Uniwersytet Warszawski
Wydział Fizyki

Raport.
Zredukowany model wzrostu kanałów i struktur dendrytycznych.
część 2.

Oleg Kmechak

Warszawa, Styczeń 2019

Treść

Wstęp

W tej pracy rozpatrujemy model wzrostu sieci dendrytycznej za pomocą wykorzystania metod numerycznych. Zakładamy, że sieć rośnie z prędkością proporcjonalną do gradientu pewnego pola spełniającego równanie Poissona w obszarze na zewnątrz sieci.
Do rozwiązywania takiego układu są stosowane Metody Numeryczne Elementów Skończonych.

Parametryzacja rozmiaru siatki

Obecnie stosowana jest metoda adaptacji siatki (adaptive mesh refinement), która działa iteracyjnie przy rozwiązaniu równań. Wiadomo jednak, że wielkie błędy numeryczne pojawiają się w miejscach szybkiej zmiany pola.
Miejsca szybkiej zmiany pola są z kolei na czubkach dendrytów.
Dlatego właśnie wokół czubków będziemy parametryzować(zagęszczać) rozmiar siatki.

1.Triangle

Dla kontroli rozmiaru siatki Triangle proponuje następne opcje:
- Dwa parametry: maksymalny rozmiar elementu(A) i minimalny kąt trójkąta(q).
- Zdefiniowanie funkcji `triunsuitable()` w pliku triangle.c

Ale obydwa sposoby generują regularny mesh.

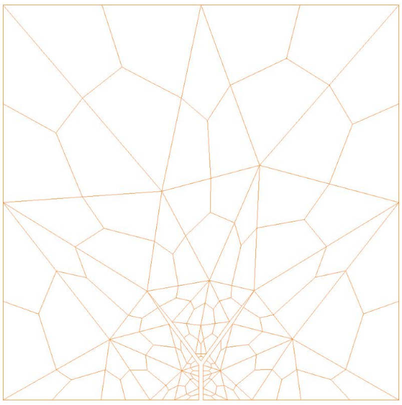
Przykład:
Rozpatrzymy następną komendę:
./riversim -g 2 -s 0 -Z 1 --steps 1 --ds 0.03 --eps 0.003 -A 1 -q 0

g(geom-type) - początkowy typ geometrii, który jest wykorzystany dla testowania
(0 - kwadrat, 1 - kwadrat i jedna gałąź, 2 - drzewo, które i jest na następnym wykresie)
s(simulation) - włączamy albo podłączamy moduł symulacji w programie
steps - ilość kroków symulacji

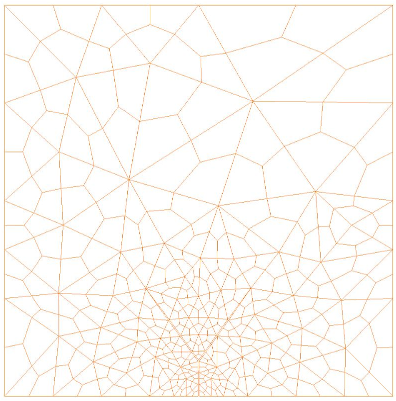
ds - parametr długości wzrostu gałęzi za jeden krok symulacji.
eps - szerokość gałęzi.
A(mesh-max-area) - maksymalny rozmiar siatki
q(mesh-min-angle) - minimalny kąt trójkąta.

rozmiar: 1x1.

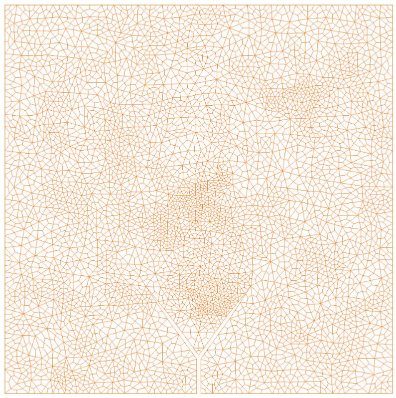
-A 1 -q 0



-A 1 -q 35



-A 0.001 -q 35



Deklaracja funkcji `triunsuitable()` była też rozpatrywana, ale żeby przekazać do funkcji współrzędne czubków, potrzebna jest większa zmiana w triangle.c.
To zaś wydaje się być bardzo czasochłonne.
Dlatego też korzystanie z `triunsuitable()` wymaga dalszej analizy.

2.GMSH

Geometria początkowa jest zawarta z obiekcie *Geometry.* Mieści ona w sobie współrzędne punktów, a także linie pomiędzy nimi. Po pewnym zmodyfikowaniu obiektu GMSH została dodana informacja o rozmiarze siatki wokół danego punktu. Też było nieoczywiste, jak przekazać informacje o brzegach (boundary_id). Jak okazało się do tego była potrzebna metoda *AddPhysicalGroup z API GMSH(a).*

Korzystanie z GMSH ma kilka swoich za i przeciw.

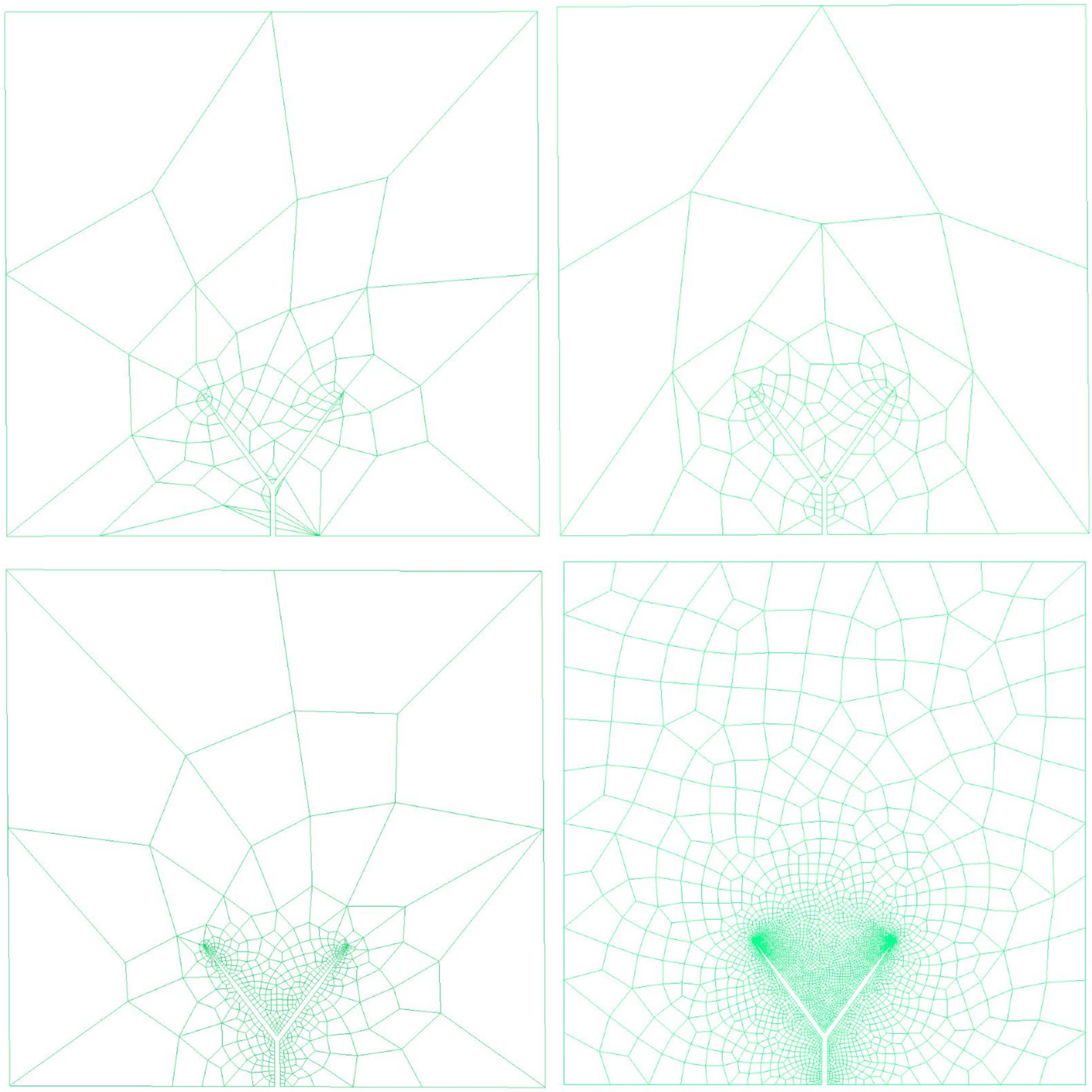
- Potrafi konwertować siatkę z trójkątów w prostokąty.
- Potrafi zagęszczać siatkę wokół danego punktu.

Ale

- Algorytm konwersji jest czuły na geometrie początkową. Żeby umożliwić jego wykorzystanie,, wcześniej była wprowadzona “szerokość” *eps* do linii dendryty. Możliwym rozwiązaniem tego problemu jest nie korzystanie z algorytmu konwersji GMSH, ale jego zamiana na to, co już jest w *Tethex.*

Przykład:
Wprowadzając następną komendę:
./riversim -g 2 -s 0 -Z 1 --steps 1 --ds 0.03 --eps 0.003 -G 1

G(use-gmsh) - zmienia generator siatki pomiędzy Triangle a Gmsh. 1 - korzystamy z Gmsh.



W tym przypadku zaimplementowano trzy rodzaje punktów którym odpowiadają trzy różne rozmiary siatki:

- Linie na krawędziach regionu - mają największy rozmiar siatki.
- Linie na krawędziach dendrytu.
- Punkty na czubkach dendrytów - mają najmniejszy rozmiar.

Całkowanie. Parametry wzrostu sieci dendrytów.

Prawa wzrostu pojedynczej gałęzi dendrytu jest bazowana na znaczeniach pola wokół czubka. A dokładnie na następnych równaniach:

$$a_1 = \frac{1}{\pi R^{1/2}} \int_0^{2\pi} \phi(R, \theta) \cos\frac{\theta}{2} \mathrm{d}\theta$$

$$a_2 = \frac{1}{\pi R} \int_0^{2\pi} \phi(R, \theta) \sin\theta \mathrm{d}\theta$$

$$a_3 = \frac{1}{\pi R^{3/2}} \int_0^{2\pi} \phi(R, \theta) \cos\frac{3\theta}{2} \mathrm{d}\theta$$

Gdzie R - odległość do czubka,

Θ - Kąt pomiędzy kierunkiem czubka, a punktem,

Φ - pole.

Z innej strony Deal.II proponuje *FEValues* klasę dla dostępu do rozwiązania.

1.Całkowanie

Obliczanie całki wokół czubka wygląda w następny sposób:

- Iteracja po całej siatce:
- Inicjalizacja *FEValues* do elementu siatki, który jest aktywny w danym momencie.
- Preinicializacja własności siatki, takie jak punkty kwadratury, czynnik Jacobiego, znaczenia pola w punktach kwadratury i inne.
- Iteracja po punktach kwadratury.
- Całkowanie, jako sumowanie po koniecznych punktach.

Szczegóły implementacji znajdują się w klasie Solver, metoda integrate().

Pierwsze kroki symulacji wzrostu.

Model sieci dendrytów składa się z kilku reguł wzrostu i iteracyjnego ich stosowania. Jest to nowym wyzwaniem dla programu.

1.Klasa: PhysModel

Klasa *PhysModel* enkapsuluje w sobie wszystkie informacje związane z fizyczną stroną zjawiska. Co w praktycznym wymiarze jest bardzo wygodne. Na danym etapie ma realizowane funkcje dla obliczania a1, a2, a3 parametrów wokół czubka.

2. Problemy i wyzwania

Iteracyjność, powoduje nowe błędy, na przykład takie, jak brak pamięci na komputerze, albo nieprawidłowa inicjalizacja i deinicjalizacja różnych obiektów w programie.

Z tego powodu była zrobiona rewizja programu na korzystanie z dynamicznej pamięci i jej prawidłowa deinicjalizacja. Ważnym zagadnieniem jest podanie argumentów do funkcji za wskaźnikiem, a nie po znaczeniu, gdzie tylko to jest możliwe.

Komunikacja pomiędzy obiektem *tethex:Mesh* a *Solver* jest nadal przez plik. Problem z przekazaniem boundary_id jest nadal aktualny.

W danym momencie są jeszcze problemy z obliczaniem następnego punktu wzrostu, ale na przykład pomijając ten punkt na inny, ale dokonując wszystkich obliczeń(generacja siatki, solver, całki) program działał powyżej 8 tys kroków.

Przyklad:
./riversim -g 1 -s 1 --steps 10000 --r 2 --ds 0.001 --eps -.000001 -dx 0.3 -q 30
Daje następne wyniki(krok symulacji ~248):