embed-tree.h documentation

Haochen(Langford) Huang

July 14, 2024

version:draft

1 Introduction

Current header file provides restriction, prolongation and refinement under tree grid for corresponding value employed in **embed.h**. The content can be separated into two parts: the refinement of cut-cell value i.e. cs and fs and the restriction/prolongation of value on cut-cell and cut-face.

2 Refinement of Embed associated value

2.1 embed_face_fraction_refine

Given face vector fs on mesh level N, function **embed_face_fraction_refine** returns fs on finer mesh level $N+1$. Computation of face fraction is achieved by first computing the fraction of 'inner' faces then the 'boundary' faces both under $2D$ and $3D$ conditions. For sake of convenience, the implementation on $2D$ cases is first introduced then followed by the implementation on $3D$ condition. The program however is arranged based on computation on 'inner' faces or 'boundary' faces.

Note that the function is duplicated by macro $foreach\_dimension$ and each function is only responsible for facial fraction computation on one direction. The introduction shall take computation on x as an example hereinafter.

Moreover, for a cut-cell ($cs \in (0,1)$) the interface exclusively yields as

$$\mathbf{n} \cdot \mathbf{x} = \alpha \tag{1}$$

where $\mathbf{n}$ is the normal direction of the interface, $\mathbf{x}$ represents coordinates of arbitrary point at the interface and α is the constant. This holds true for both condition and is the critical in identifying different condition in the upcoming sections.

2.1.1 $2D$ Condition

First consider the fine faces contained inside the cell as shown by A, B in figure 1.

The very first step to compute face fraction is to identify whether the interface comes across the inner fine faces. Given $\alpha = \alpha_1$ and $\mathbf{n} = (n_x, n_y)$ of the interface and coordinate system whose $(0,0)$ locates at center of the cell, the lower and upper boundary for the condition is the line set passing $(0,0.5), (0,-0.5)$ respectively. According to equation 1, once

$$|2\alpha_1| < |n_y| \tag{2}$$

the interface will come across the inner fine faces otherwise both the face fraction are 0 or 1 determined by interface direction. If equation 2 holds true the intersection point $(0, y_i)$ between interface and y-axis yields $y_i = \frac{\alpha}{n_y}$. The face fraction for face A, B therefore can be obtained accordingly.

Following a similar process, the boundary fine face fraction is calculated first by identifying the intersection point (if exists) then computing the fine face fraction. Different from the previous, the face fraction can be visited directly without equation 2. The orientation of the interface is obtained by checking corresponding transverse faces as shown by (1-4) in figure 2. Combined with face fraction on original face, the fine face fraction is therefore obtained.

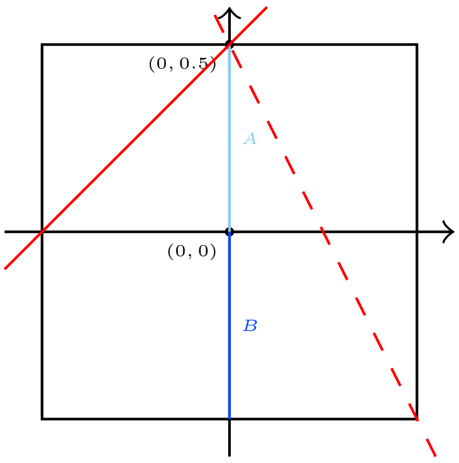


Figure 1: Sketch for inner fine faces A, B, two red lines display example threshold value for the interfaces.

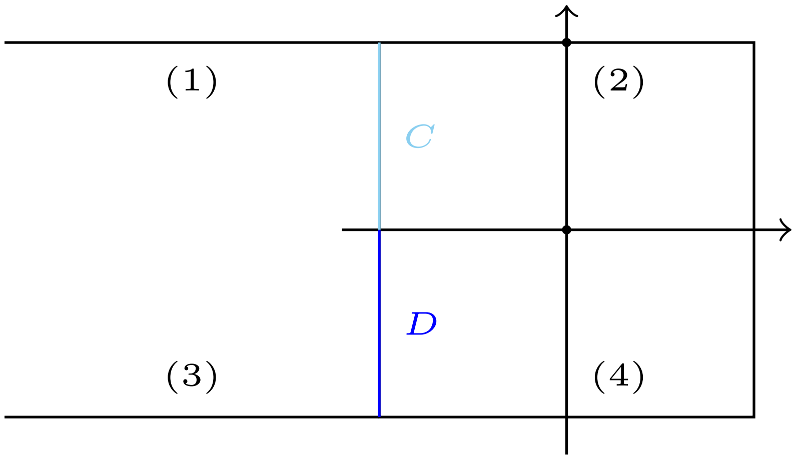


Figure 2: Sketch for boundary fine faces C, D, the numerical labels mark the corresponding transverse faces.

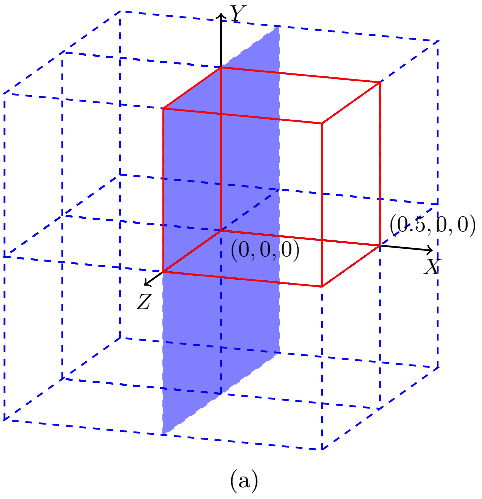
2.1.2 $3D$ Condition

The fine face fraction for both inner and boundary faces are computed directly using $2D$ projection from a $3D$ interface.

First consider the inner face, as shown by figure 3a, the fine face fraction is computed from $2D$ projection in red contour cell. Moreover the iteration is accomplished among four subcells by rotating the whole interface as in section (unfinished). Specifically, given $\mathbf{n} = (n_x, n_y, n_z)$ and $\alpha = \alpha_1$ of the example interface shown by figure 4a, the face fraction on the inner fine face can be obtained by calculating the volume fraction of reconstructed interface shown by figure 4b. Since the volume calculation can be achieved by tool provided by **geometry.h**, the problem degenerates as how to reconstruct the interface and get its $\mathbf{n}, \alpha$.

Note the fact that the reconstructed interface is perpendicular to the projection plane and comes across the same line as original interface on projection plane, its $\mathbf{n} = (0, n_y, n_z), \alpha = \alpha_1$.

Similarly, the fine face fraction on boundary face is also computed by projection but on the boundary plane. An example is shown by figure 4c where reconstructed plane should pass the intersection line on boundary face (blue line) as well as the red dashed. Given an arbitrary point set $(0.5, y_1, z_1), (0, y_1, z_1)$



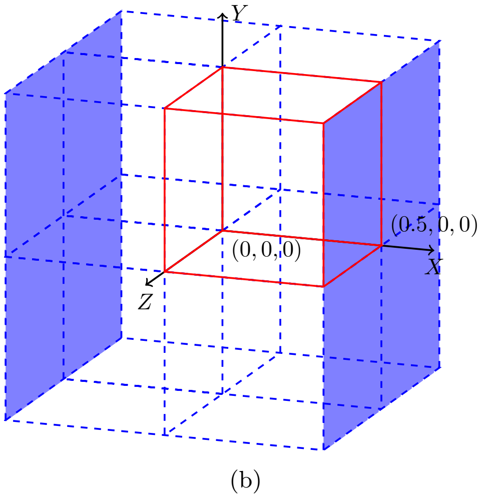
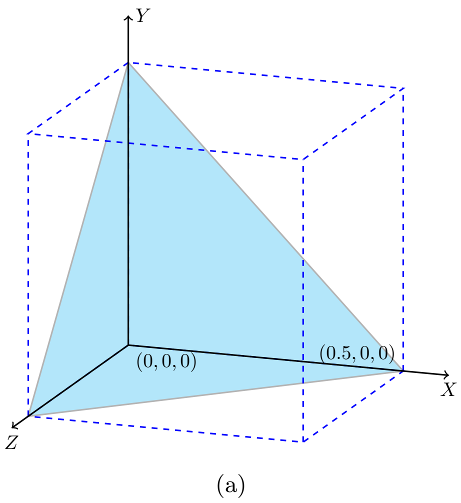
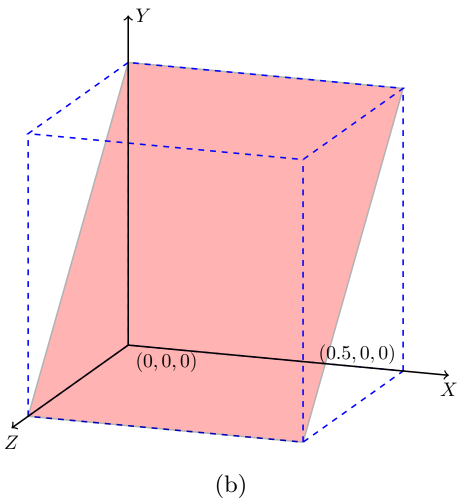


Figure 3: Inner fine face (a) and boundary fine face (b) under 3D condition. The red contour highlights the cell where projection takes place.





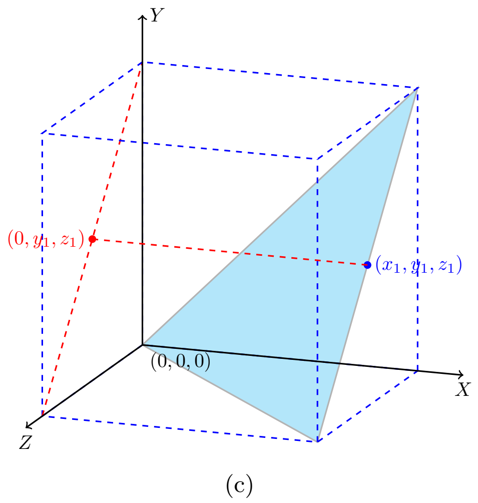


Figure 4: A zoom in for highlighted cell in figure 3. (a) The original interface and (b) the projection interface for inner face in figure 3a and (c) the original interface for boundary fine face in figure 3b.

the $\alpha = \alpha_2$ of the reconstructed interface has relation:

$$0.5n_x + y_1 n_y + z_1 n_z = \alpha_1 \tag{3}$$
$$y_1 n_y + z_1 n_z = \alpha_2 \tag{4}$$
$$\alpha_2 = \alpha_1 - 0.5n_x \tag{5}$$

Consider the normal direction $(0, n_y, n_z)$, the face fraction therefore can be calculated.

3 Readme

Normally, a documentation would consist of two major parts: Introduction & Backround and Function. The first part will introduce the purpose of the corresponding program and the governing equations it solved and other thing developers and users should be aware of *e.g.* in which method the program solve the overall problem. Pragmatic program will be explored line by line in the second part. It first contains a table to clearify all the parameters and their physical representatives as shown in Table.1. The

Name	Data type	Status	Option	Representation (before/after)
a	scalar*	update	complusory	$\delta\mathbf{u}^{*,k}/\delta\mathbf{u}^{*,k+1}$
b	scalar*	unchange	complusory	RES
dt	double	unchange	complusory	Δt
l	int	unchange	complusory	mesh level
data	struct Vsicosity	unchange	complusory	$\mu^{n+\frac{1}{2}}, \rho^{n+\frac{1}{2}}, \Delta t$

Table 1: Referenc table of parameters.

highlighted row in the table indicates such paramter is either the output or has been updated. Second subsection always concerns with detals and specific technique the function employed. Finally the third part is the workflow of the program.

Throughout documentation font *para* represents exact name of parameters and **function** represents exact name of the function.

Tikz inside text example: 🟫.

4 Program Workflow Example

Starting Point
input:
$f = \Phi^n$ $uf = u_f^{n+\frac{1}{2}}$
flux(empty) dt=Δt
$src = \mathbf{g}^n$
gradient:
$g = \nabla f = \nabla\Phi$

```
void tracer_fluxes (scalar f,
                    face vector uf,
                    face vector flux,
                    double dt,
                    (const) scalar src)
{
  vector g[];
  gradients ({f}, {g});
```