주요 모듈

pgrow3
construct Family
convolve Family
        loc_levset_to_volfluid
presence; get_nhd_grains2original
updatelevelsetdata2dIsotropic

레벨셋은 thresholding되지 않고, (min_phi - phi) 값만 없데이트 된다. 실제 thresholding 은 VoF representation을 사용할때 일어난다.

레벨셋 값이 양수면 VoF=1 (그레인 내부),
레벨셋 값이 음수면 VoF=0 (그레인 외부),

레벨셋은 굳이 ± 1을 만족시켜야하는 게 아니다. $l(x) = 0$ 인 값이 그레인 바운더리를 표현하기만 하면 될 뿐이다.

[1] Voronoi tesselation 과정 이해
[2] Grain Data structure 표현 방식 이해
[3] 그레인 주위의 인접 그리드를 검색하는 pgrow3이해
[4] 그레인들을 패밀리로 구성하는 방법 이해
[5] 그레인 패밀리를 convolution 시키는 방법 이해
[6] 하나의 픽셀 위치에 대하여 가장 큰 convolution 값을 만들고 있는 그레인을 찾는 방법을 이해

[2] Grain Data structure 표현 방식 이해

Each grain is defined by the list of grid index that (a) belong to the grain and other grid that is near to (a)

grains{k,1} = List of pixel index that belongs to grain k
grains{k,2} = Level set values. Initially, they are either +1,-1. Outside -1, inside +1
grains{k,3} = 0*ind2; % Convolution vals. of alpha kernel init to 0.
grains{k,4} = 0*ind2;   % Convolution vals. of beta kernel init to 0.

[5] 그레인 패밀리를 convolution 시키는 방법
아래 함수는 패밀리 그레인들의 인접 영역에서만 convolution을 계산하고 저장한다.

컨볼루션은 FFT해서 주파수 도메인 얻고 여기에 커널 곱하고 다시 IFFT 시키는 방법으로 계산되고 있다.

function cval = convolvefamily2doriginal(ind,val,dims)

```
% This is C function
vf = loc_levset_to_volfluid(int32(x),int32(y),val,Z);

%Family들의 level set value들을 모아서 volume of fluid representation을 시키는데 이 과정을 시각적으로 확인해볼 수 있을까? -해보았음.

%% Carry out the convolution:
Ku = zeros(dims);
Ku(ind) = vf; % Characteristic function of the union of grains.
Ku = real(ifft2( fft2(Ku) .* KERNEL ));
cval = Ku; % Convolution values, returned.
```

[6] 하나의 픽셀 위치에 대하여 가장 큰 convolution 값을 만들고 있는 그레인을 찾는 방법을 이해. 아래 함수는 이 코드에서 가장 복잡한 섭함수로 2중 포인터 개념이 들어가있다. 메모리 절약에 가장 핵심이 되고 있는 코드이다.

모든 *grid* 포인트에 대하여 인접한 그레인 라벨들과 그 *convolution values*를 리스트로 저장하는 함수이다.

presence = get_nhd_grains2doriginal(grains,dims(1)*dims(2));

```
The output "presence" is an dim1*dim2-by-3 cell array:
presence{index,1} = Grains in a nhd. of pixel at loc. index.
presence{index,2} = Conv. vals. of those grains, in same order.
presence{index,3} = Used to locate this pixel in each grains's
                    data structure.
```

Data structure의 가장 기본은 그리드 포인트이다. (*Index :* grid index)
다시말하자면 presence structure의 갯수는 그리드 포인트 갯수만큼 존재한다.

Each Buffer grain에 대해서 루프를 돈다. (k)
현재 그레인에 속한 pixel들에 대해서 루프를 돈다. (i)
전체 픽셀중에서 현재 그레인에 현재 픽셀의 presence cell에 접근한다.
현재 presence cell에 현재 그레인 아이디 k를 입력한다.
현재 presence cell에 현재 그레인 structure에서

이 픽셀 위치 (loc)를 저장한다.
현재 presence cell에 현재 그레인의 컨볼루션 값을 저장한다.

Prescence cell의 데이터 순서를 cval값으로 정렬한다.
(c언어 빌트인 함수인 qsort가 사용된다)

updatelevelsetdata2doriginal(presence,grains,ID,ORI,angBrandon);

골때리는 것은 Surface tension을 정의하는 함수가 여기에도 정의되어 있다.

모든 그리드 포인트에 대하여 j-루프를 돈다.
현재 그리드 포인트의 presence cell (근접한 k-그레인들)에 접근한다.
presence cell안의 정보는 컨볼루션값 (cval)순으로 정렬 되어 있다.
cval값들을 모은다.

phi값들을 모은다. —phi[k]
(k-그레인 and 다른 ell-그레인 (ell≠k)와의 surface tension값이 계산된다.)

presence cell 안의 그레인에 대해 k-루프를 돈다.
k-그레인과 가장 작은 phi를 형성하는 ell을 찾는다.
그리고 그 최소 phi값을 minphi[k]에 저장한다.

이를 이용하여 레벨셋값을 복원할 것이다.
presence cell 안의 그레인에 대해 k-루프를 돈다.
pgrainlevsetvals[i] = (minphi[k] - phi[k]);
현재 인덱스 위치에서의 가장 최소 phi값 - phi[k]
레벨셋값은 ± 1임을 기억하자.

pgrainlvelsetval[i]는 현재 j-그리드 포인트에 근접한 모든 그레인들 데이터셀에 접근하여서 레벨셋 값을 $\phi_{min} - \phi_k$로 입력하고 있다. ϕ를 최소화하는 그레인에 대해서는 레벨셋 값이 0으로 입력되고, 그 외에는 음의 값으로 입력될 것이다.

Thresholding은 레벨셋을 VoF로 표현할때 일어난다.

[newgrains,newid] = removeemptygrainsoriginal(grains,id)

Optional인데 사용되면 알고리즘 속도를 높여준단다.

그냥 새로운 그레인 스트럭쳐 구조를 만듦. 중간에 사라져버린 그레인이 있으면 데이터 구조를 이 데이터를 어레이에서 삭제하고 뒤에 있는 데이터를 앞으로 땡겨서 저장한다.

RefreshGrainBuffers

이 부분은 함수는 아니지만 본 코드 내부에서 하나의 역할을 하는 식으로 작업을 한다.
고쳐쓸수 있으면 함수로 고쳐쓰는 것을 추천한다.

그레인 데이터 구조는 내부와 그 인접 영역으로 구성된다. 각 타임스텝마다 그레인 바운더리가 이동했으면 그레인 인접 영역 (Bufferred Grain) 역시 다시 설정되어야 한다.

여기서 Z는 디폴트 값이 -1인 워크스페이스일 뿐이며 값을 전달하는 매개체로만 사용된다. 전달해야할 픽셀들에서 해당 함수값을 저장- 전달했다가 용도를 마친뒤에는 곧바로 -1로 초기화 시킨다.

레벨셋값은 ± 1임을 기억하자.

k-그레인을 돌면서 그레인 내부 영역의 픽셀을 모은다 (level set이 positive)
이 픽셀들을 pgrow시켜서 새로운 버퍼 영역을 만든다.

새 버퍼 영역에 이전 버퍼의 레벨셋 값을 전달한다.
(그레인 내부는 변하지 않았기 때문에 그레인 내부에는 그대로 $+1$이 전달되고, 변경사항이 있는 그레인 외부에는 항상 -1이 전달 될 것이다. Z의 디폴트 값이 -1이므로.

똑같은 방법을 convolution값을 전달하는 데에도 사용한다.

---

get_results2D32 (arguments) //Driver function

run simulation2D (dimension, time step size, Rgrain, Rfamily)
initialvoronoidata2d(N,dims,Rgrains,th_ang)

[grains,ori]=initialvoronoidata2d(N,dims,Rgrains,th_ang)

Generates initial condition with "N" grains on a grid of size “dims".
ori = 2*pi*rand(N,1);

Each grain is defined by the list of grid index that (a) belong to the grain and other grid that is near to (a)

grains{k,1} = List of pixel index that belongs to grain k
grains{k,2} = Level set values. Initially, they are either +1,-1. Outside -1, inside +1
grains{k,3} = 0*ind2; % Convolution vals. of alpha kernel init to 0.
grains{k,4} = 0*ind2;   % Convolution vals. of beta kernel init to 0.

ind2의 의미를 이해해야함
그러려면 pgrow를 이해해야함. pgrow는 c파일로 작성되어있음.
pgrow는
  * Grows (dilates) the subset of grid by w pixels outwards.
  * Periodic boundary conditions used.
  * The vectors x and y should be COLUMN VECTORS of type int32!

최초 세팅이 끝나면 함수 gdm2d_sim 함수가 Evolution을 시작함

gdm2d_sim(dt, grains,dims,ori, Rgrains, Rfamilies, angBrandon,list_t, filename, mobility_option)

[alpha, beta] = kernel_widths2d(ori,angBrandon,option);

surface_tension2d(ori,angBrandon): This function construct the surface tension matrix with size NxN.

list_t = vector with times at which grain is saved.
%   list_t_original = vector with times at which grain is saved.

gdm2original_sim —-sim은 시뮬레이션이고 original은 mobility=surface tension

Original이 아니면 Mobilities determine by input variable "option".

gdm2doriginal_sim을 뜯어보기 시작하면됨

실행:
gbm2doriginal_sim(dt,grains,dims,ori,Rgrains,Rfamilies,angBrandon,list_t,filename)

[a] Generate kernel on the specified grid.
[X,Y] = [1,128] x [1,128] :Grid

[b] for fast convolutoin, far away grains are grouped into families
Then, the entrie family (union of grains) is covolved at once

[families,famgrains] = grains2families2doriginal(grains,Rfamilies,dims); % Group grains into families.

What are they?

initialvoronoidata2d(N,dims,Rgrains,th_ang);

% How the grain data is structured? during the initializatoin?
[grains,ori] = initialvoronoidata2d(N,dims,Rgrains,th_ang);

Ind2sub (change index between 1D array and matrix)

Family 만들기 (Far apart grain을 기준으로)

Family라는 구조는 먼저 아래와 같이 사이즈가 확정된 구조이다.
우리는 물론 최소한의 Familiy로 그룹화 할 것이다. 나중에 슬라이싱이 사용된다.

maxFamilies=1000;
families = cell(maxFamilies,4); % Maximum of maxFamilies families anticipated.

Loop over family: Family

Family 1번부터 시작해서 최대한 많은 그레인을 채워넣는다.

이를 위해 intersect에 현재 패밀리와 근접한 그레인들을 누적해서 마크할 것이다.

1번부터 N번까지의 그레인을 다 루프로 검색한다.
만약에 k-그레인이 아직 어떤 패밀리에도 속하지 않고 있으면 (remaininggrains) +
현재 패밀리안에 있는 그레인과 접하지 않고 있으면 (intersect)

k-그레인을 현재 패밀리에 추가한다.
k-그레인을 확대하면서 인접한 그레인들을 찾아서 intersect에 마크한다.

현재 패밀리에 그레인이 채워지고 있으면 이를 max label에 기록한다.
% maxlabel will identify the family size used

패밀리 안에 그레인 숫자를 하나 더 증가시킨다.
패밀리에 listofgrain를 마크한다.

Family Convolution 하기

loc_levset_to_volfluid.c

Volume of fluid (VOF) method is a free-surface modeling technique for tracking and locating the free surface. Fraction function C is defined as the integral of a fluid's characteristic function in the conrol volume. The volume fraction of each fluid is tracked through every cell in the computational grid, while all fluids share a set of momentum equations.

To convert volume of fluid and level set method, you need intergral of characteristic funciton (level set)

And in this step, you typically need step, heavi side step functions.

Level set 장점 curvature 구하기가 쉽다.
Volume of fluid 장점 mass conservation 이 쉽다.

Convolution을 level set이 아니라 volume of fluid representation에서 계산하고 있는데 왜일까?
level set은 언제 계산했었지? 레벨셋은 +1,-1로만 가지고 있고 VOF 표현방식으로 구현해서 Convolution을 취해준다.

MexFunction 문법

```
#include "mex.h"
void mexFunction(int nlhs, mxArray *plhs[], int nrhs,
  const mxArray *prhs[])
```

nlhs, number of output arguments
plhs, array of pointers to the expected mxArray output arguments
nrhs, number of input arguments
prhs, array of pointers to the mxArray input arguments

mxGetM(C): returns the number of rows in mxArray C

Variable dictionary

Seeds = list of seeds to use when generating the random initial data

Rfamily: minimum distance to maintain between grains in the same family
angBrandon - Brandon angle in read Shockeley model model (degree)

Rgrains=10; radius by each a grain is enlarged.?
Rfamilies=30; Minimum pixel distance to maintain between grains in the same family.

th_ang - threshold angle to merge grains
ori: orienation of grains, shape of $(N \times 1)$, e.g. ori = 2*pi*rand(N,1);