

geometry.h 文档说明 (上)

Haochen Huang

西安交通大学 MFM 课题组

版本: 1.02test

日期: 2023 年 4 月 12 日

目录

| | |
|------------------------------------|-----------|
| 1 文件中主要函数及其目的 | 1 |
| 1.1 文件目的 | 1 |
| 1.2 主要函数 | 2 |
| 2 line alpha/plane alpha 函数 | 2 |
| 2.1 函数目的及原理 | 2 |
| 2.2 具体实现代码 | 3 |
| 3 line area/plane area 函数 | 6 |
| 3.1 函数目的及原理 | 7 |
| 3.2 具体实现代码 | 9 |
| 4 rectangle fraction 函数 | 11 |
| 4.1 函数目的及原理 | 12 |
| 4.2 具体实现代码 | 12 |

摘 要

本文为 geometry.h 的文档说明, 该头文件的目的在于为 Basilisk 提供一些精巧的几何工具, 以便为之后的多相流算法 (例如 VOF 算法等) 提供工具支持。由于本文件工具种类繁多, 现暂分上下两部, 上部中为常用, 最值得注意的相关函数。下部则是部分辅助输出函数。

2.02 版本更新, 解决内部排版问题, 增改部分注释

1. 文件中主要函数及其目的

1.1 文件目的

以一个被两相边界分割的 2D 单元为例:

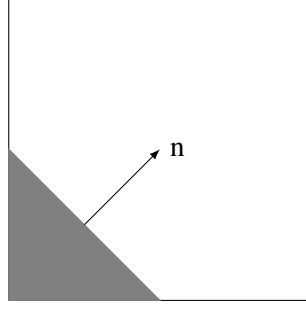


图 1: 2D 示例单元

其中灰色部分代表边界内，白色部分代表边界外，边界的法向量统一指向外侧。假设两相边界的方程为

$$n_x x + n_y y = \alpha \quad (1)$$

其中 n_x, n_y 为单位化后法向量的两个分量，我们由此引入描述边界的三个参数：

- 表示边界内所占该单元体积面积的体积分数 c
- 边界的法向量 (n_x, n_y) (三维则为 (n_x, n_y, n_z))
- 以及边界函数表达中的参数 α

此三者并非相互独立，在已知边界法向量以及剩下两者中的一者后，可以轻易的推导出余下一个参数，而本头文件中的工具就是针对该问题进行一系列的求解。

1.2 主要函数

- line alpha/plane alpha: 用于求解 2 维/3 维情况下已知体积占比 c 以及边界法向量 n 时，该边界在本单元中的参数 α
- line area/plane area: 用于求解 2 维/3 维情况下已知界面参数 α 以及边界法向量 n 时，该边界在本单元中的面积/体积分数 c
- rectangle fraction: 用于求解网格内方形部分的边界内体积/面积分数 c
- facets: 用于输出界面端点坐标，在后处理中可以将端点坐标用直线连接
- line length center/plane area center: 用于存储 2 维/3 维中边界中心，同时输出边界内部在单元中的面积/体积
- line center/plane center: 用于输出边界内面积体积重心坐标

2. line alpha/plane alpha 函数

2.1 函数目的及原理

该函数目的是在给定界面法向量 (n_x, n_y) 以及界面内体积占比 c 后计算界面参数 α 。

算法重点在于坐标转换以及临界面积推导；其中坐标转换会在下一节详细阐述，本节主要讲解面积推导。

我们首先将原本的坐标及图形经历坐标变换全部变为零点在左下角，法向量分量均大于零的状态，见8，具体推到公式及返回见下一节。

当 $c \in (0, 1)$ ，且 n_x, n_y 均不等于 0 时，为了方便区分，算法将分量中较大的一个定义为 n_2 ，较小的一个定义为 n_1 ，我们以 n_y 为较大数为例，有：

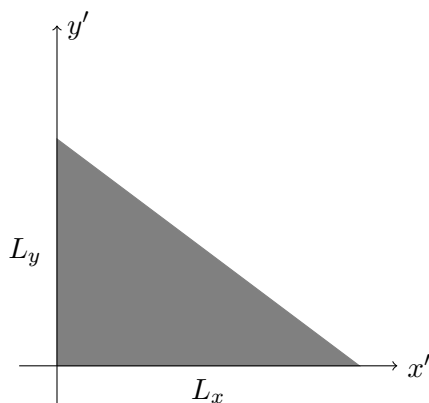


图 2: 界面与两坐标轴的交点

令落在 x' 轴上的长度为 L_x ，令落在 y' 轴上的长度为 L_y ，则 $\frac{L_y}{L_x} = \frac{n_x}{n_y}$ ，且 $L_y \leq L_x$ 。面积共有三种情况图形一共有三种形态，分别是三角形，梯形，以及五边形，详见6，当前目标是：使用已知的 c 与 (n'_x, n'_y) 对其进行判定，首先为三角形临界情况：

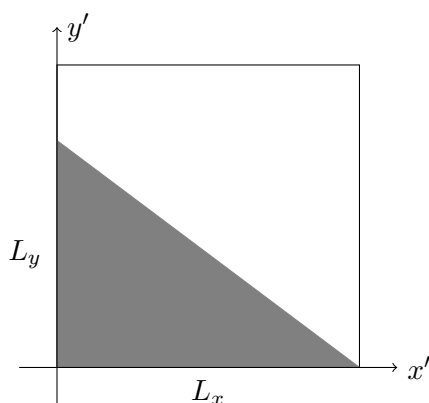


图 3: 三角形面积临界情况

即 $L_x = 1$ 此时其面积为：

$$c = \frac{n_1}{2n_2} \quad (2)$$

是故当 $c \leq \frac{n_1}{2n_2}$ 时，图形为三角形，即可以求出 α ，具体公式详见下节。

而梯形的临界状态则为：

此时单元内右上角空白三角形的面积为：

$$S_{blank} = \frac{n_1}{2n_2} \quad (3)$$

则当 $c \leq 1 - \frac{n_1}{2n_2}$ 时，计算为梯形，则其余均为五边形。相关计算公式同见下节。

2.2 具体实现代码

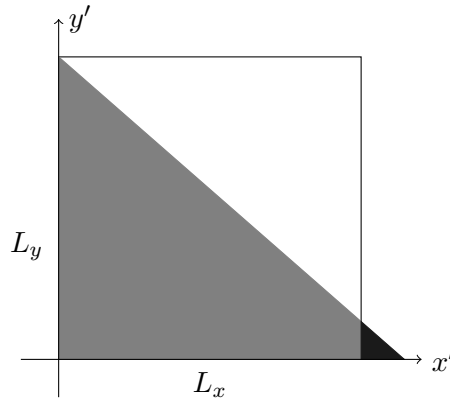


图 4: 梯形面积临界情况

```

1  #if dimension >= 2
2  double line_alpha (double c, coord n)
3  {
4      double alpha, n1, n2;
5
6      n1 = fabs (n.x); n2 = fabs (n.y);
7      if (n1 > n2) //判断法向量中的较大值, 并将其赋于 n2
8          swap (double, n1, n2);
9
10     c = clamp (c, 0., 1.);
11     double v1 = n1/2.;
12     if (c <= v1/n2) //面积为三角形的判断
13         alpha = sqrt (2.*c*n1*n2); //相关  $\alpha$  计算
14     else if (c <= 1. - v1/n2) //梯形判断
15         alpha = c*n2 + v1;
16     else //五边形判断
17         alpha = n1 + n2 - sqrt (2.*n1*n2*(1. - c));
18     //坐标反变换
19     if (n.x < 0.)
20         alpha += n.x;
21     if (n.y < 0.)
22         alpha += n.y;
23
24     return alpha - (n.x + n.y)/2.;
25 }
26 #endif // dimension >= 2
27

```

```

28  #if dimension >= 3
29  double plane_alpha (double c, coord n)
30  {
31      double alpha;
32      coord n1;
33
34      n1.x = fabs (n.x); n1.y = fabs (n.y); n1.z = fabs (n.z);
35
36      double m1, m2, m3;
37      m1 = min(n1.x, n1.y);
38      m3 = max(n1.x, n1.y);
39      m2 = n1.z;
40      if (m2 < m1) {
41          double tmp = m1;
42          m1 = m2;
43          m2 = tmp;
44      }
45      else if (m2 > m3) {
46          double tmp = m3;
47          m3 = m2;
48          m2 = tmp;
49      }
50      double m12 = m1 + m2;
51      double pr = max(6.*m1*m2*m3, 1e-50);
52      double V1 = m1*m1*m1/pr;
53      double V2 = V1 + (m2 - m1)/(2.*m3), V3;
54      double mm;
55      if (m3 < m12) {
56          mm = m3;
57          V3 = (m3*m3*(3.*m12 - m3) + m1*m1*(m1 - 3.*m3) + m2*m2*(m2 - 3.*m3))/pr;
58      }
59      else {
60          mm = m12;
61          V3 = mm/(2.*m3);
62      }
63
64      c = clamp (c, 0., 1.);

```

```

65     double ch = min(c, 1. - c);
66     if (ch < V1)
67         alpha = pow (pr*ch, 1./3.);
68     else if (ch < V2)
69         alpha = (m1 + sqrt(m1*m1 + 8.*m2*m3*(ch - V1)))/2.;
70     else if (ch < V3) {
71         double p12 = sqrt (2.*m1*m2);
72         double q = 3.*(m12 - 2.*m3*ch)/(4.*p12);
73         double teta = acos(clamp(q,-1.,1.))/3.;
74         double cs = cos(teta);
75         alpha = p12*(sqrt(3.*(1. - cs*cs)) - cs) + m12;
76     }
77     else if (m12 <= m3)
78         alpha = m3*ch + mm/2.;
79     else {
80         double p = m1*(m2 + m3) + m2*m3 - 1./4., p12 = sqrt(p);
81         double q = 3.*m1*m2*m3*(1./2. - ch)/(2.*p*p12);
82         double teta = acos(clamp(q,-1.,1.))/3.;
83         double cs = cos(teta);
84         alpha = p12*(sqrt(3.*(1. - cs*cs)) - cs) + 1./2.;
85     }
86     if (c > 1./2.) alpha = 1. - alpha;
87
88     if (n.x < 0.)
89         alpha += n.x;
90     if (n.y < 0.)
91         alpha += n.y;
92     if (n.z < 0.)
93         alpha += n.z;
94
95     return alpha - (n.x + n.y + n.z)/2.;;
96 }
97 #else // dimension < 3
98 # define plane_alpha line_alpha
99 #endif

```

3. line area/plane area 函数

3.1 函数目的及原理

该函数的目的是在给定界面的法向量 n_x, n_y 以及界面参数 α 后，求解界面内在单元中的占比。

本算法的精髓在于**坐标变换**；依旧以 2 维为例，法向量的组合共有四种即：($n_x > 0, n_y > 0$), ($n_x > 0, n_y < 0$), ($n_x < 0, n_y > 0$), ($n_x < 0, n_y < 0$)，为了简便计算，我们应该在保证图形不发生改变的情况下使用坐标变换，将坐标中心移至左下角，并将法向量变为 ($n'_x > 0, n'_y > 0$)，如下图：

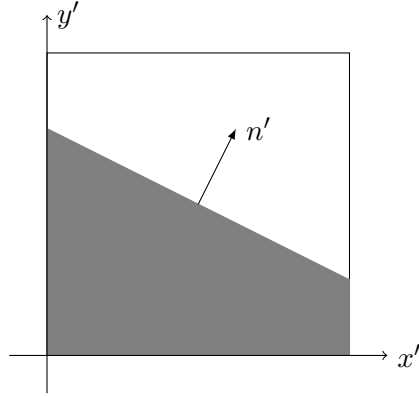


图 5: 坐标变换示例

其中 ' 表示坐标变换。根本原理如下：

有原方程式：

$$n_x x + n_y y = \alpha \quad (4)$$

现将坐标平移至左下，即：

$$\begin{cases} x' = x + \frac{1}{2} \\ y' = y + \frac{1}{2} \end{cases} \quad (5)$$

则原式变为：

$$n_x x' + n_y y' = \alpha + \frac{1}{2}(n_x + n_y) \quad (6)$$

接下来对法向量进行变换，当法向量两分量均大于零时可以跳过此步，当两分量中有一个小于零时，可以通过使直线关于 $x = \frac{1}{2}$ 或 $y = \frac{1}{2}$ 进行对称，该步骤具有叠加性，现取 $n_x < 0$ 为例：

$$\frac{1}{2} - x'' = x' - \frac{1}{2} \quad (7)$$

$$(1 - x'') = x' \quad (8)$$

有：

$$-n_x x'' + n_y y' = \alpha \frac{n_y - n_x}{2} \quad (9)$$

面积计算时，算法将情况同样分为了四种，如下图 我们将边界延长至与两对称轴相交，形成三角形，再用三角形减去多余部分即是体积占比，如下图：

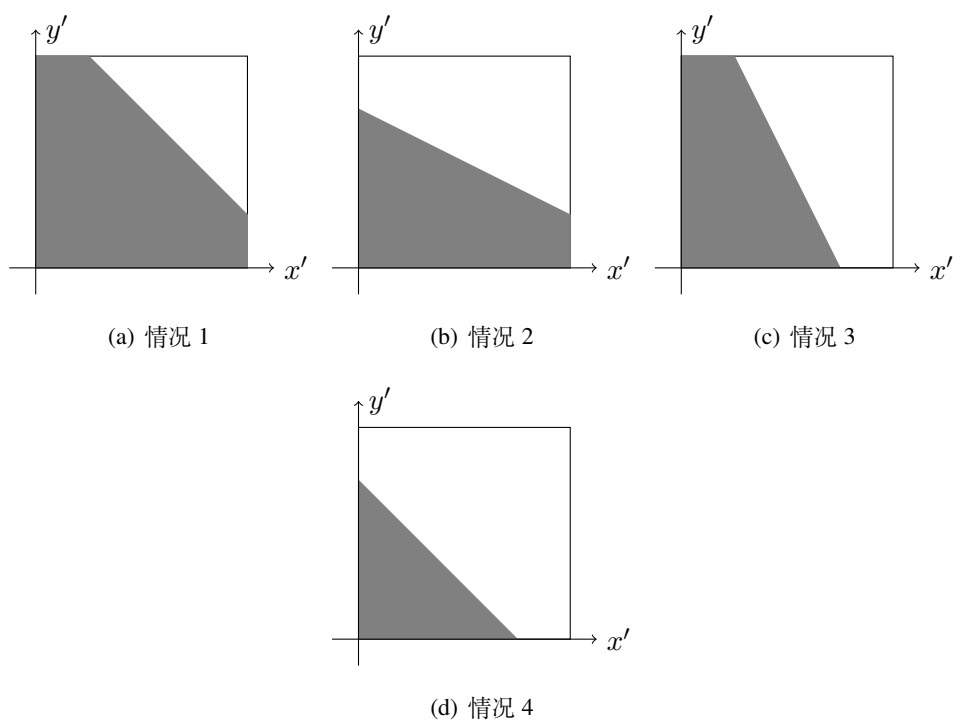


图 6: 面积计算中出现的四种情况

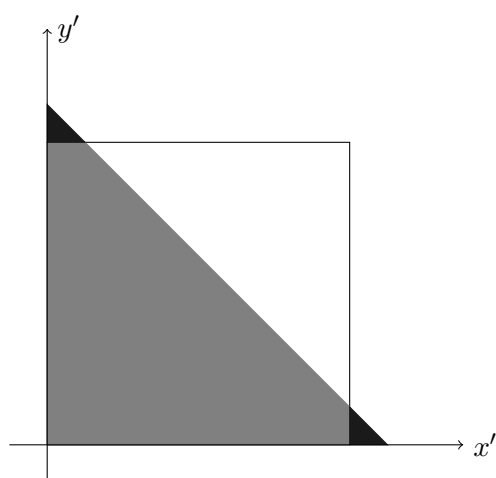


图 7: 面积计算示意

我们可以轻松计算得，大三角形的面积永远为：

$$S = \frac{\alpha^2}{2n'_x n'_y} \quad (10)$$

是故我们只需要判断，图形是否与 x', y' 轴的交点大于 1，并减去相应的面积即可，我们依旧以 x' 方向为例：

首先判断是否有多余面积，即若：

$$n'_x(x' - 1) + n'_y y' = \alpha - n'_x > 0 \quad (11)$$

则表明其有多余的面积部分，多余面积部分的面积为：

$$S_r = \frac{(\alpha - n'_x)^2}{2n'_x n'_y} \quad (12)$$

其余方向同理，减去即可。

需要注意的是，形状还有可能是标准的长方形，此时只需要判断斜率，并做简单计算即可。三维与二维的算法原理同理，在此不多做赘述。

3.2 具体实现代码

```

1  #if dimension >= 2
2  double line_area (double nx, double ny, double alpha)
3  {
4      double a, v, area;
5
6      alpha += (nx + ny)/2.; //坐标偏移至左下角
7      if (nx < 0.) { //判断法向量分量正负，通过坐标变换全部变为正值
8          alpha -= nx;
9          nx = - nx;
10     }
11     if (ny < 0.) {
12         alpha -= ny;
13         ny = - ny;
14     }
15
16     if (alpha <= 0.) //判断单元内是否有界面内部分
17         return 0.;
18
19     if (alpha >= nx + ny) //界面过点 (1,1) 为临界状态，代表整个单元均在界面内
20         return 1.;
21
22     if (nx < 1e-10) //判断是否为长方形
23         area = alpha/ny;

```

```

24     else if (ny < 1e-10)
25         area = alpha/nx;
26     else {
27         v = sq(alpha);
28
29         a = alpha - nx; //图形是否有多余的三角形面积
30         if (a > 0.)
31             v -= a*a;
32
33         a = alpha - ny;
34         if (a > 0.)
35             v -= a*a;
36
37         area = v/(2.*nx*ny);
38     }
39
40     return clamp (area, 0., 1.);
41 }
42 #endif // dimension >= 2
43
44 #if dimension >= 3
45 double plane_volume (coord n, double alpha)
46 {
47     double al = alpha + (n.x + n.y + n.z)/2. +
48         max(0., -n.x) + max(0., -n.y) + max(0., -n.z);
49     if (al <= 0.)
50         return 0.;
51     double tmp = fabs(n.x) + fabs(n.y) + fabs(n.z);
52     if (al >= tmp)
53         return 1.;
54     if (tmp < 1e-10)
55         return 0.;
56     double n1 = fabs(n.x)/tmp;
57     double n2 = fabs(n.y)/tmp;
58     double n3 = fabs(n.z)/tmp;
59     al = max(0., min(1., al/tmp));
60     double al0 = min(al, 1. - al);

```

```

61     double b1 = min(n1, n2);
62     double b3 = max(n1, n2);
63     double b2 = n3;
64     if (b2 < b1) {
65         tmp = b1;
66         b1 = b2;
67         b2 = tmp;
68     }
69     else if (b2 > b3) {
70         tmp = b3;
71         b3 = b2;
72         b2 = tmp;
73     }
74     double b12 = b1 + b2;
75     double bm = min(b12, b3);
76     double pr = max(6.*b1*b2*b3, 1e-50);
77     if (a10 < b1)
78         tmp = a10*a10*a10/pr;
79     else if (a10 < b2)
80         tmp = 0.5*a10*(a10 - b1)/(b2*b3) + b1*b1*b1/pr;
81     else if (a10 < bm)
82         tmp = (a10*a10*(3.*b12 - a10) + b1*b1*(b1 - 3.*a10) +
83 ^I    b2*b2*(b2 - 3.*a10))/pr;
84     else if (b12 < b3)
85         tmp = (a10 - 0.5*bm)/b3;
86     else
87         tmp = (a10*a10*(3. - 2.*a10) + b1*b1*(b1 - 3.*a10) +
88 ^I    b2*b2*(b2 - 3.*a10) + b3*b3*(b3 - 3.*a10))/pr;
89
90     double volume = a1 <= 0.5 ? tmp : 1. - tmp;
91     return clamp (volume, 0., 1.);
92 }
93 #else // dimension < 3
94 # define plane_volume(n, alpha) line_area(n.x, n.y, alpha)
95 #endif

```

4. rectangle fraction 函数

4.1 函数目的及原理

如下图：

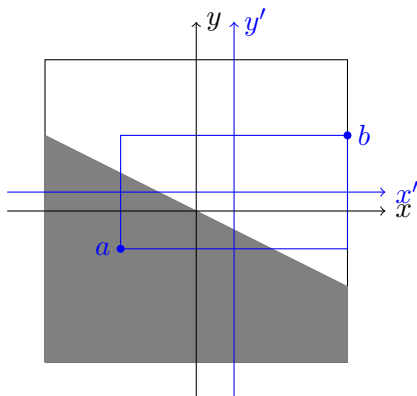


图 8: 偏移坐标变换示例

该函数的目的在于在原本单元中给定两点 a, b 构成一个新的直角单元网格，通过在原单元中的边界法向量 n ，以及边界参数 α 计算在变形单元中的面积/体积占比。

对原坐标进行坐标变换有

$$\begin{cases} x' = (x_b - x_a)x - \frac{x_b + x_a}{2} \\ y' = (y_b - y_a)y - \frac{y_b + y_a}{2} \end{cases} \quad (13)$$

即可得到在该坐标中：

$$\alpha' = \alpha - n_x \cdot (x_b + x_a)/2 - n_y \cdot (y_b + y_a)/2 \quad (14)$$

$$n' = (n_x \cdot (x_b - x_a), n_y \cdot (y_b - y_a)) \quad (15)$$

值得注意的是，改代码将直角单元长宽缩放转化为了正方形单元，我们再使用函数 `plane volume` 进行面积占比计算，这一操作并不会影响结果。

4.2 具体实现代码

```
1 double rectangle_fraction (coord n, double alpha, coord a, coord b)
2 {
3     coord n1;
4     foreach_dimension() {
5         alpha -= n.x*(b.x + a.x)/2.;
6         n1.x = n.x*(b.x - a.x);
7     }
8     return plane_volume (n1, alpha);
9 }
```

1111

单行代码展示 `double un = dt*uf.x[]/(fm.x[]*Delta + SEPS), s = sign(un);`