heights.h Documentation

Haochen(Langford) Huang

April 23, 2024

version:1.0

1 Introduction

'heights.h' together with 'parabola.h' serve as toolbox for 'curvature.h' to compute surface curvature in multiphase flow. The aim of the file is to allocate value which represents distance towards surface to every single cell. And the value is named 'height', therefore the name of current headfile. Before diving into details, I shall first introduce several conceptual definitions with 1D example.

1.1 Surface

For each cell, if a 'coherent surface' occurs within certain distance, then the individual will be assigned with valid height value. Otherwise, invalid data 'nodata' will be allocated to such cell. 'coherent surface' defined as process where color function (volume fraction in VOF method) c changes from 0 to 1 or vice versa.

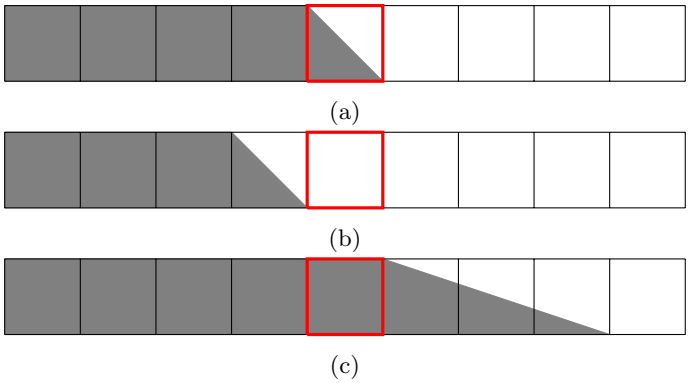

Figure 1: Graphical representation of coherent surface. Cell highlighted by red square represents current cell. Color grey indicates volume fraction.

Figure 1 displays occasions where height value is valid for current cell (the one hightlighted by red square). Within certain distance, cell fully immersed in color function (represents by color grey) and those with 0 value both occur. On the contrary, figure 2 demonstrates occasions in which 'coherent surface' is not observed. Therefore 'nodata' is assigned.

1.2 Height value and 0 value point

Nature of height value is the summary of color function, and the value of each cell is represented by that at center of the individual. In order to obtain specific value, the position where 0 is located should first be defined. Consider a condition as illustrated in figure 3 where surface is indeed two cell with color function of 0.6 and 0.1. 0 point then located at the place 0.7 away from the final occupied cell i.e. blue line in figure 3. Then value of each cell is calculated based on the distance between cell center and 0 position as demonstrated beneath the figure. The positive direction points towards inner side of surface.

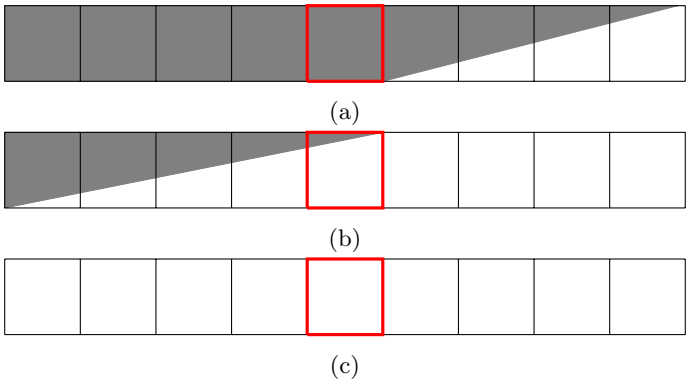

Figure 2: Graphical representation of not coherent surface. Cell highlighted by red square represents current cell. Color grey indicates volume fraction.

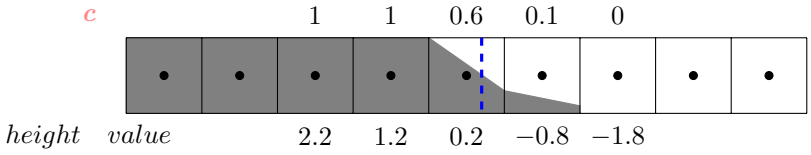

Figure 3: Graphical representation of 0 position. Blue dashed line indicates the exact position of 0 position. Numbers on the top reveals the color function of each cell while those beneath displays corresponding height value at center of each cell as shown by black dot.

1.3 Direction of surface

Except example shown in figure 3 there do exist another possibility that surface stems following opposite direction (from right to left in this example). In order to distinguish these two occasions the value of 0 position is defined to be 20 for cases show in figure 4. Typically, only cells positioned within 5.5 distance away from 0 position has valid height value. Therefore the range of height value is $[-5, 5]$ or $[15, 25]$ based on surface direction.

2 Functions

2.1 Overall configuration

Before introducing parameters and workflow, I shall firstly deliver a brief introduction about overall configuration of three functions which will be discussed in the following sections. Figure 5 depicts the overall function configuration of current headfile. **heights** serves as console and delivers command to **half_column** in which volume fraction integration is conducted on neighbourhood cells. After iterating the adjoining 8 cells (4 for both positive and negative direction), the height value is allocated to the current cell. Note that special care is taken for tree girds and the additional function **refine_h_x** is employed to prolongate color function on tree grid.

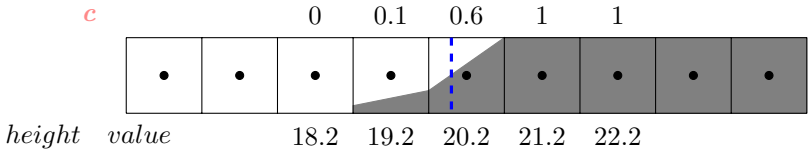

Figure 4: Graphical representation of 0 position. Blue dashed line indicates the exact position of 0 position. Numbers on the top reveals the color function of each cell while those beneath displays corresponding height value at center of each cell as shown by black dot.

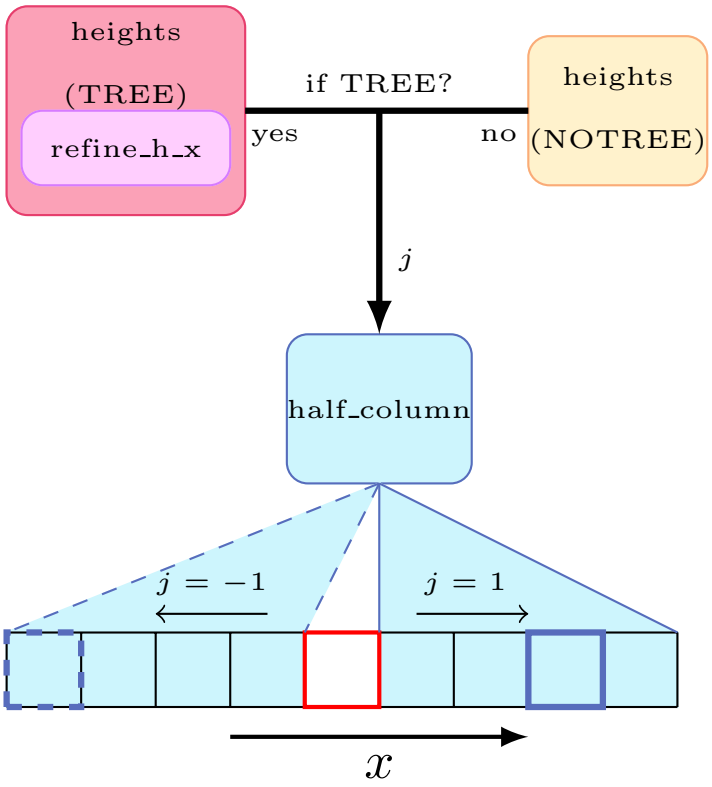

Figure 5: Overall configuration of current headfile. Red square highlights current cell and the blue square represents cell scanned by **half_column**.

2.2 height&orientation

2.2.1 Worth mentioning details

These two functions are 'tricky' functions. As introduced in section 1, in order to distinguish the orientation of the surface, range of final value has two options: $[-5, 5]$ or $[15, 25]$. **height** is the reverse function which takes height value as input, erases disguise of orientation and returns the real distance between current cell and 0 position. The range of corresponding output turns out to be $[-10, 10]$.

By contrary **orientation** returns surface orientation based on height value. To save memory, tricky function manifests as 'inline' function, i.e. substitute corresponding code when it is called.

2.2.2 Workflow

height
The disguise value *HSHIFT* is 20. By adding/removing such value, true interfacial distancne is revealed.

```
#define HSHIFT 20.

static inline double height (double H) {
  return H > HSHIFT/2. ? H - HSHIFT : H < -HSHIFT/2. ? H + HSHIFT : H;
}
```

orientation
return TRUE or FALSE based on disguise value. Two layers of ghost value is set on every boundaries.

```
static inline int orientation (double H) {
  return fabs(H) > HSHIFT/2.;
}

#define BGHOSTS 2
```

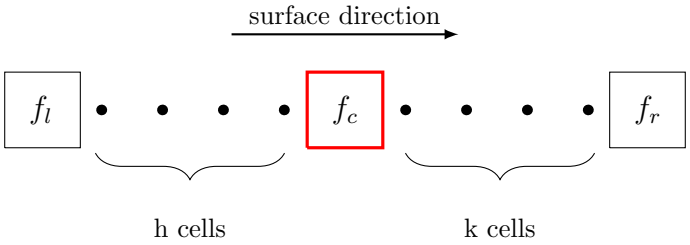

Figure 6: Sketch of position switching between cells. f_c, f_r, f_l represent color function value of current cell (red square) and its right/left side neighbor, respectively.

2.3 half_column

Based on previous discussion **half_column** plays major role in height computing whose duty is to scan over neighborhood cells and feedback status value of surface finding as well as height value. Owing to its complexity, I shall carefully interpret the algorithm part by part instead of briefly introducing selected details.

2.3.1 Parameters

Name	Data type	Status	Option/Default	Representation (before/after)
point	Point	unchanged	compulsory	current cell position
c	scalar	unchanged	compulsory	c (volume fraction)
h	vector	**output**	compulsory	**h**
cs	scalar	unchanged	compulsory	$c[2 * j]$
j	int	unchanged	compulsory	j (direction control)

2.3.2 Purpose of the function

half_column has two purposes:

1. To check whether there is a coherent surface (defined in section 1) within 8 adjoining cells.
2. If there is coherent surface, compute corresponding height value by integrating color function of each cell.

There are three possible situations for current cell: $f_c = 1, f_c = 0, f_c \in (0, 1)$. Where f_c represents color function value of current cell. For first two situations, the threshold for coherent surface is to find certain neighborhood in which $f = 0, f = 1$ respectively. However for third condition i.e. the cell located on surface, one should iterate both directions to find a pair of opposite (full and empty) cells on each side.

Once coherent surface is found, the problem remaining is how to calculate the height value. The key to understand such process is position switch between cells. Figure 6 displays an example of such condition, where f_c, f_r, f_l represent color function value of corresponding cell. Following the surface direction (defined as vetor points from inner side to outside of surface, e.g. figure 3.), relationship between three values reads

$$f_c - (k + 1) = f_r \tag{1}$$
$$f_c + (h + 1) = f_l \tag{2}$$

Among all the cells, the one whose height value is easiest to obtain is the final full cell close to the surface (hereinafter as final cell). Figure 7 demonstrates same example in figure 3 but highlight the final cell and corresponding surface direction. Based on discussion in section 1, the height value of final cell can be compute by integrating the color function along surface direction till the very first empty one (cell E) reads:

$$h_B = h_B + h_C + h_D + h_E - 0.5 = 1 + 0.6 + 0.1 + 0 - 0.5 = 1.2 \tag{3}$$

Three terms in R.H.S represents color function of cell B, C, D, E. While the -0.5 stems from the fact that height value of one cell is determined from its center instead of left boundary. Combined with the

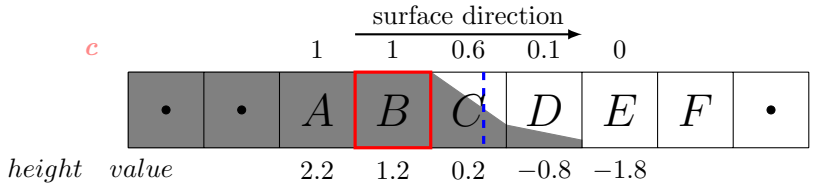

Figure 7: Same example as in figure 3 with two example stencils A, B. Red square here used to highlight the final full cell next to surface.

position switch discussed previously, we can then obtain height value of every cell. In short, to obtain a height value, one should first switch to 'final cell' and integrate through surface. For example, height value of cell F yields $h_F = h_B - 4 = -2.8$ and that of cell A yields $h_A = h_B + 1 = 2.2$.

2.3.3 Configuration of the function

Figure 8 carefully depicts configuration of **half_column** which consists of three layer. As illustrated in figure 5 and section 2, **half_column** will be called twice during the process, first for negative direction ($j = -1$) then for positive direction ($j = 1$). Except for second layer 'iteration', the rest two will be altered according to iterating direction.

2.3.4 Program Workflow

Starting Point
input:
$a = x^k$ $res = RES^k$ $da = \delta x^k$
relax=relax function
data = Poisson structure
nrelax=times relax function applied to each level
minlevel=minimum of mesh level
maxlevel=maximum of mesh level

```
1 void mg_cycle (scalar * a, scalar * res,
  ↪  scalar * da,
2                 void (* relax) (scalar *
                  ↪  da, scalar * res, int
                  ↪  depth, void * data),
3                 void * data, int nrelax,
                  ↪  int minlevel, int
                  ↪  maxlevel)
4 {
```

RES restriction
To restrict residual from finer mesh to coarser mesh.
Entering of iterative
Iterate starting form lowest (coarsest mesh) level.

```
1     restriction (res);
2     minlevel = min (minlevel, maxlevel);
3     for (int l = minlevel; l <= maxlevel;
      ↪  l++) {
```

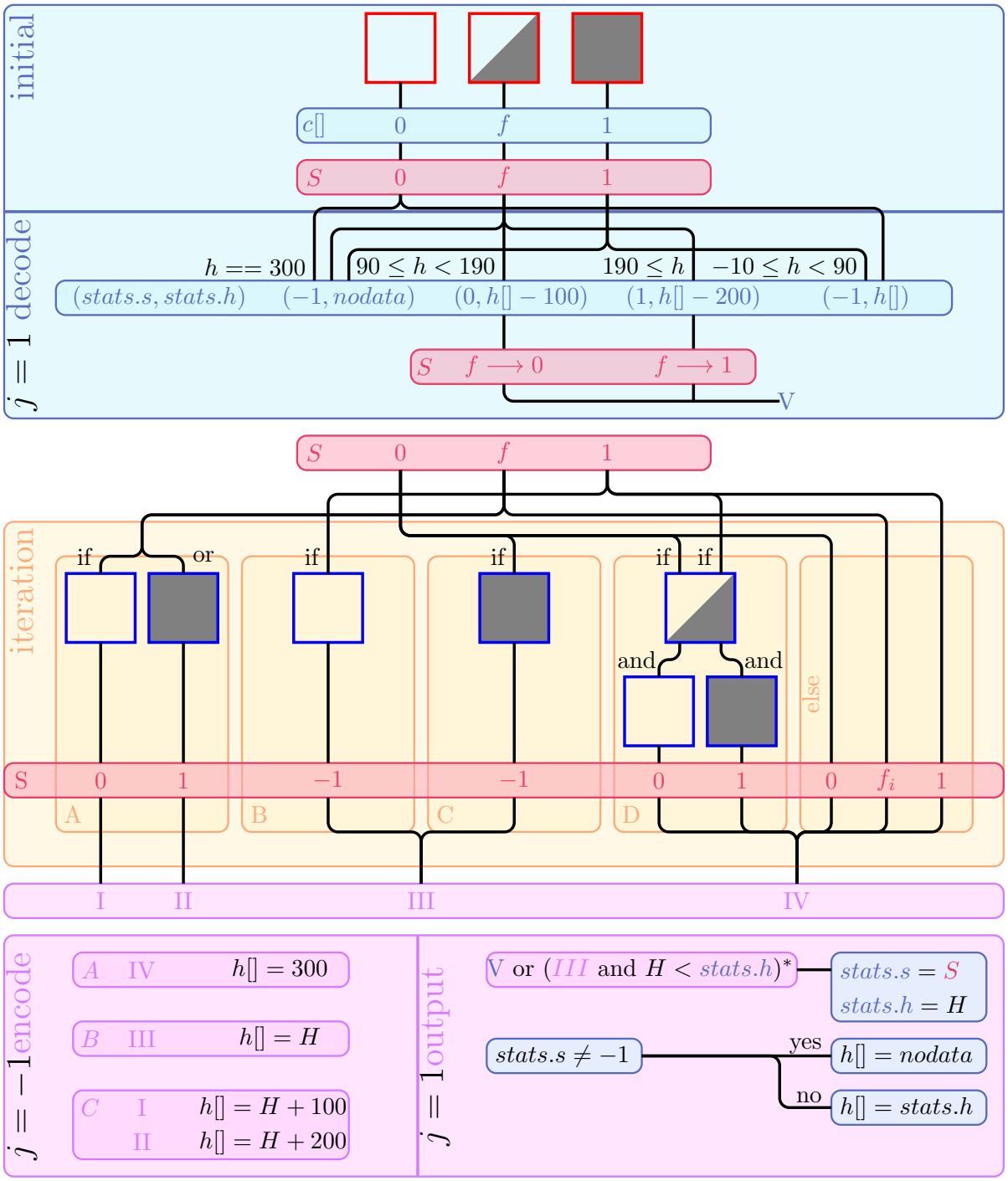

Figure 8: Detailed description about **half_column**. For sake of simplicity, some tricky treats have been hidden. For instance judgement in third layer highlighted by $*$ compares surface distance instead of direct value of H and $stats.h$. Therefore the actual form of judgement reads $fabs(height(H)) < fabs(height(stats.h))$.