Uniwersytet Warszawski
Wydział Fizyki

Raport.
Zredukowany model wzrostu kanałów i struktur dendrytycznych.
część 1.

Oleg Kmechak

Warszawa, Grudzień 2018

Treść

- Wstęp
- Biblioteki
    - Jakiego języka użyć?
    - FEM
    - Generacja Siatki
    - Tethex
    - Boost
- Struktura programu
    - Struktura plików
    - Klasy i ich zależności
- Instalacja i korzystanie z programu
    - Zależność
    - Kompilacja
    - Opcje programu
- Pierwsze wyniki
    - Adaptywna siatka, warunki Laplace'a
    - Warunki Poissona

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.

Biblioteki

Opis wykorzystanych bibliotek i języków w programie, motywacja ich wyboru i krótki opis ich cech i zalet.

1.Jakiego języka do programowania użyć?

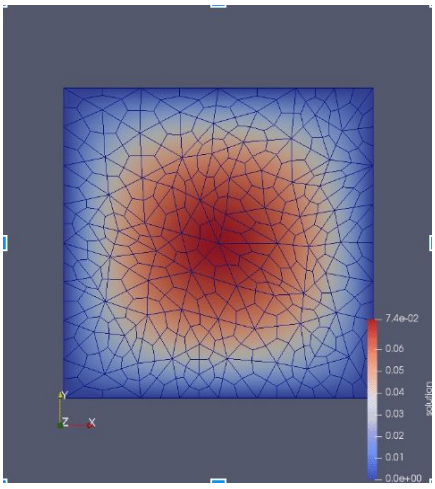
W tym zagadnieniu są intensywnie stosowane metody numeryczne. I już na tym etapie powstają następujące pytania:

- Jakiego języka użyć?
- Z jakich bibliotek numerycznych można skorzystać w danym języku?

Przy odpowiedzi na pierwsze pytanie kierowaliśmy się przede wszystkim prędkością działania programu, co spowodowało wybór C++. Odpowiedź na drugie pytanie była konsekwencją pierwszego. Większość programów numerycznych jest zorientowana na graficzny interfejs i jeżeli i mają C++ API, to słabo udokumentowany. W większości przypadków po prostu go nie ma. Jest też część bibliotek z otwartym źródłowym kodem, które promują własny scryptowy język do pracy. Z tego całego mnóstwa jedynie Deal.II jest całkowicie zorientowany na wykorzystanie w programie C++ i ma wielką dokumentację, ilość przykładów i aktywną liczbę użytkowników.

2.FEM

Rozpatrywany był również Python i biblioteka FEniCS. FEniCS w tym czasie jest bazowany na DOLFIN(fem biblioteka), która wówczas ma C++ API. Ale problemem okazała się dokumentacja której prawie nie było i nie było udokumentowanej możliwości jak zregenerować segment wbudowany w siatkę.



Na tym zdjęciu słabo widać ale w centrum jest przeprowadzona linia która jest potem wbudowana w siatkę, co nie każdy pakiet do generacji siatki potrafi

Dwa ważne aspekty w Deal.II:

- Potrafi generować tylko prostą regularną siatkę niektórych geometrycznych figur
- Elementami siatki mogą być tylko prostokąty

Z tego powodu jest potrzeba generowania siatki i znalezienia biblioteki dla tego.

3.Generacja siatki

Innymi słowy, jest potrzebny generator siatki. Przetestowano następujące programy: Triangle: potrafii generować Delaunay i Constrained Delaunay siatkę. Można również zadać ograniczenie na miarę powierzchni, kat trójkątów, albo zdefiniować własną funkcję która będzie odpowiedzialna za udoskonalenie elementów. W programie tym istnieje też możliwość zadania segmentów i dziur. Program działa wyłącznie w 2D.

Był również rozpatrywany TetGen, który jest następcą Triangle. Łączy w sobie wszystkie te same funkcje co Triangle, ale działa w 3D i jest napisany w C++, co ułatwia jego czytelność i modyfikacje w razie potrzeby(Triangle jest w C). Ale jak się okazało, TetGen generuje siatkę wyłącznie w 3D.

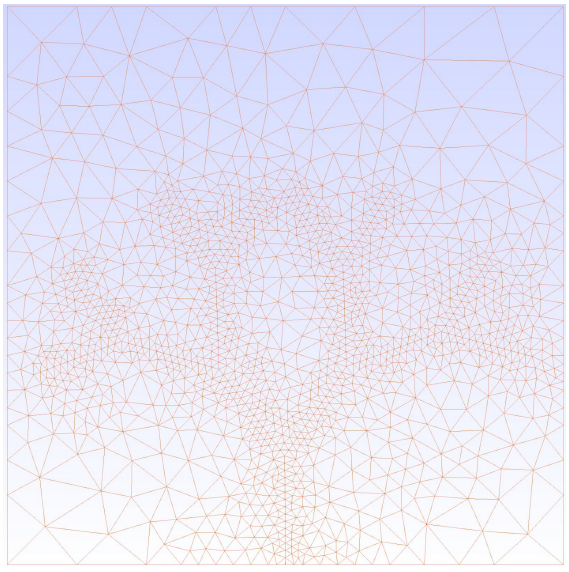
ViennaMesh, NetGen - mają możliwości podobne do Triangle, ale nie były przetestowane. Niektóre inne biblioteki dla pracy z grafiką komputerową: Libigl(bazowany na Triangle, TetGen, OpenGL i in.) i bardzo wielka biblioteka CGAL(Computational Graphical Algorithms Library) - która też ma generator siatki i mnóstwo innych procedur dla pracy z geometrią.

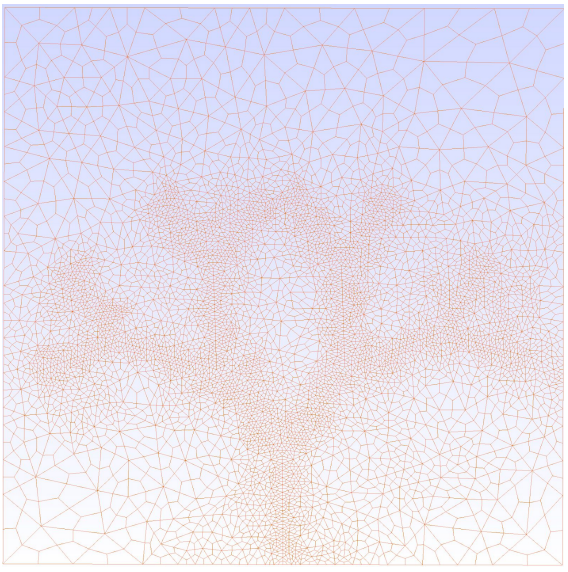
Przy pracy z siatką ważna jest jej wizualizacja. Jednym z najbardziej promowanych projektów z otwartym kodem jest - Gmsh. Gmsh - łączy w sobie TetGen, NetGen a takze CAD systemy(OpenCASCADE i in). I ma sporo możliwości dla wizualizacji siatki. Też ma bardzo ważną dla biblioteki Deal.II możliwość konwersji siatki z trójkątów do siatki z prostokątów. Ale ten algorytm pracuje dość niestabilnie i jest potrzebne dopasowanie parametrów rozmiaru siatki i pewne ograniczenie na geometrie. Też trzeba pamietać, że Deal.II podtrzymuje tylko 2 - 2.2msh format.

4.Tethex

Biblioteka dla konwersji trójkątów do prostokątów. Umożliwia połączyć Triangle i Deal.II, a też w perspektywie dla 3D możemy konwertować siatke TetGen(a) dla Deal.II tez. Też ma realizację kilku klas które zostały adaptowane dla programu.

Przed konwersją: Po:





5.Boost

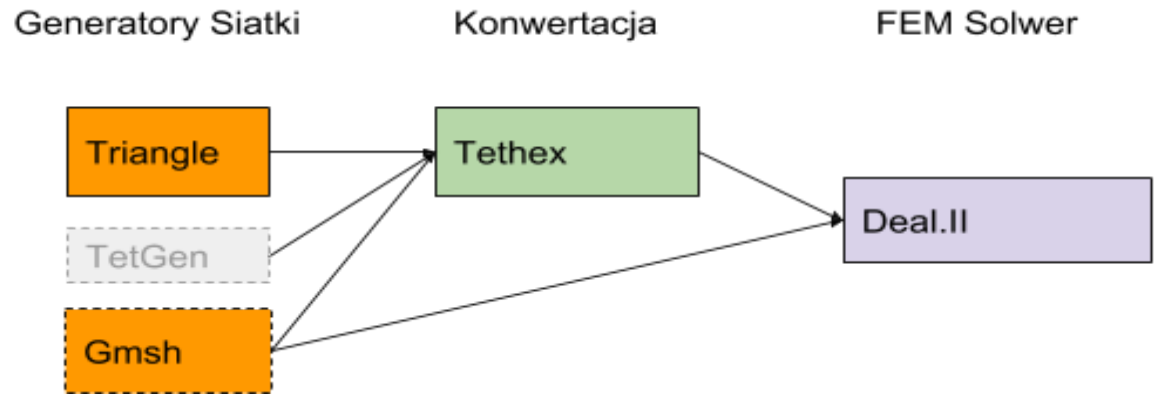
Boost - to wielkie rozwinięcie biblioteki STL w C++. Korzystamy z jej komponentu program_options, który pozwala wprowadzić do programu przez konsolę różne opcje. Też jest dodana możliwość napisania testów. Ale testy jeszcze nie są napisane, ponieważ nie ma na tym etapie potrzeby na nie.

6.Wniosek

Po wypróbowaniu wszystkich powyższych bibliotek i sposobów pracy z nimi, zostały wybrane tylko następujące biblioteki:

- Triangle
- Tetgen
- Gmsh
- Tethex
- Deal.II
- Boost

Diagram zależności realizowany w programie między bibliotekami:



Struktura Programu

Program jest bazowany na przykładach podobnych

1.Struktura plików

Struktura plików programu jest następująca:

- docs: mieści plik Doxyfile.in który jest wykorzystany przez Doxygen dla generacji dokumentacji(Doxyfile.in jeszcze nie jest do końca skonfigurowany)
- Research: w tym folderze są różne przykłady kodów dla korzystania z Gmsh C++ API, Deal.II, Triangle i innych bibliotek
- Results: niektóre ciekawe wyniki, albo przykłady różnych możliwości programu

- Source: Ma w sobie punkt wejściowy programu: main.cpp. Kod źródłowy dla:
    - Tethex
    - TetGen
    - Triangle

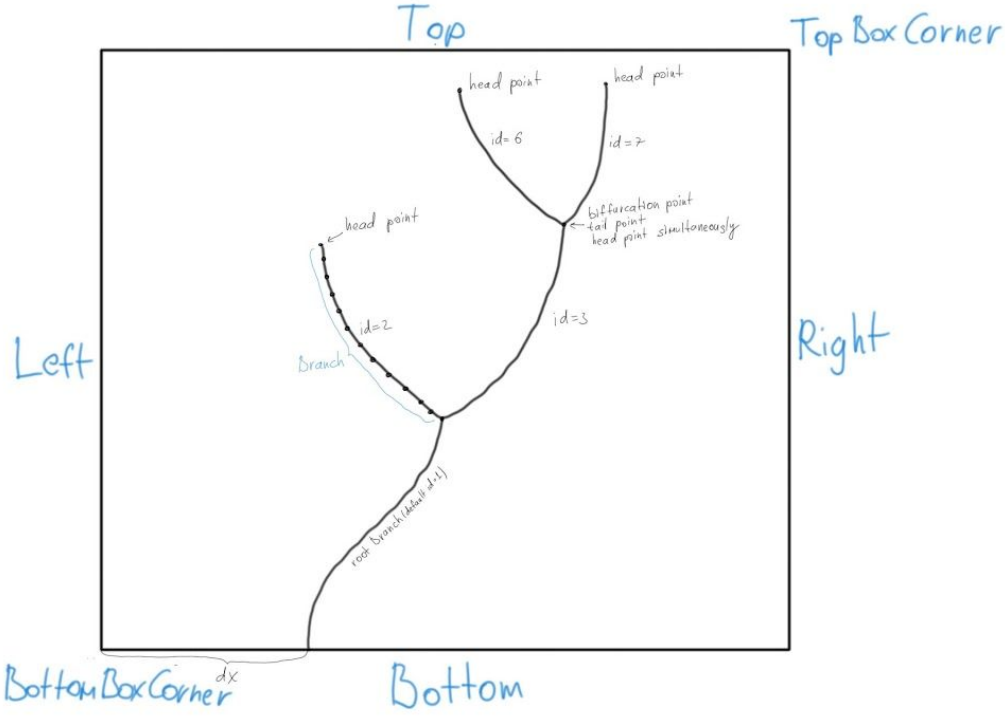
  A także common.* - mieści deklaracje klasy które są korzystne powszechnie w programie
  geometry.* - mieści kod który ułatwia pracę z geometrią dendrytu.
  mesh.* - klasy dla pracy z siatką, odczytać z pliku, zapisać plik, konwertować do kwadratowej siatki, wizualizacja i inne.
  solver.* - realizacja klasy wokół biblioteki Deal.II. I ma następujące możliwości: Odczytać siatkę, zadać warunki brzegowe, udoskonalenie(eng: adaptive mesh) siatki, rozwiązywanie systemu, i opracowanie rozwiązania(na przykład branie całki wokół źródła, co jest w stanie implementacji)
  utilities.* mieści różne dodatkowe funkcje do programu, jak opracowanie danych wejściowych do konsoli i in.
- Tests: w tym folderze są umieszczone unit testy

2.Klasy i ich zależności

Klasy dla opisania geometrii dendrytu

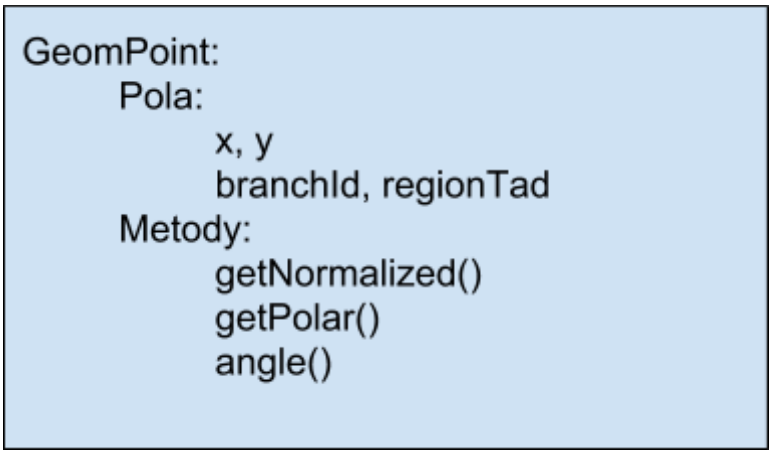
Dla ułatwienia pracy z geometrią dendrytu, była stworzona klasa - Geometry.

Konwencje nazw na rysunku są również wykorzystane w programie:



Rysunek można podzielić na następne bazowe obiekty - punkty, linie albo gałęzie i brzegi. Te obiekty mają odpowiedniki i w programie:

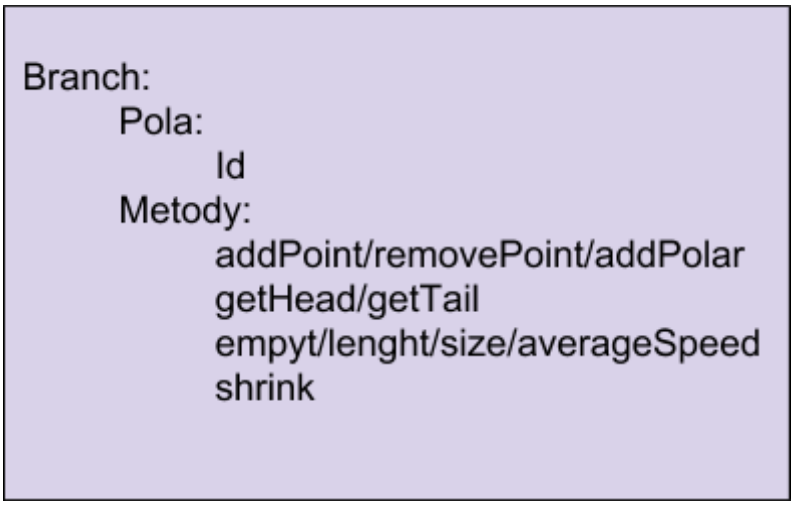
GeomPoint



Ta klasa opisuje geometryczny punkt. I ma ważne dwa pola branchId i regionTag po których wiemy jakie warunki brzegowe stosować i do jakiej gałęzi należą.

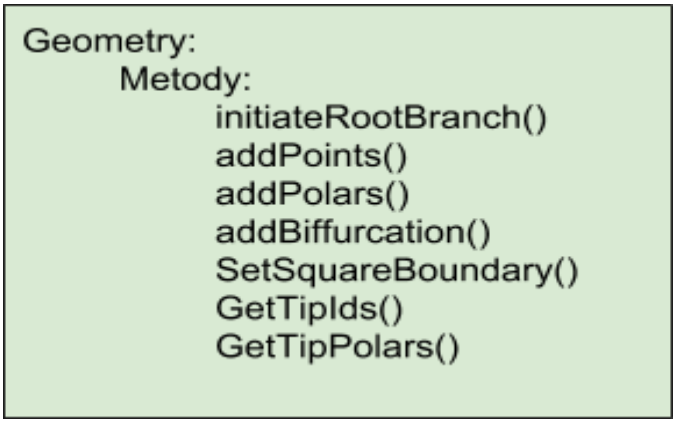
getNormalized() - traktuje punkt jako wektor i wydaje jego długość. GetPolar() - wydaje wektor w polarnych współrzędnych. Są i inne metody, ale te są najważniejsze, ponieważ korzysta się z nich w następnej klasie.

Branch



Ta klasa, reprezentuje pojedynczą gałąź i ma takie metody jak dodać/usunąć punkt.
Też kilka metod o statystyce - długość, średnia prędkość wzrostu(ilość punktów podzielona przez długość).
Shrink - jeszcze nie implementowana, ale będzie skracać gałąź o pewną długość.

Gałęzie są połączone między sobą, a także są połączone z brzegami i za te połączenie odpowiada klasa Geometry:



SetSquareBoundary - zadaje geometrie brzegów i to ma być zrobione na samym początku, zatem
Inicjalizujemy pierwszą gałąź z pomocą - initiateRootBranch()

Dana klasa też dodaje punkty, ale może już to robić do wszystkich gałęzi jednocześnie albo znając “id” pojedynczej gałęzi.

addBiffurcation - dodaje bifurkacje do gałęzi “id”.
GetTipPolar - wydaje kąt i długość czubków co jest wygodne przy całkowaniu wokół czubka.
GetInitialMesh - wydaje siatkę związana tylko z punktów i segmentów, która nadal jest wykorzystana w klasie Triangle.

Triangle Klasa
Z punktów i segmentów generuje siatkę. Ma metody dla pracy z plikami i dla wizualizacji. Dana klasa jest naśladowana z klasy tethex::Mesh. Przez co Triangle też potrafi konwertować w kwadratową siatkę.

Po wygenerowaniu siatki dodajemy ją do klasy Solver:

Solver
Solver enkapsuluje w sobie całą bibliotekę deal.ii, struktura programu jest mało przejrzysta ale są dość dokładnie opisaneana przykład tutaj (https://www.dealii.org/8.5.1/doxygen/deal.II/step_1.html)

Realizacje danej klasy mieszczą następujące metody:
setMesh() - odczytuje mesh od klasy Triangle
setBoundaryValues() - nadaje znaczenia na granicach w zależność od regionu(regionTag)
run() - uruchamia symulacje.

Instalacja i korzystanie z programu

Wystarczy ściągnąć Installer Package dla deal.II (https://www.dealii.org/download.html) i rozpakować go w lokacji /usr/local/dealii

I SDK dla Gmsh z oficialnej strony(http://gmsh.info/#Download) i rozpakować go w lokacji /usr/local/gmsh

Jest też potrzebny pakiet Boost co dokonuje się w następny sposób:

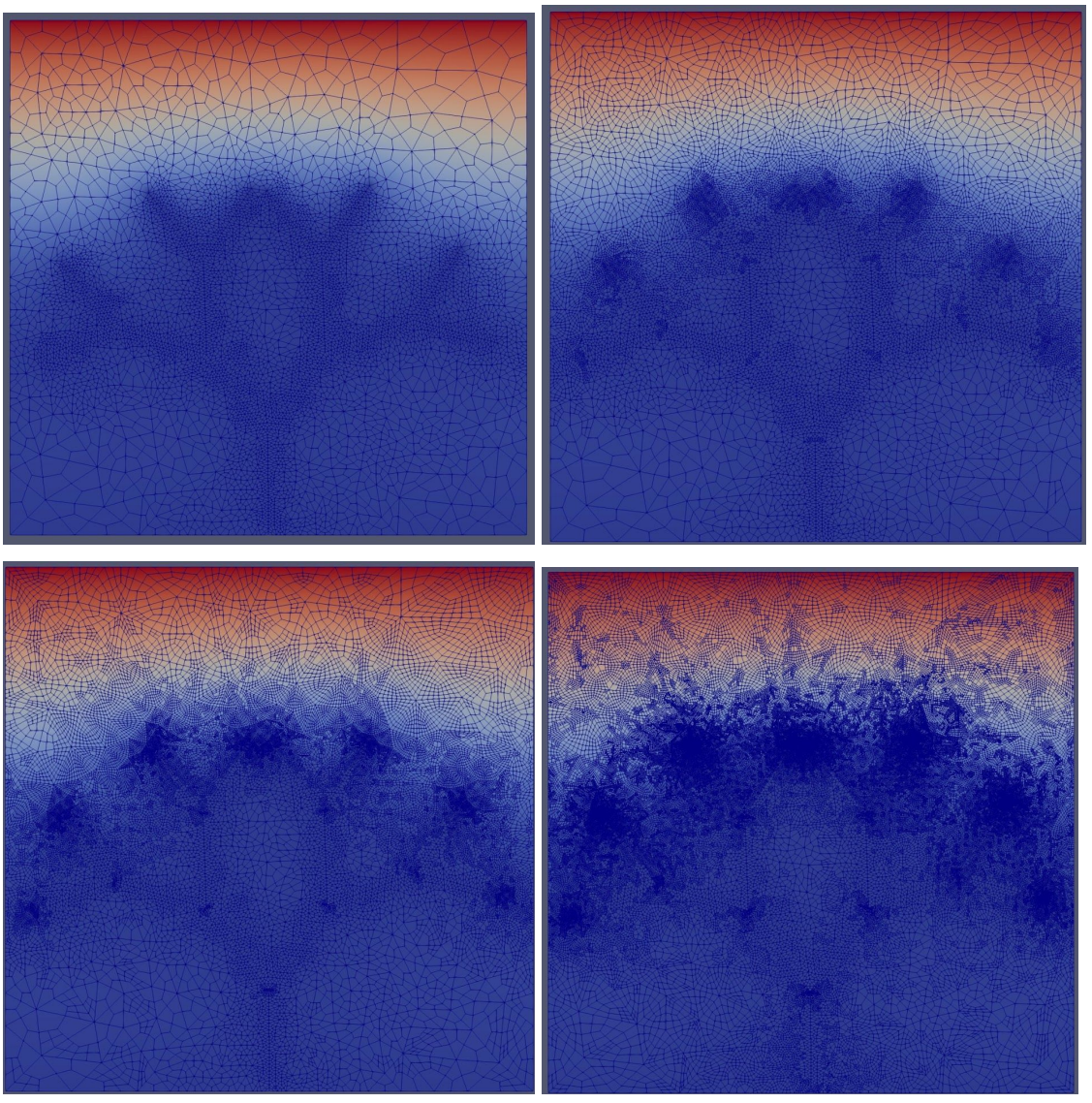
sudo apt install libboost-all-dev

Pierwsze Wyniki

W następnych wynikach geometria dendrytu była zadana ręcznie, a nie wygenerowana za prawami wzrostu.

1.”Prawie” warunki Laplace'a i adaptywny mesh

“Prawie” ponieważ na górze jest narzucony warunek brzegowy typu Dirichleta. Zadanie warunków Neumanna wymaga modyfikacji równania, co jest w trakcie.



2.Warunki Poissona