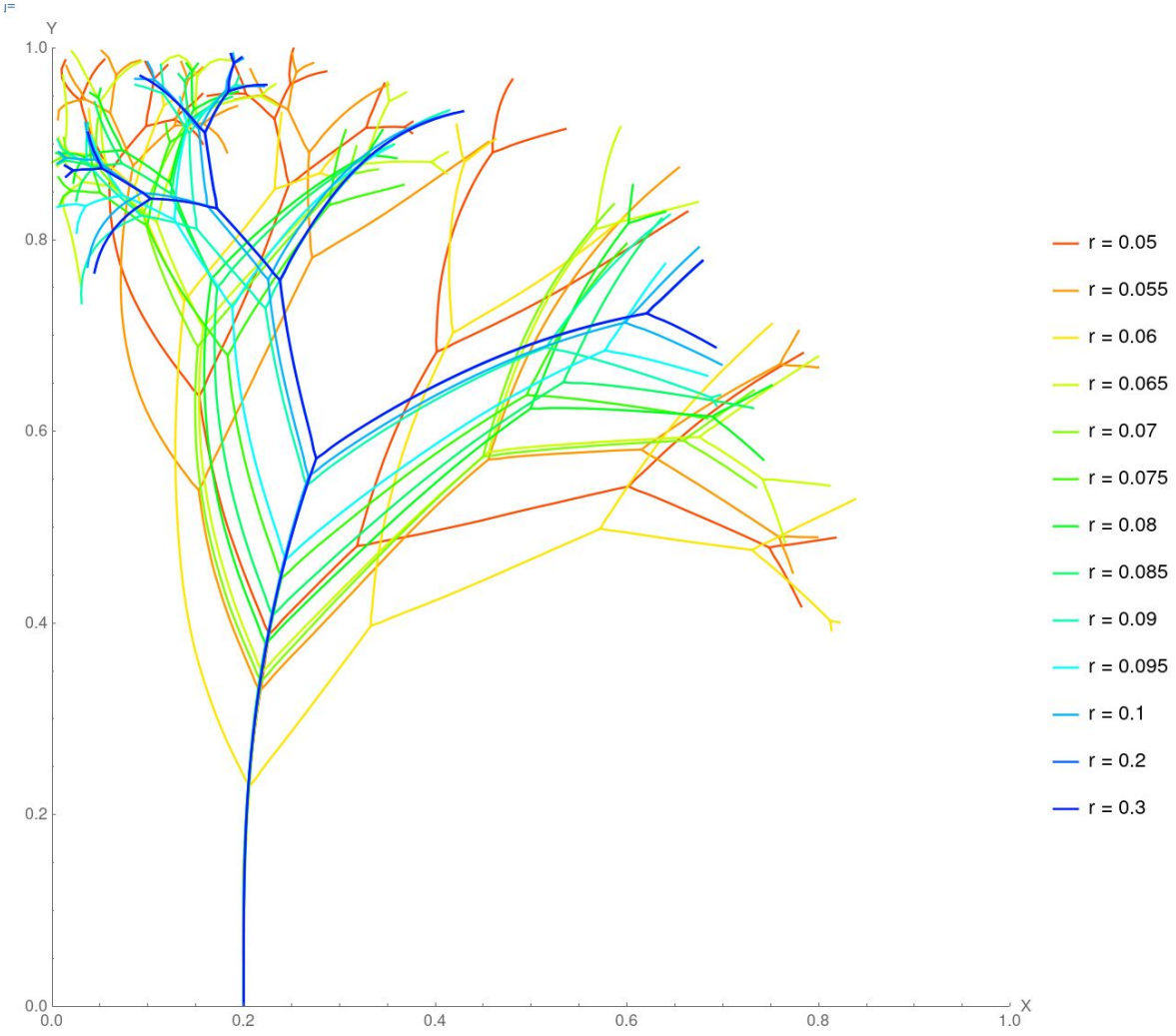
Wykresy sieci z parametrami ds od 0.01 do 0.001:

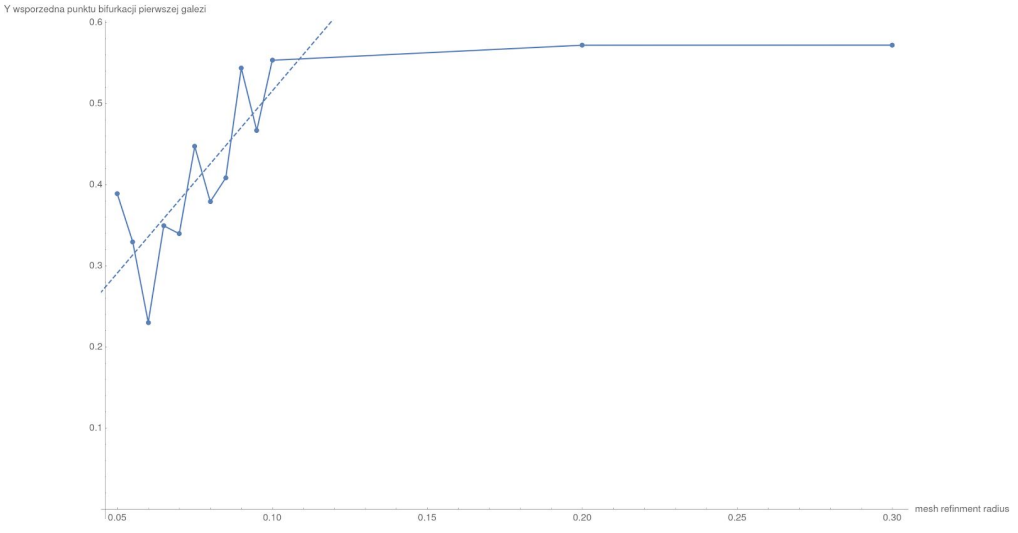




Jak widać na zmianę parametru **ds** model reaguje bardzo słabo.

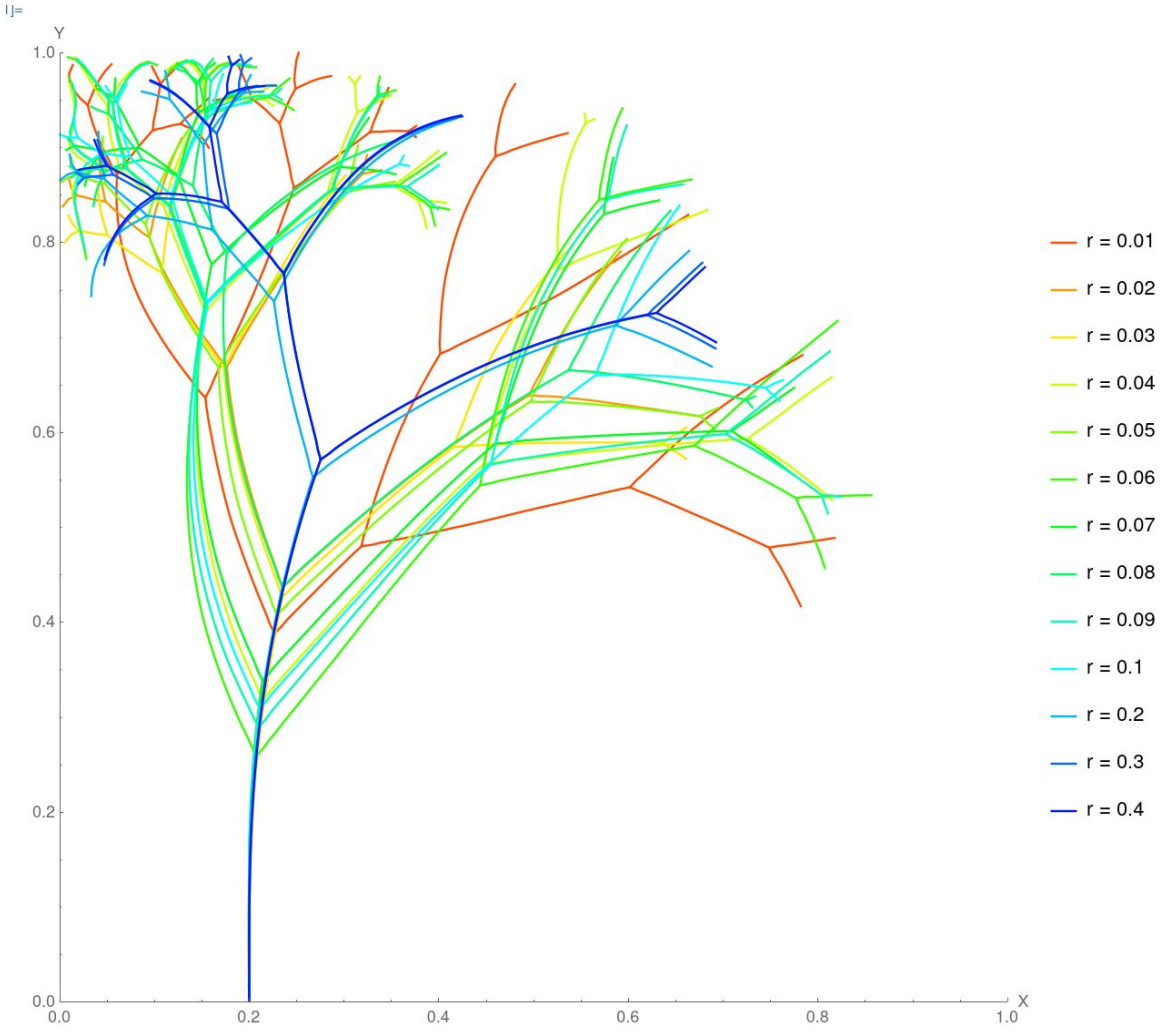
Wykresy z różnym zasięgiem zagęszczenia siatki meshu (promień od 0.05 do 0.3):

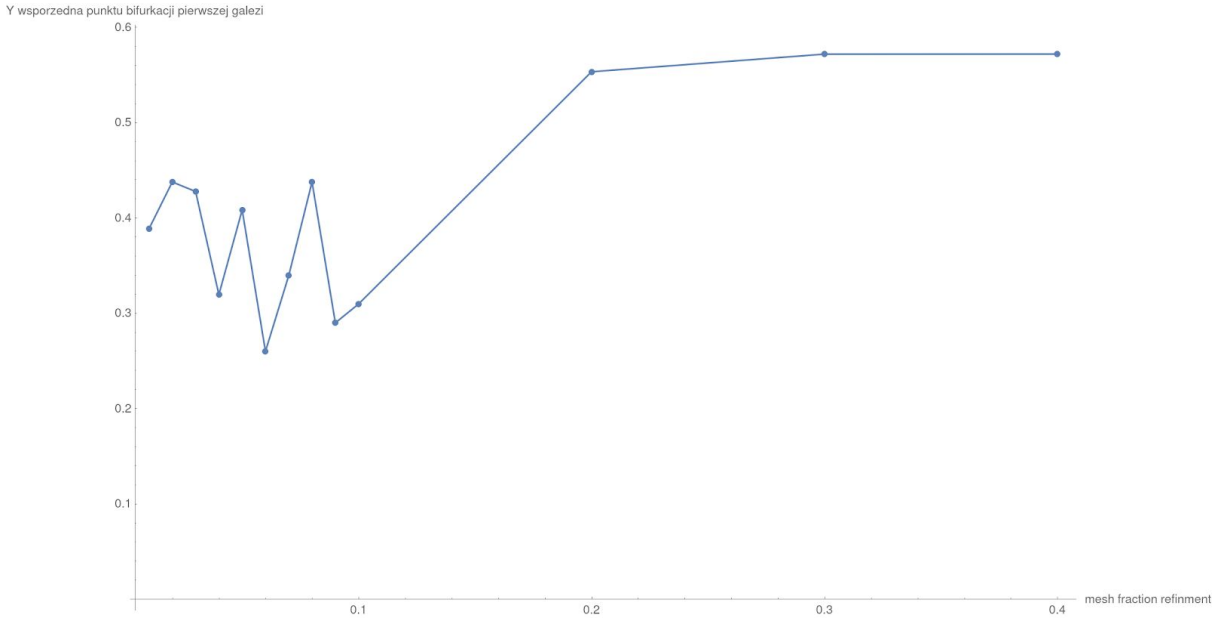




Na parametr *refinment-radius* model reaguje drastycznie i osiąga stabilność już powyżej znaczenia 0.1.

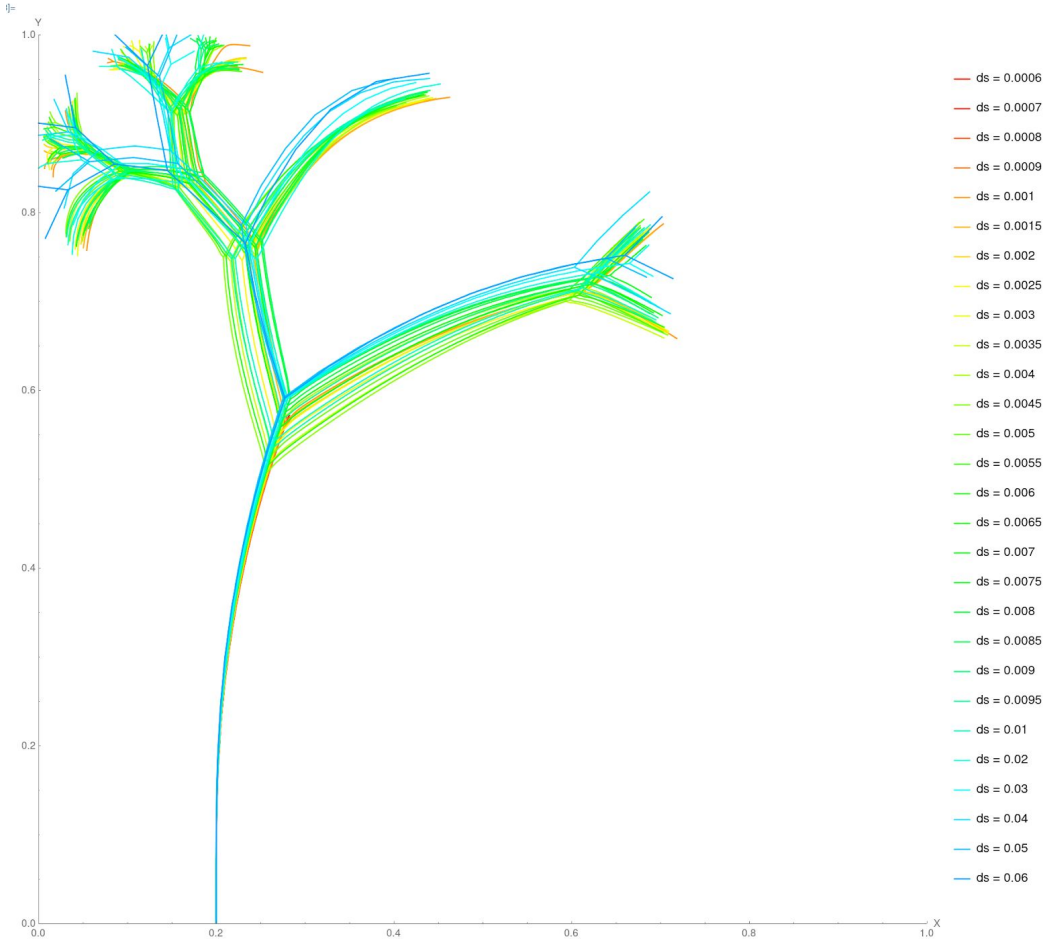
Innym parametrem związanym z siatka jest: *mesh-refinment-fraction* - który odpowiada za procent zmniejszanych elementów siatki za każdym krokiem. Wykres dla różnych znaczeń parametru związanego z adaptatywną siatką (mesh-ref-fraction) od 0.01 do 0.3:

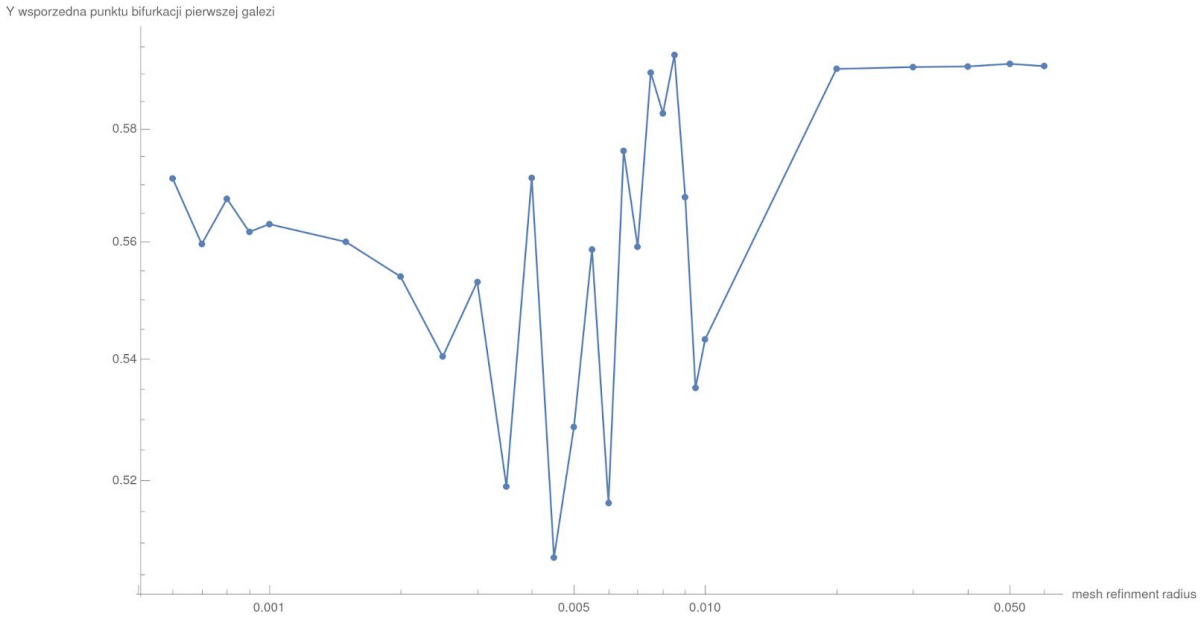




Jak i z parametrem *refinment-radius,* parameter *refinment-fraction* osiąga stabilność powyżej 0.2 znaczeń.

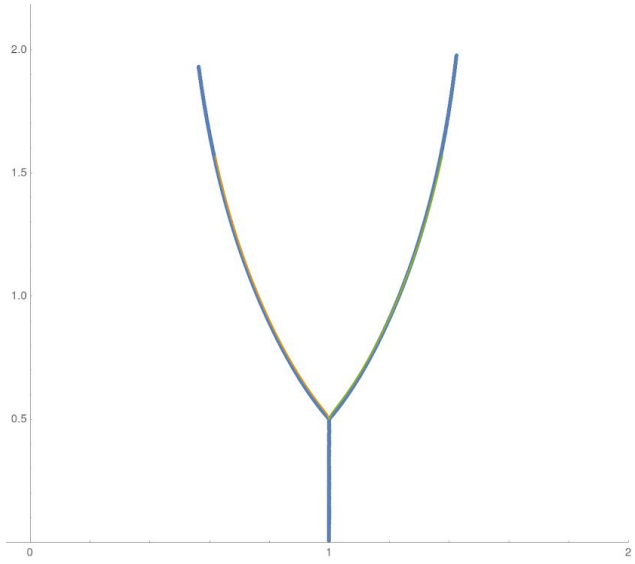
Biorąc pod uwagę powyższe znaczenia parametrów sprawdzimy ponownie jak reaguje model na zmianę parametra *ds* i czy ta zmiana dąży do granicy:



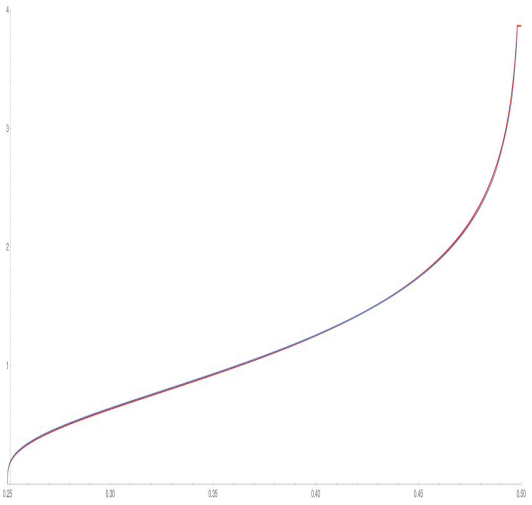


Przy dążeniu parametra *ds* da zera widać mniejszą chaotyczność, ale wciąż wynik nie dąży do granicy.

W drugim - granica powinna być taka sama jak analityczne rozwiązanie.



Porównanie w punkcie bifurkacji.



Porównanie wzrostu jednej gałęzi.

Skrypt w Mathematice.

Dla ułatwienia zarządzania symulacją i jej wynikami został stworzony skrypt pod adresem w repozytorium: */riversim/research/ExperimentRunner.nb*
W momencie napisania raportu, skrypt ma wpisane globalne ścieżki.

Zaimplementowane funkcje:

Importowanie:
OpenJSONDialog[] - dialog dla importowanie jednej albo wicej sciezek do plików symulacji.
OpenJSONData[] - dialog dla importowanie danych symulacji.

Wykresy:
plotSimData[data, color, legend] - wykres danych symulacji.
plotJSON[] - wykres pliku symulacji.

Symulacja:
ModelJSONStructure[] - objekt ktory jest wykorzystywany dla parametryzacji symulacji.
RunSimulation[numOfSteps, modelParams, outputName] - synchronne uruchamianie symulacji z srodowiska Mathematica.