MInDes

##### **M**icrostructure **In**telligent **Des**ign Software

**快速指导手册**

（材料）微结构智能设计软件

**April, 2023**

Science center for phase diagram, phase transition, material intelligent design and manufacture, Central South University, China

相图、相变及材料智能设计与制备科学中心，中南大学，中国

##### **MInDes – a program for microstructure intelligent design software**

##### **Copyright (c) 2019-2023**

##### **Science center for phase diagram, phase transition, material intelligent design and manufacture, Central South University, China**

##### **This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.**

##### **This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.**

##### **You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.**

目录

[1. MInDes介绍 4](#_Toc134432920)

[2. 软件的配置安装、模拟及可视化 5](#_Toc134432921)

[2.1. 运行环境 5](#_Toc134432922)

[2.2. 软件安装 5](#_Toc134432923)

[2.3. 模拟流程及可视化 8](#_Toc134432924)

[2.4. 基本输入参数简介 14](#_Toc134432925)

[3. 模块介绍 18](#_Toc134432926)

[3.1. 微结构初始化 18](#_Toc134432927)

[3.2. 预处理 23](#_Toc134432928)

[3.3. 相界面 28](#_Toc134432929)

[3.4. 固体力学 33](#_Toc134432930)

[3.5. 流体力学 35](#_Toc134432931)

# MInDes介绍

**MInDes是由中南大学相图、相变及材料智能设计与制备实验室主任、教育部长江学者、973项目首席科学家——杜勇教授提出以耦合实际热力学、扩散、热物性、力学等数据库为目标的，耦合多物理场演化的介观微结构模拟软件。**

**MInDes采用C++高级程序语言搭建了基本程序框架。程序底层设置了必要的数据（函数）类型、运算方法、数据网格结构、程序核心循环、各物理场求解器和并行框架。程序中层是对接各求解器的接口模块，由研究人员进行模型、程序功能模块、数据库的二次开发设计。程序表层将在可视化界面上处理MInDes的输入（.mindes）、输出（.log、.vts、.dat和.txt等）等。MInDes可在windows、linux双系统上运行，将使用OpenMP、CUDA等并行库加速计算，使用差分法、傅里叶谱方法、格子玻尔兹曼法等对各物理场进行求解。**

**开发者：**

**MInDes程序的开发自2019年始，是中南大学相图、相变及材料智能设计与制备实验室在读博士黄奇（2018-2023）博士期间的主要成果。多年后续开发期间，许多科研工作者进行了贡献：**

**黄奇、…**

**手册更新历史：**

**2023年4月（第一版）、…**

**合作联系：**

**杜勇教授:** [yong-du@csu.edu.cn](mailto:yong-du@csu.edu.cn)**;**

**黄奇博士:** [qihuang0908@163.com](mailto:qihuang0908@163.com)**;**

# 软件的配置安装、模拟及可视化

## 运行环境

Windows:

64位（x64）系统。

Linux:

暂无。

## 软件安装

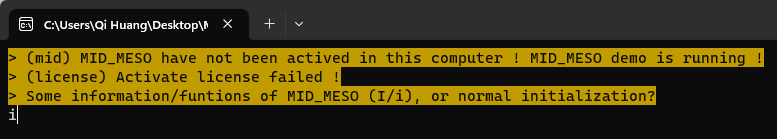
解压安装包



从左至右：指导文件、案例文件（.mindes）、动态库（.dll）、MInDes（linux系统执行文件）、exe运行文件（x64 windows系统执行文件）

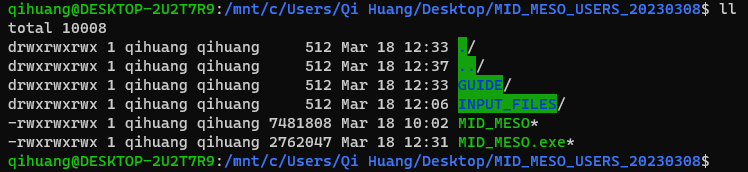
运行执行文件、进入软件配置界面

Windows系统下，双击打开MInDes.exe：

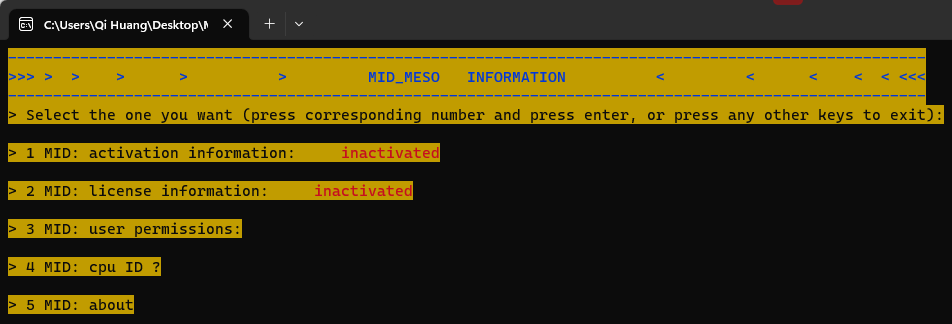


敲击键盘“i”并回车，进入配置界面。

Linux系统下，在该文件夹内执行MInDes，

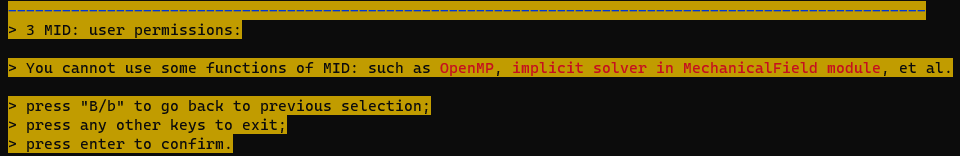


直接进入配置界面，总界面：

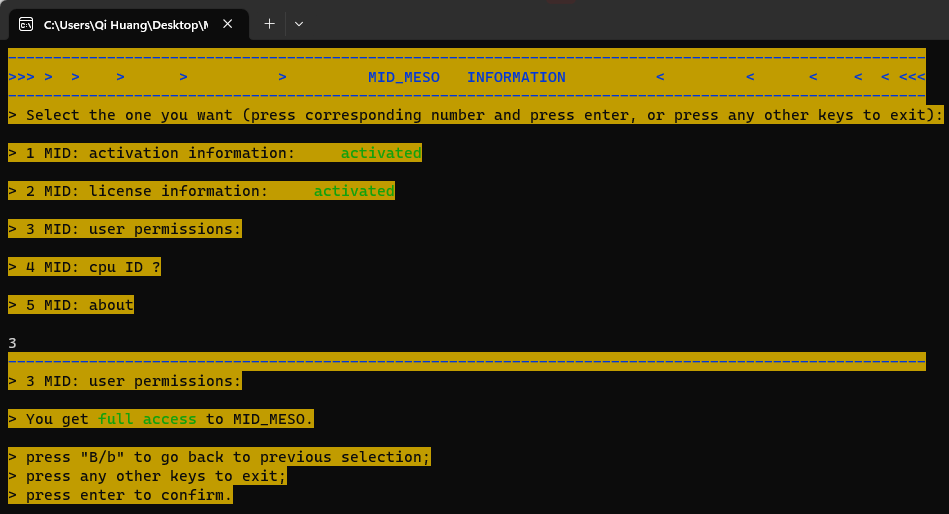


需要开发人员辅助进行软件激活及license生成，可在此界面查看激活信息（1）及license信息（2），具体请联系开发人员。

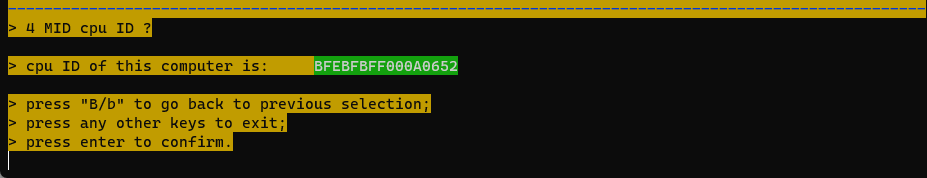
在此界面可查看当前软件权限（3），在未激活情况下不具有并行加速及部分高效算法（不影响功能/模型的常规使用）：



激活情况下可进行高效模拟：



激活文件及license文件绑定**个人计算机/服务器主节点**，可查看绑定ID信息，以在license到期后联系开发人员申请延续：

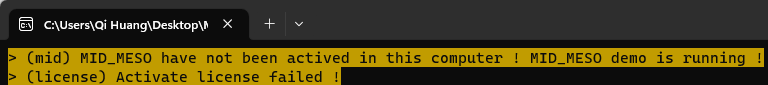


最后，可查看软件概况：



## 模拟流程及可视化

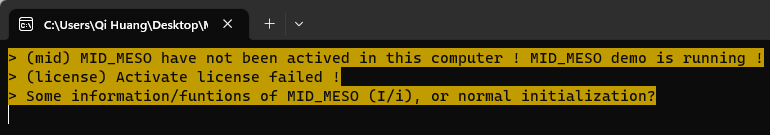
假设MInDes软件未激活：



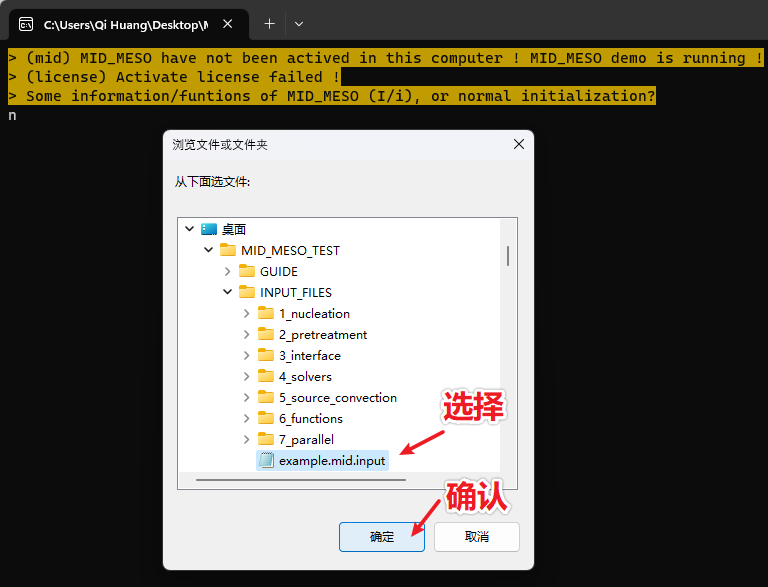
**读取.mindes文件**

Windows读取方法：

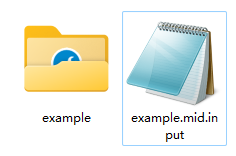
打开MInDes：



输入N/n，enter确认，弹出windows文件浏览系统，可在该界面找到你的.mid.input文件，比如INPUT\_FILES文件夹中的example.mid.input文件（其他的案例可以读取对应文件夹中的.mid.input文件）：



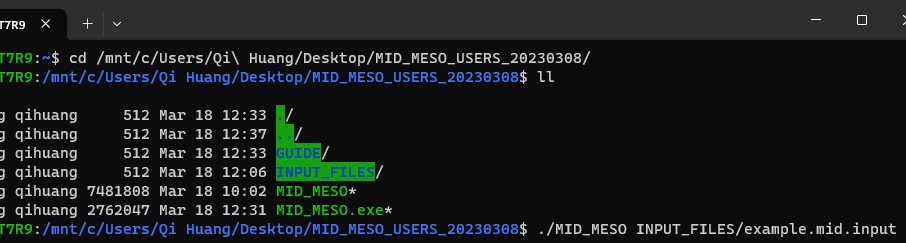
软件将自动读取该**输入文件**，并在**同目录下**生成**同名文件夹**用于保存**输出文件**：



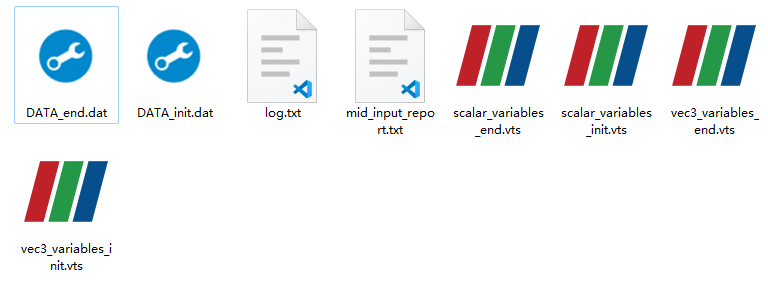
因此也需要注意，同目录下、同名文件夹内的同名输出文件文件会在输出过程中被覆盖，可通过修改该文件夹名称保留输出文件。

Linux读取方法：

定位到MInDes所在文件夹（必须在该文件夹内启动MInDes软件），在启动软件时，在后面输入读取.mid.input文件的索引，同样可读取INPUT\_FILES文件夹中的example.mindes文件：



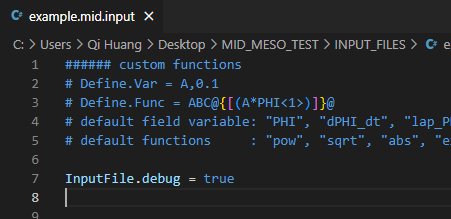
其他情况与Windows系统一致，可以看到example文件夹中默认存在四类文件：



保存模拟网格的所有信息的.dat文件、输出日志信息的log.txt文件、输入文件反馈的mid\_input\_report.txt文件和可视化数据分析的.vts文件。

**设置.mindes文件（mid\_input\_report.txt）**

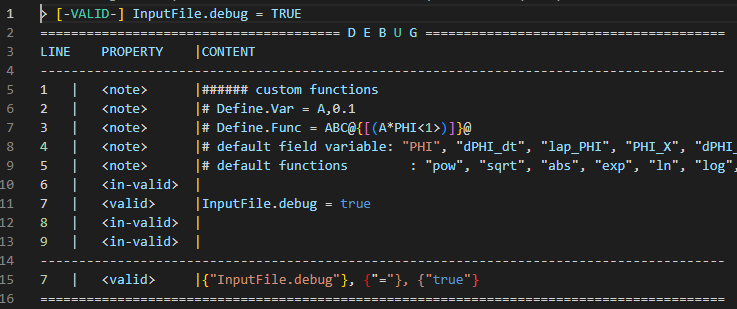
打开example.mid.input文件，观察到除了上方被注释的5行，仅有一行命令在起作用：InputFile.debug = true



命令行开头、末尾都不能空格，中间两个空格将命令行分为3部分，第一部分为命令行的钥匙（key）：“InputFile.debug”，第二部分统一都为等号：“=”，第三部分为命令行的值（value）：“true”。

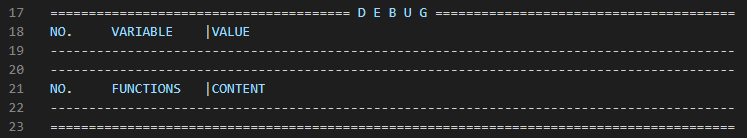
该命令行的功能是对该输入文件进行测试，并在输出文件夹中输出mid\_input\_report.txt文件，使用者可通过mid\_input\_report.txt文件来测试、检查.mid.input文件的书写情况。

如，输出文件夹example中的mid\_input\_report.txt是一个89行的文本文件，其中第一部分（2-16）：

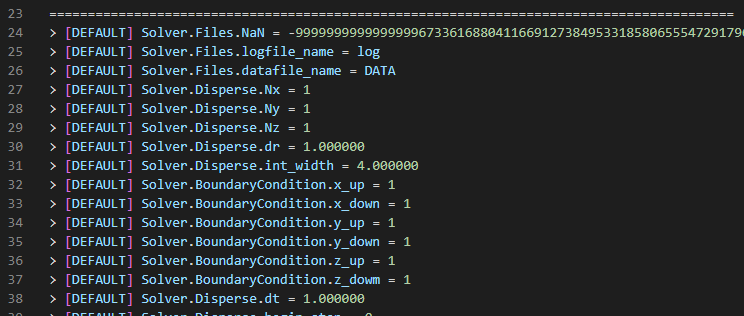


第5-13行读取example.mid.input文件中所有文本，包括空行等。这些文本会按行被分为3类：注释<note>、有效行<valid>、无效行<in-valid>，注释及无效行会被忽视，有效行会被软件分配管理（如第15行）。

同样我们可以在输入文件中定义变量和函数，以辅助进行一些复杂的输入，这些定义在第17-23行中被测试：



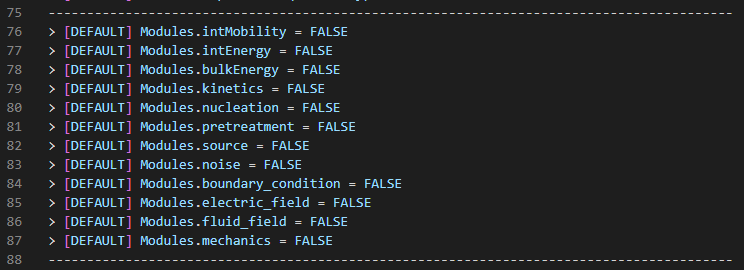
然后第23-75行，我们可以看到主求解器所需的输入命令：



命令前方的**[DEFAULT]**代表没有读取到这个命令，因此采用默认值。类比读取到的命令，前方为**[-VALID-]**：



最后第75-88行，我们可以看到所有的模块默认都是关闭状态：

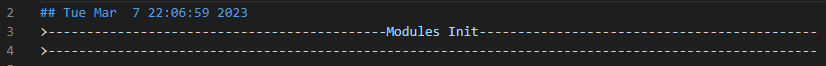


开启模块可以解锁更多的默认命令行，即更多的功能。因此使用者可以通过mid\_input\_report.txt及已有的案例来反复测试、修改输入文件，以达到想要的效果。

**解读日志信息（log.txt）**

日志信息会直接被输出在控制台上（不论是Windows还是Linux），同时也会以文件的形式（log.txt）存储在输出文件夹中。例如输出文件夹example中的log.txt：

第2-4行，为模块初始化过程中的输出（一般模块在这一步解析使用者输入的命令行），因example.mid.input没有使用模块，所以这里没有反馈



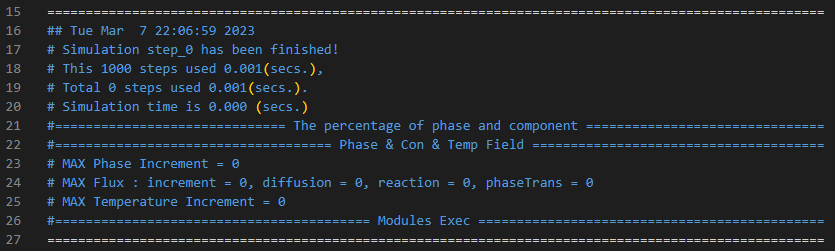
第6-8行，开启软件的并行的功能，因假设没有激活软件，这里会被限制为单线程求解



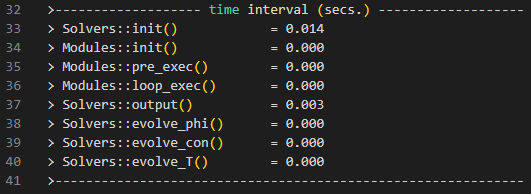
第11-13行，为模块在主循环开始前的预处理，其反馈会输出到这里



之后为主循环，这里会每隔固定步数输出统计的信息，以帮助使用者了解软件运行情况。同时，一些物理场管理模块在主循环运行过程中的求解信息也会输出在统计信息的下方



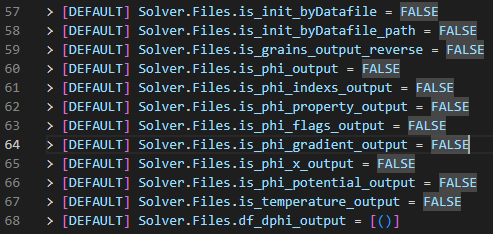
最后，会统计模拟各阶段的耗时，以辅助开发者进行程序高效运行设计



**可视化（.vts）**

需要安装paraview软件辅助MID\_MESO进行可视化。

在主求解器和各个模块中，有相应的命令来控制一些数据的输出：



在输出的vts中，scalar代表输出数据为标量，vec3代表输出数据为向量



打开vts文件后，会自动来到paraview界面，或者可以在paraview软件中批量打开vts文件，以形成动图。Paraview的细节操作可自行学习。

**再（套嵌）模拟（.dat）**

输出的.dat文件包含模拟网格中几乎所有的信息，使用者可以通过命令来控制是否通过.dat文件来初始化一个模拟



在初始化过程中，数据网格保留了上个模拟域大小、单网格大小、相信息、组分信息等，因此在重新开始新的模拟时需要注意模拟网格、相和组分等信息需要匹配，否则将导致难以预料的后果。

在INPUT\_FILES/3\_interface文件夹中的multigrains\_junction.mid.input案例就用到了该方法，可用作参考。在Windows下也可通过文件浏览系统索引需要的.dat文件。

## 基本输入参数简介

Solver.Files.NaN = 极大/小的一个数

在paraview对数据进行展示时，无限大的数可以被自动剔除（即NaN的效果），效果为只展示特定区域的数据。

Solver.Files.logfile\_name = log

日志文件名称

Solver.Files.datafile\_name = DATA

二进制数据文件名称

Solver.Disperse.Nx = 1

模拟网格x方向网格数

Solver.Disperse.Ny = 1

模拟网格y方向网格数

Solver.Disperse.Nz = 1

模拟网格z方向网格数

Solver.Disperse.dr = 1.000000

模拟网格的网格空间长度

Solver.Disperse.int\_width = 4.000000

相界面宽度

Solver.BoundaryCondition.x\_up = 1

Solver.BoundaryCondition.x\_down = 1

Solver.BoundaryCondition.y\_up = 1

Solver.BoundaryCondition.y\_down = 1

Solver.BoundaryCondition.z\_up = 1

Solver.BoundaryCondition.z\_dowm = 1

边界条件：0 – 固定, 1 – 周期, 2 – 绝热

Solver.Disperse.dt = 1.000000

时间步长

Solver.Disperse.begin\_step = 0

Solver.Disperse.end\_step = 0

开始、结束模拟步

Solver.Disperse.Auto\_dt = FALSE

是否自动调整时间步长

Solver.Disperse.Auto\_dt.dstep = 1000

每多少模拟步进行一次调整

Solver.Disperse.Auto\_dt.max\_scale = 1000.000000

时间步长最大放大倍数

Solver.Disperse.Auto\_dt.show\_dt\_change = FALSE

是否在模拟过程中展示时间步长的改变

Solver.DifferenceMethod = 0

Phi，c，T差分法求解的差分方法：0 – 五点差分， 1 – 九点差分

Solver.PhaseField.normalize\_phi = FALSE

Phi求解过程中是否归一化处理

Solver.Parallel.OMP\_thread\_counts = 1

并行线程设置

Solver.PhaseField.increment\_limit = 0.001000

Solver.ConcentrationField.increment\_limit = 0.001000

Solver.TemperatureField.increment\_limit = 0.001000

Phi，c，T的变化量限制设置，与自动调整时间步长一起使用有效

Solver.Files.screen\_output\_step = 1000

Solver.Files.vts\_output\_step = 1000

Solver.Files.data\_output\_step = 1000

屏幕统计信息、vts文件、dat数据文件输出，每多少模拟步一输出

Solver.Files.is\_init\_byDatafile = FALSE

Solver.Files.is\_init\_byDatafile\_path = FALSE

模拟是否以一个dat文件进行初始化，以及初始化的路径

Solver.Files.is\_grains\_output\_reverse = FALSE

Solver.Files.is\_phi\_output = FALSE

Solver.Files.is\_phi\_indexs\_output = FALSE

Solver.Files.is\_phi\_property\_output = FALSE

Solver.Files.is\_phi\_flags\_output = FALSE

Solver.Files.is\_phi\_gradient\_output = FALSE

Solver.Files.is\_phi\_x\_output = FALSE

Solver.Files.is\_phi\_potential\_output = FALSE

Solver.Files.is\_temperature\_output = FALSE

Solver.Files.df\_dphi\_output = [()]

Vts文件输出的数据控制，控制与phi，c，T相关的数据的输出

Solver.ConcentrationField.component = ()

定义模拟体系内所有的组分

Solver.PhaseField.property = {}

定义模拟体系内的所有相信息

Solver.AllenCahnEquation.Type = 0

相场方程的类型：0 – 不演化, 1 – 标准AC方程, 2 – 反对称AC方程

Solver.CahnHilliardEquation.Type = 0

浓度场方程类型：0 – 不演化, 1 – 标准CH方程, 2 – 相浓度, 3 – 巨势

Solver.TemperatureEquation.Type = 0

浓度场方程类型：0 - 不演化, 1 – 标准温度场方程

# 模块介绍

## 微结构初始化

微结构初始化由模块nucleation控制，打开该模块即可进行微结构初始化设计：

Modules.nucleation = true

**首先使用者需要声明微结构的基体，所有的结构都会被初始化在这个基体上（按作画所需的画布理解）：**

Modules.Nucleation.matrix = {[(相索引),(相属性),(相组分1,相组分2,…)],[(组分1,组分2,…)],[(温度)]}

**接着可以在基体上生成几何体，每个几何体位于一个几何层中，几何层代表了生成几何体的先后顺序，后生成的几何体会覆盖先生成的几何体（如同作画过程中的图层一般），因此可以通过设计几何层形成层次感：**

Modules.Nucleation.geometry\_layer\_number = 几何层数

然后，使用者需要声明几何层的一些属性。如仅有一层几何层，只需要声明第一层的属性（按0开始排序）

Modules.Nucleation.geometry\_layer\_0.property = (相索引, 相属性,几何类型,生成步,旋转规则,反向填充)

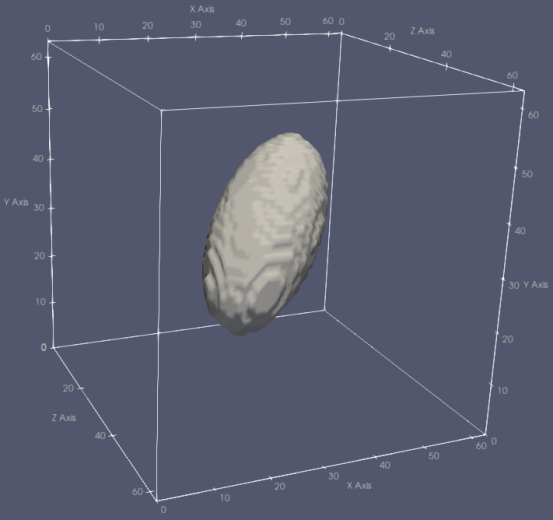
相的索引用于区分每个晶粒，对应、、等，而相的属性可以认为在某些方面具有相同性质的晶粒。如具有相同的热力学吉布斯自由能函数，具有相同弹性模量等，可用于进一步分类。几何类型目前支持椭球、多面体的生成。生成步与无量纲模拟步对应，一般情况下设置为第0步，即在模块预处理过程中就会生成结构。旋转规则可用于二维、三维结构的旋转，一般取几何体的相对坐标轴作为旋转参照物。反向填充即为填充几何体外的部分。

初始化一个椭球/椭圆

如果在设置几何层时，定义几何类型为椭球，使用者就需要在该几何层上进一步定义椭球的参数

Modules.Nucleation.geometry\_layer\_0.ellipsoid = [(球心x, 球心y, 球心z),(x轴半径, y轴半径, z轴半径),(旋转角1, 旋转角2, 旋转角3)]

案例见INPUT\_FILES/1\_nucleation/ellipse.mid.input 及 INPUT\_FILES/1\_nucleation/ellipsoid.mid.input

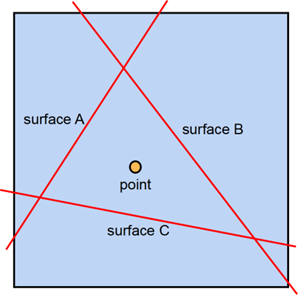


初始化一个多边形/多面体

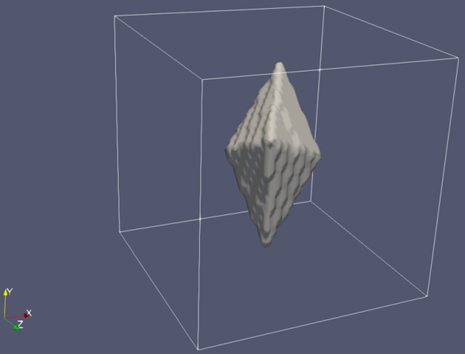
如果在设置几何层时，定义几何类型为多面体，使用者就需要在该几何层上进一步定义多面体的参数

Modules.Nucleation.geometry\_layer\_0.polyhedron = {[(体内点)],[多面体的面1],[多面体的面2],…,[旋转角1, 旋转角2, 旋转角3]}

采用面切割+体内点锚定的方法确定一个多面体/多边形，该方法也可轻易拓展到Voronoi结构生成逻辑中。

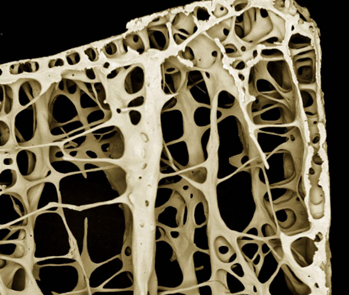


案例见INPUT\_FILES/1\_nucleation/ellipse.mid.input 及 INPUT\_FILES/1\_nucleation/ellipsoid.mid.input



**除了几何体形核之外，还可以根据图片（bmp24格式）来初始化一个更加复杂/自然的结构：**

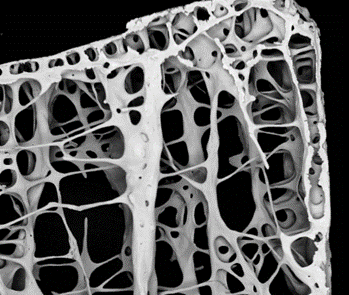
比如面对下图的复杂结构，



调用bmp24方法，

Modules.Nucleation.bmp24.property = ( 相索引个数, 生成步, 于input文件的相对路径)

该方法会读取bmp24格式的图片，并将其转化为灰度图片，如图



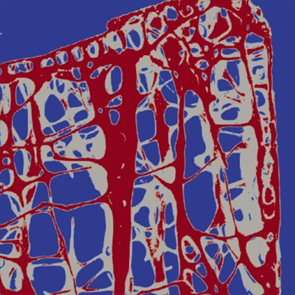
相索引的个数为从该图片生成相的个数，这里定义的个数会自动读取对应的相信息及灰度值范围，然后根据对应灰度值范围0~1（黑~白）在模拟域的相对空间位置上生成对应的相，如第一个相的定义如下

Modules.Nucleation.bmp24\_0.phi = ( 相索引, 相名称)

Modules.Nucleation.bmp24\_0.gray\_threshold = ( 灰度下限, 灰度上限)

这里定义了第一个相的相索引跟相的名称，这里相的名称需要在求解器中被定义，这里才能使用。灰度值的区间则需要认为设计，以生成想要的相结构。

具体案例可见INPUT\_FILES/1\_nucleation/read\_bmp24.mid.input，通过假设两个相，在背景相（蓝色）中生成第一个相（红色）及第二个相（白色）。



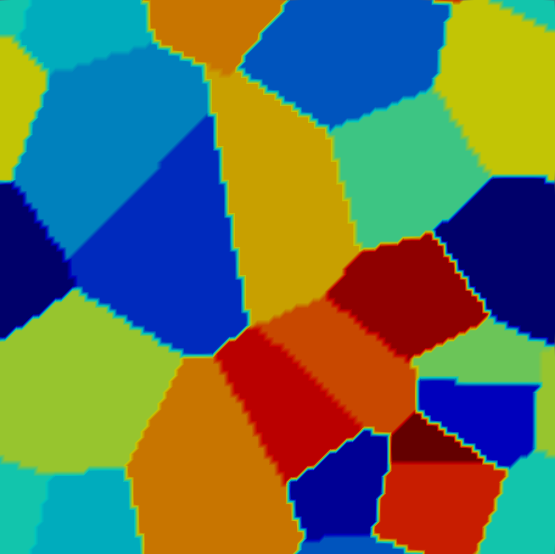
**Voronoi结构是相场模拟中较为常见的结构，可通过如下命令快速生成一个voronoi结构：**

Modules.Nucleation.voronoi.box = [( 盒子x轴起始, 盒子y轴起始, 盒子z轴起始), ( 盒子x轴终止, 盒子y轴终止, 盒子z轴终止)]

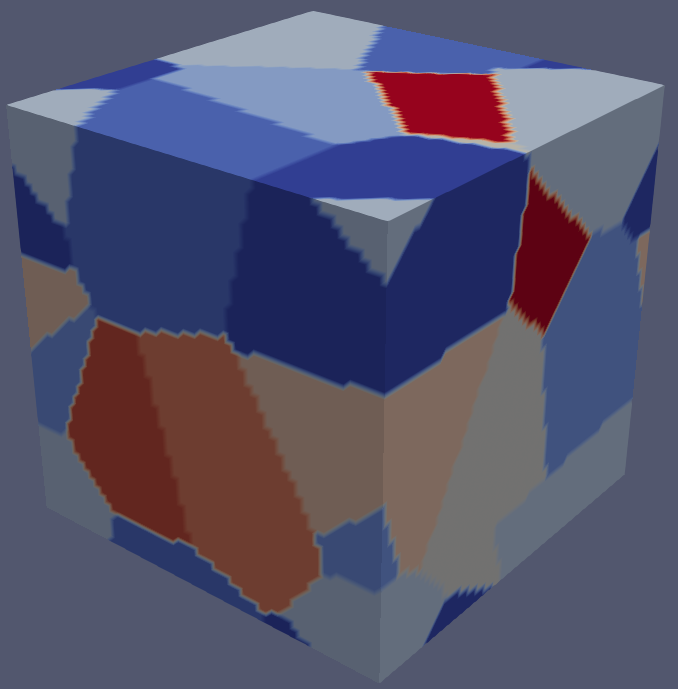
Modules.Nucleation.voronoi.property = [( 相索引起始, 相索引终止), ( 相名称, …),( 相的权重, …),(生成步)]

其中盒子为生成voronoi结构的区域，可以定义为二维、三维区域。在定义的区域内会随机撒点生成晶粒，晶粒数量由相索引起始和终止决定。各个晶粒具体分别属于什么相，由相名称及权重来随机分配。

具体二维案例见INPUT\_FILES/1\_nucleation/voronoi2d.mid.input

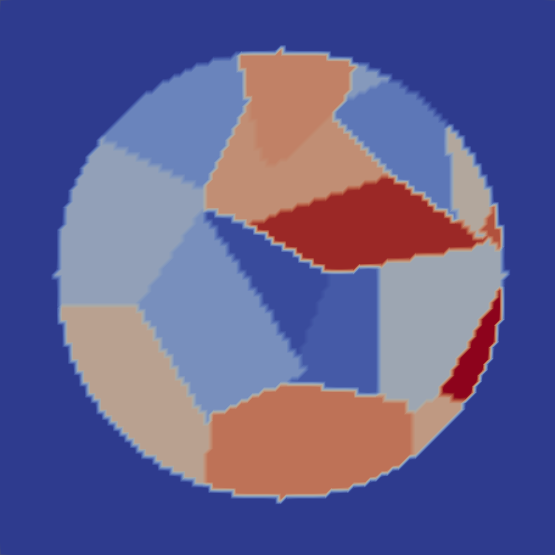


三维案例见INPUT\_FILES/1\_nucleation/voronoi3d.mid.input



**形核功能的叠加：例生成椭圆中的Voronoi**

见INPUT\_FILES/1\_nucleation/voronoi2d\_ellips.mid.input



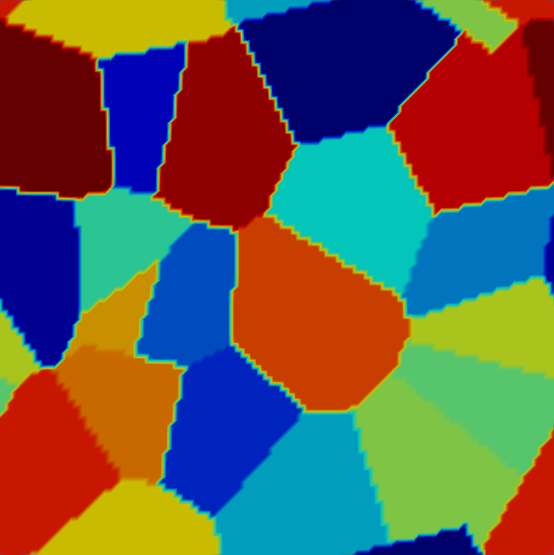
## 预处理

在预处理模块中集成了一些辅助模拟功能。

**相混合**

案例见INPUT\_FILES/2\_pretreatment/merge\_phis.mid.input

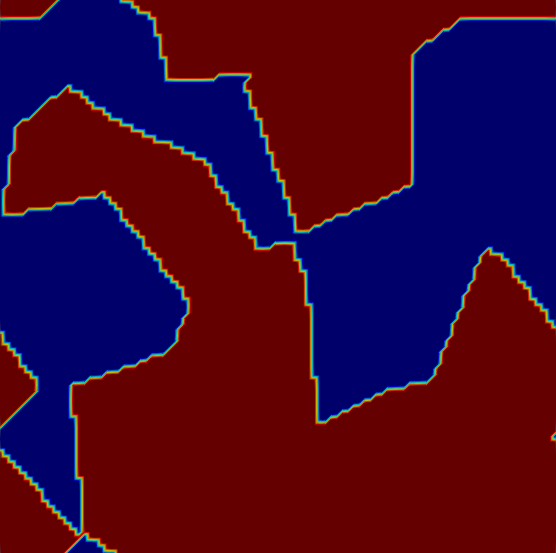
针对如下图所示随机生成的20个晶粒，可以将其中部分晶粒合并（在不影响模拟结果的情况下，可以节约计算内存，提高计算效率）



使用命令：

Modules.Pretreatment.merge\_phis = {[( 相索引1, 相索引2, …),(是否混合组分)], …}

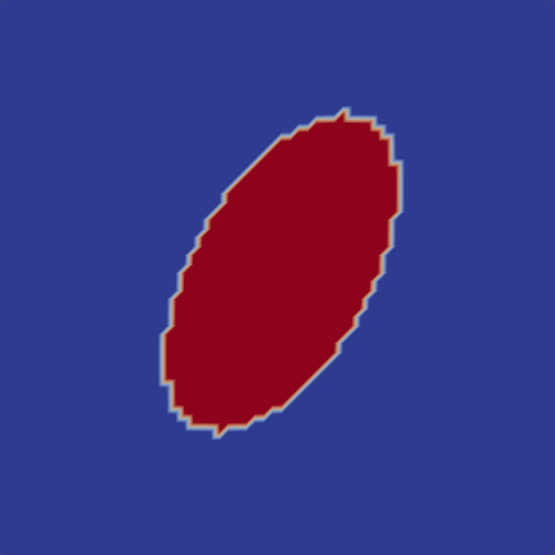
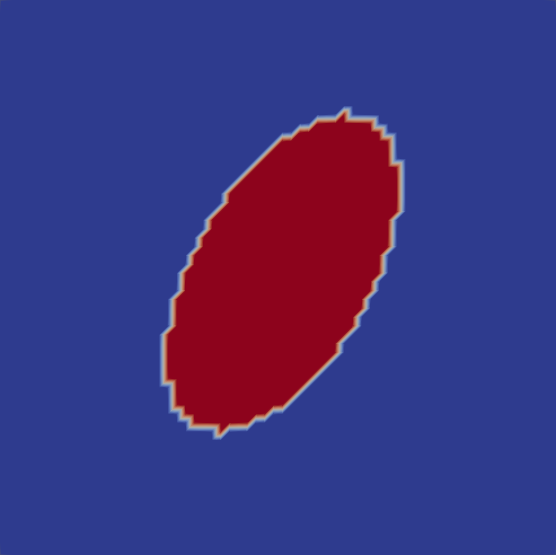
可将随机生成的20个晶粒混合为两个晶粒：



**相填充**

案例见INPUT\_FILES/2\_pretreatment/fill\_phis.mid.input

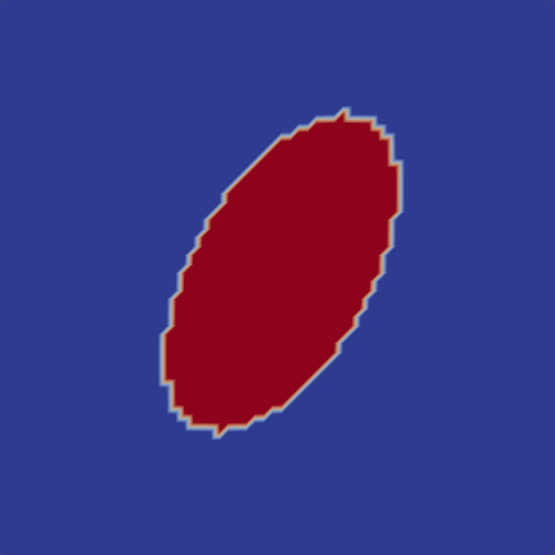
该功能可在已生成结构中填充组分及温度，也可修改对应索引相的相名称，比如对于如下结构（按顺序：相索引、相名称、浓度、化学势、温度）（红色值为1.0，蓝色为0.0）：

、、、、

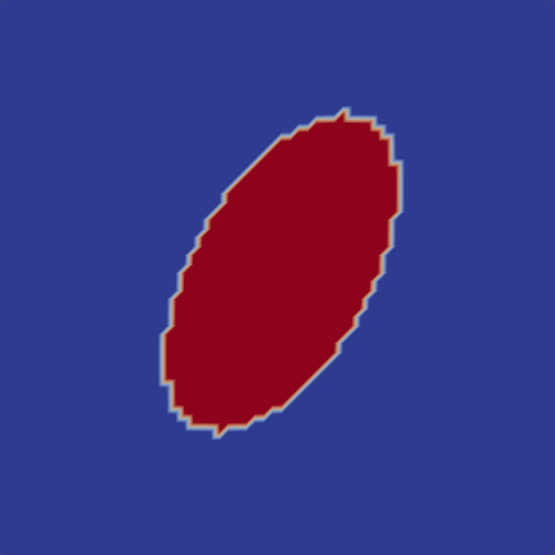
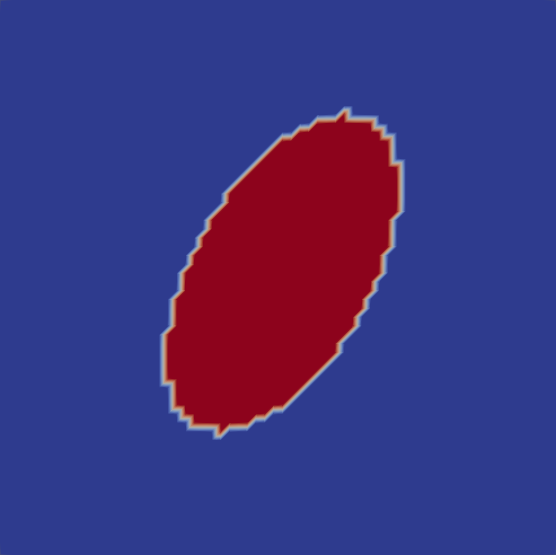
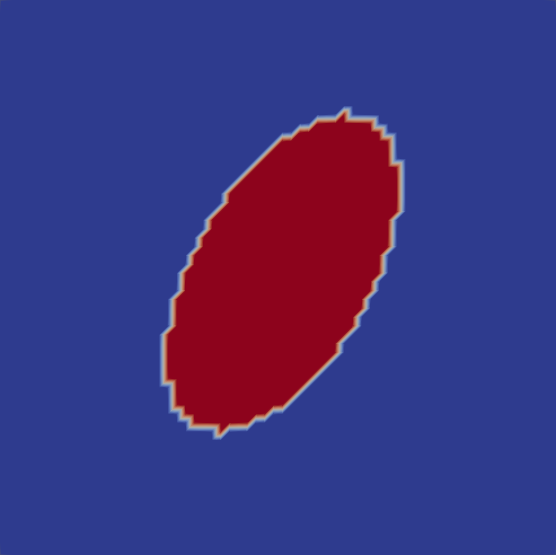
使用命令：

Modules.Pretreatment.fill\_phis = {[(相索引),(相名称),(相组分, …),(组分, …),(化学势, …),(温度)]}

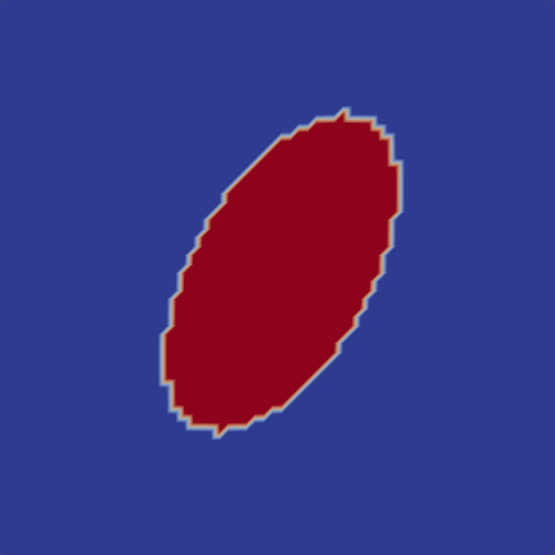
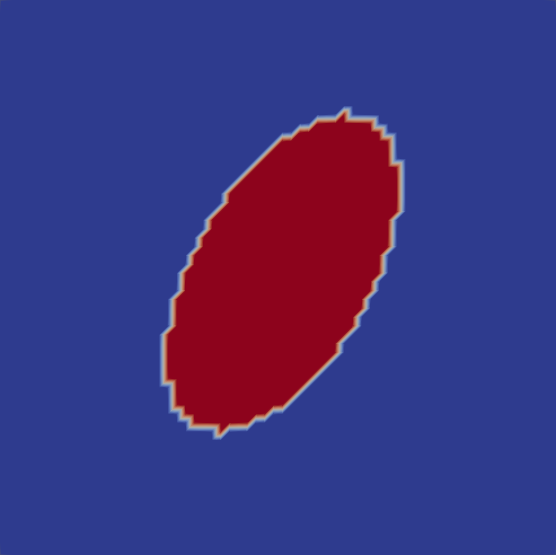
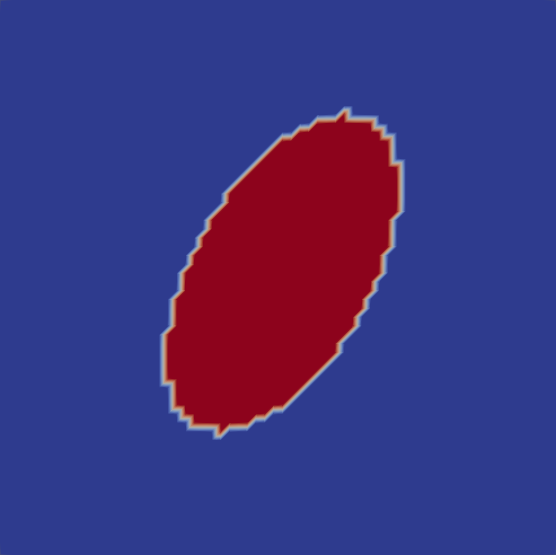
修改晶粒1的相名称与晶粒0相同：

、、、、

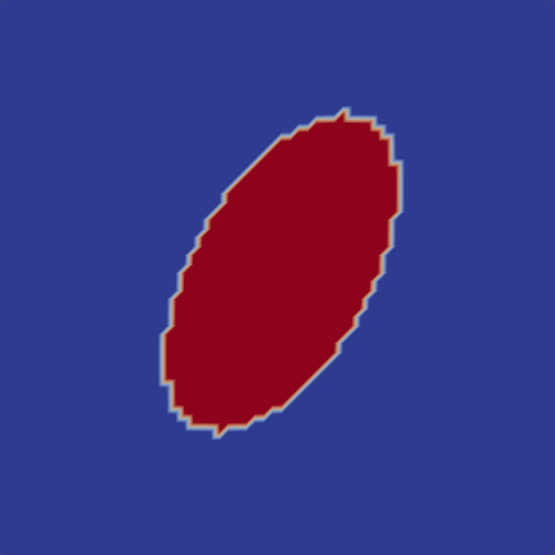
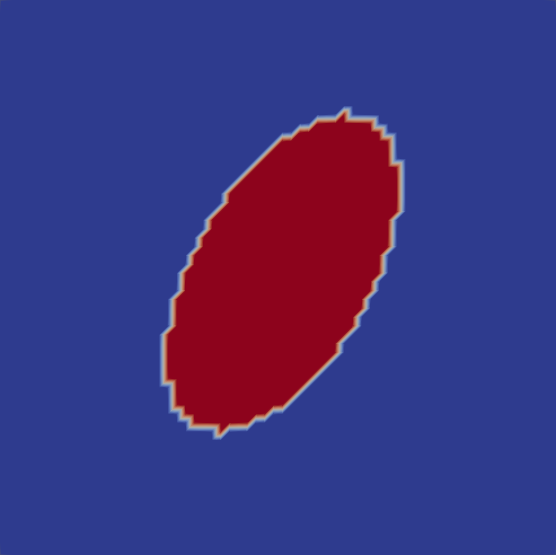
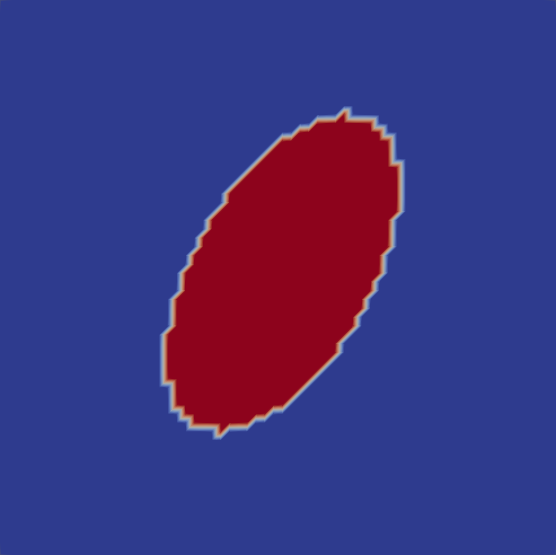
修改晶粒1内组分的浓度为1.0：

、、、、

修改晶粒1内组分的化学势为1.0：

、、、、

修改晶粒1内的温度为1.0：

、、、、

以配合初始化/再处理微结构。

**弛豫界面**

案例见INPUT\_FILES/2\_pretreatment/relax\_interface.mid.input

相场模型计算得到的是扩散界面，在相场法中，部分数值问题源自于还没有形成一个扩散的界面，该功能可在模拟前迭代弛豫出一个扩散的界面，以满足部分需求，比如对于一个具有尖锐界面的多边形结构



通过命令：

Modules.Pretreatment.relax\_interface = (迭代步数, 提示步数, 弛豫后界面是否锁定)

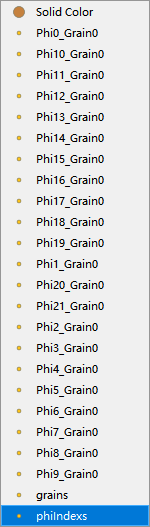
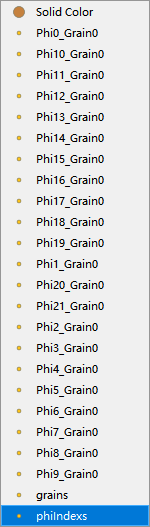
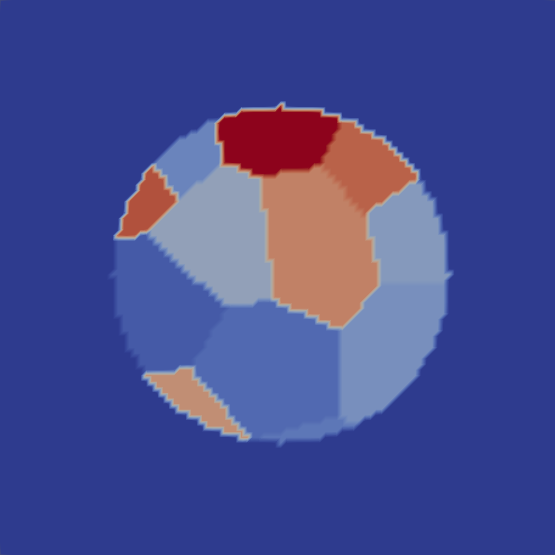
弛豫结果为：



**剔除不存在的相**

案例见INPUT\_FILES/2\_pretreatment/remove\_inexistence\_phis.mid.input

在构造复杂结构时难免会有相被覆盖或者消失，为节约内存、提高计算效率，剔除不存在的相是有必要的。以在椭圆中生成voronoi结构为例：

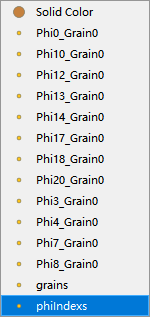
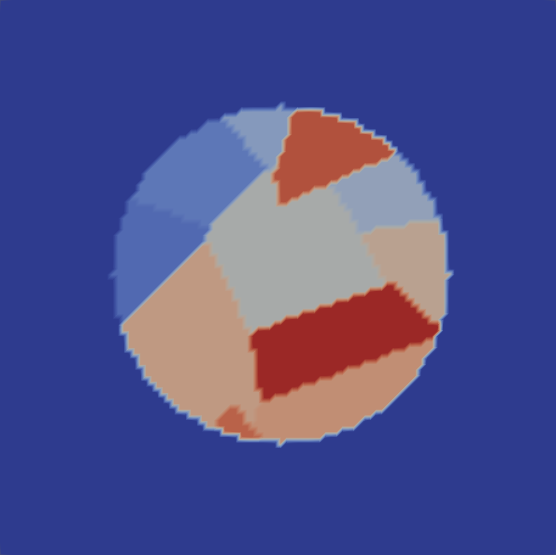


可见在该结构中存在22个晶粒，然而其中大部分在椭圆形核过程中被覆盖了，是不存在的，因此可使用该功能剔除这些晶粒。

通过命令：

Modules.Pretreatment.remove\_inexistent\_phis = true

因此相体积分数为0.0的晶粒会被自动剔除，结果只剩下12个晶粒：



## 相界面

相界面的形成、演化由模块intEnergy、intMobility控制，打开这两个模块即可设置形成扩散相界面的参数：

Modules.intEnergy = true

Modules.intMobility = true

**一维、二维相界面**

首先，可以选择界面能的模型：

Modules.InterfaceEnergy.int\_gradient = 梯度能模型

Modules.InterfaceEnergy.int\_potential = 势能模型

梯度能：0 - Steinbach\_1996, 1 - Steinbach\_1999, 2 - Steinbach\_G2009

势能：0 - Nestler\_Well, 1 - Nestler\_Obstacle, 2 - Steinbach\_P2009

Modules.Mobility.type = 迁移率形式

迁移率形式：0 - MOBILITY\_CONST, 1 - MOBILITY\_MATRIX

Modules.Mobility.Lij.const = 任意两相迁移率的值

Modules.Mobility.Lij.matrix = [(相索引1, 相索引2, 两相迁移率的值), …]

Modules.InterfaceEnergy.xi\_ab.type = 两相结能量形式

界面能形式：0 - XI\_CONST, 1 = XI\_MATRIX

Modules.InterfaceEnergy.xi\_ab.const = 任意两相界面能的值

Modules.InterfaceEnergy.xi\_ab.matrix = [(相索引1, 相索引2, 两相界面能的值), …]

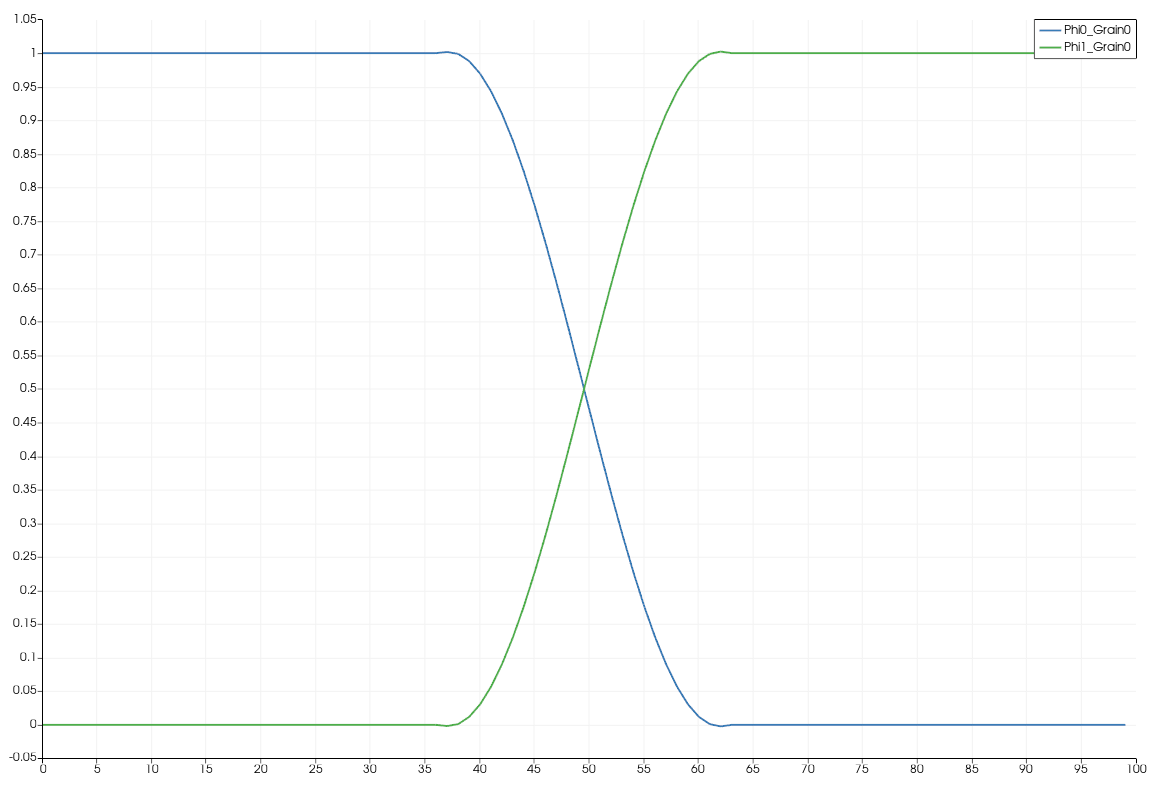
Modules.InterfaceEnergy.xi\_abc.type = 0

界面能形式：0 - XI\_CONST, 1 = XI\_MATRIX

Modules.InterfaceEnergy.xi\_abc.const = 任意三相界面能的值

在设置完界面能模型、界面能数值、界面迁移率后，如果存在相界面（形核过程生成的多相及相界面），即可对相界面进行演化，此过程一维、二维、三维统一：

案例见INPUT\_FILES/3\_interface/diffuse\_interface\_1d.mid.input



案例见INPUT\_FILES/3\_interface/diffuse\_interface\_2d.mid.input



**多相结**

案例见INPUT\_FILES/3\_interface/multigrains\_junction.mid.input

通过命令：

Modules.InterfaceEnergy.xi\_ab.type = 1

Modules.InterfaceEnergy.xi\_ab.matrix = [(0,1,1.0),(0,2,1.2),(1,2,0.6)]

控制两相界面能不同，不同的界面夹角：



**给一个界面驱动**

相场框架下，界面的驱动力由体相能及组分势来驱动，所以首先需要打开

Modules.bulkEnergy = true

然后通过设置简单能量密度差异，提供界面驱动力

Modules.BulkEnergy.type = 体相能的形式

体相能的形式：0 - BEType\_CONST, 1 - BEType\_PHI\_INDEPENDENT, 2 - BEType\_MIXED

Modules.BulkEnergy.dfdphi.const = [(相名子, 相的能量密度)]

通常能量密度低的相更容易长大，两相能量密度的差异为相变驱动力。

Modules.BulkEnergy.function\_type = 各相的能量函数形式

各相的能量函数形式：0 - BEFType\_Polynomial\_1X, 1 - BEFType\_Custom

Modules.BulkEnergy.dfdphi.polynomial\_1x = [(参数a, 参数b, 参数c), …]

Modules.BulkEnergy.dfdphi.custom = (自定义能量密度函数, …)

Modules.BulkEnergy.dfdx.custom = [(自定义化学势函数, …)), …]

Modules.BulkEnergy.dfdphi.interpolation\_function = 插值函数形式

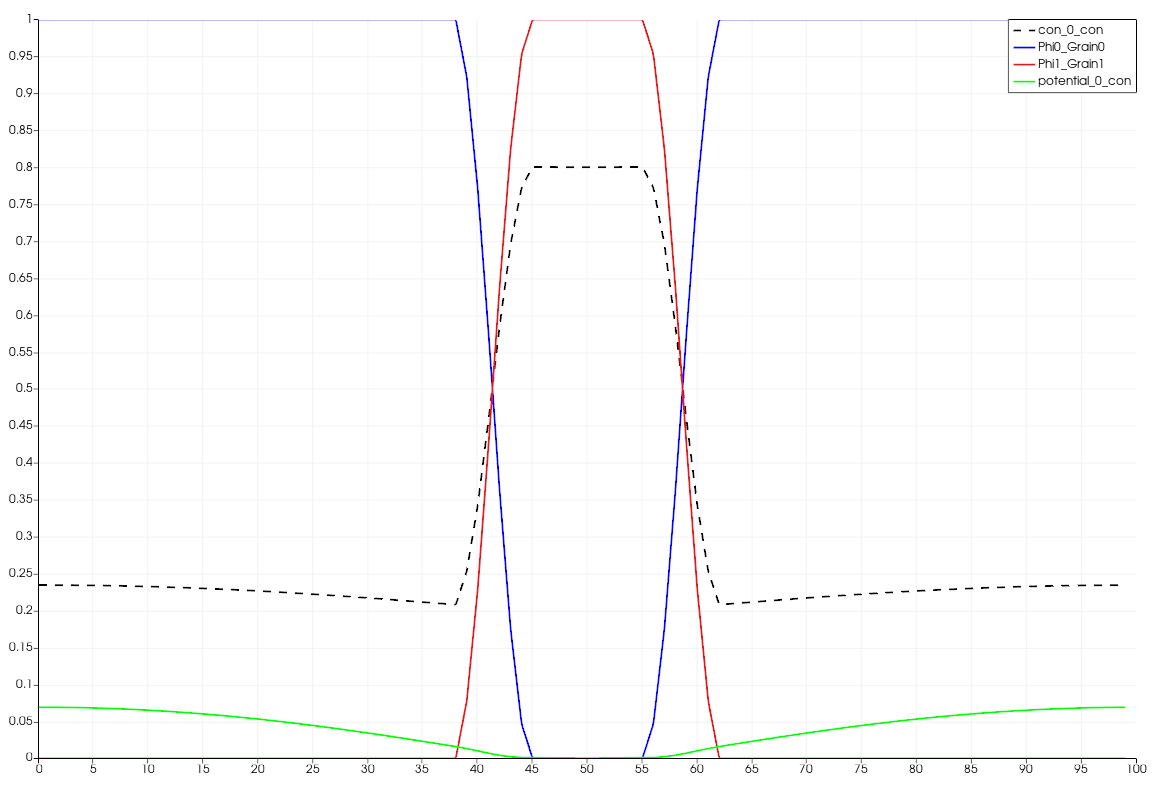
插值函数的形式：0 - IFType\_SIMPLE

案例见INPUT\_FILES/3\_interface/const\_driving\_force\_1d.mid.input

及INPUT\_FILES/3\_interface/const\_driving\_force\_2d.mid.input相变过程动图，

见INPUT\_FILES/6\_functions/phase\_x\_default\_funcs.mid.input

及INPUT\_FILES/6\_functions/phase\_x\_custom\_funcs.mid.input的相、组分热力学平衡：



其中蓝色、红色实线为两相相分数分布，黑色虚线为组分的分布，绿色实线为组分的化学势。组分在化学势的驱动下扩散，相变驱动力函数由能量密度、组分及组分化学势构成。

**界面上的相浓度**

浓度场的演化受动力学模块kinetics控制，需先打开该模块：

Modules.kinetics = true

然后需要定义动力学参数

Modules.Kinetics.Dij\_a.type = 扩散系数类型

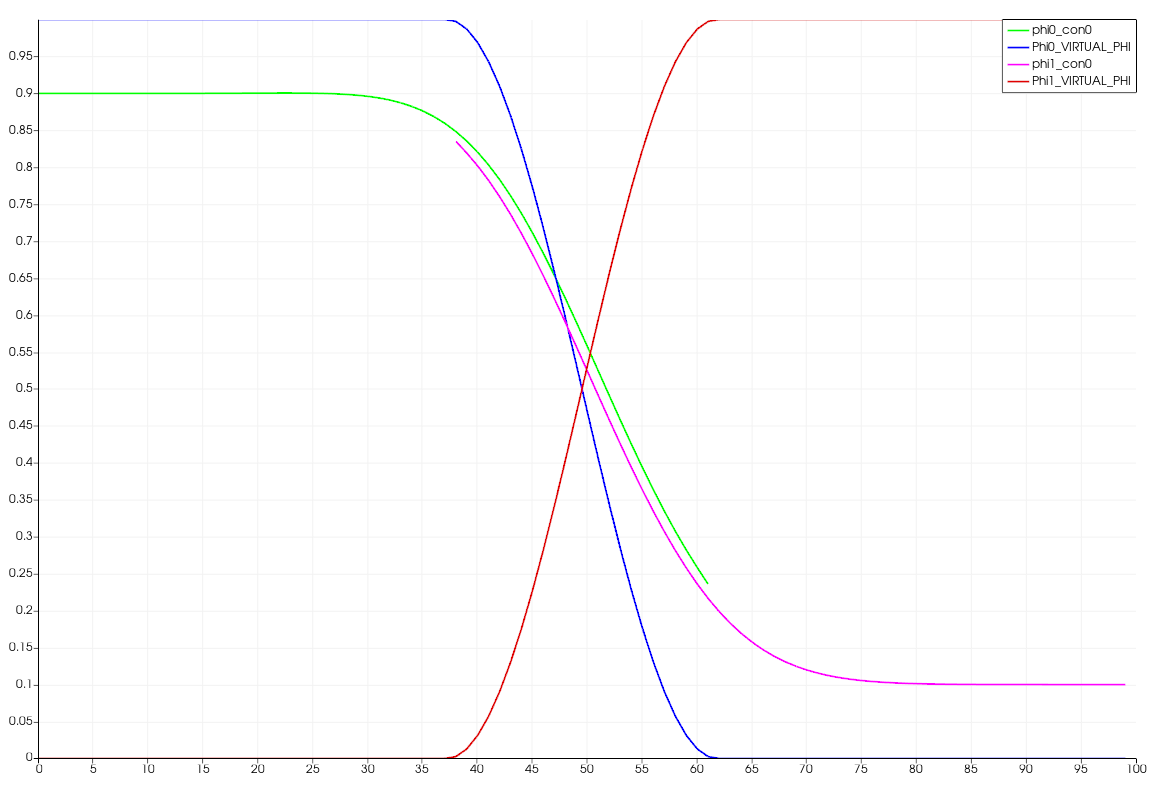
扩散系数类型：0 - D\_CONST, 1 - D\_MATRIX

Modules.Kinetics.Dij\_a.const = (相1的扩散系数)

Modules.Kinetics.Kab\_i = [(相索引1, 相索引2, 组分名, 耗散速率)]

案例见INPUT\_FILES/3\_interface/interface\_dissipation\_1d.mid.input 及

INPUT\_FILES/3\_interface/interface\_dissipation\_2d.mid.input相浓度的演化：



其中蓝色、红色实线为两相相分数分布，绿色线为蓝色相的组分，紫色线为红色相的组分。各相组分在各相内自由扩散，在相界面上受耗散模型控制，在两相间耗散。

## 固体力学



（详见GUIDE/固体力学模块ppt，案例见INPUT\_FILES/4\_solvers）

Modules.mechanics = true

打开固体力学开关，启动固体力学模块。

Modules.MechanicalField.type = 0

求解方法：0 – 差分法 1 – 谱方法

Modules.MechanicalField.max\_iteration\_steps = 20000

最大迭代步，用于限制求解器求解步数，防止难以收敛时的无限迭代。

Modules.MechanicalField.strain\_accuracy = 1e-5

应变的收敛精度，求解器求解结束的判据。

Modules.MechanicalField.debug = true

是否对求解器进行测试，选true时会在模块预计算阶段进行测试。

Modules.MechanicalField.mass\_density = 1

差分法求解器中的必需参数：质量密度。

Modules.MechanicalField.mechanical\_dt = 1e-2

差分法求解器中的必需参数：位移场收敛时间步。

Modules.MechanicalField.write\_U = true

光谱法求解器中的必需参数：是否输出位移场

Modules.MechanicalField.fix\_boundary.type = (0,0,0)

固体力学边界条件，固定边界1或周期边界0

Modules.MechanicalField.Stiffness.相名称 = [(C11, C12, C13, C14, C15, C16), (C21, C22, C23, C24, C25, C26), (C31, C32, C33, C34, C35, C36), (C41, C42, C43, C44, C45, C46), (C51, C52, C53, C54, C55, C56), (C61, C62, C63, C64, C65, C66)]

Modules.MechanicalField.EigenStrain.相名称 = (ε11, ε22, ε33, ε23, ε13, ε12)

定义每个相的弹性常数和本征应变。

Modules.MechanicalField.plasticity = true

塑性求解开关。

Modules.MechanicalField.Plasticity.write\_info = true

是否输出统计的力学信息。

Modules.MechanicalField.Plasticity.max\_iteration\_steps = 1

塑性最大迭代步，防止塑性求解难以收敛。

Modules.MechanicalField.Plasticity.yield\_stress = (相的屈服应力, …)

塑性求解器中的必需参数，用于判断模拟域何处需要屈服。

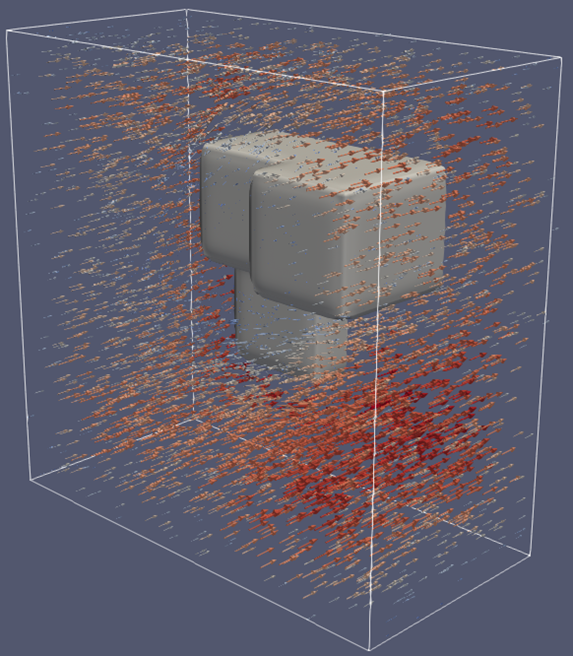
Modules.MechanicalField.Plasticity.hardening\_modulus = (相的强化模量, …)

塑性求解器中的必需参数，用于材料屈服过程中的强化作用。

Modules.MechanicalField.Plasticity.shear\_modulus = (相的剪切模量, …)

塑性求解器中的必需参数，用于塑性求解器的迭代收敛。

## 流体力学



（详见GUIDE/流体力学模块ppt，案例见INPUT\_FILES/4\_solvers）

Modules.fluid\_field = true

打开流体力学开关，启动流体力学模块。

Modules.FluidField.solver = 0

求解器类型：0 – 传统差分法, 1 – 格子玻尔兹曼法

Modules.FluidField.viscosity = 流体黏度

Modules.FluidField.mass = 传统差分法流体密度

Modules.FluidField.max\_solver\_iterate\_steps = 传统差分法最大迭代步数

Modules.FluidField.fluid\_dt = 传统差分法求解时采用的时间步长

Modules.FluidField.Pressure.max\_correction\_steps = 传统差分法压强求解最大迭代步

Modules.FluidField.Pressure.accuracy = 传统差分法压强求解精度

Modules.FluidField.Velocity.accuracy = 传统差分法速度求解精度

Modules.FluidField.density = LBM法流体密度

Modules.FluidField.max\_iterate\_steps = LBM法最大迭代步数

Modules.FluidField.momentum\_accuracy = LBM法动量求解精度

Modules.FluidField.debug\_solver = true

Modules.FluidField.debug\_output\_step = 100

是否在模块预求解阶段进行调试，及调试时每多少步进行一次输出。

Modules.FluidField.solid\_phases = (相名称, …)

通过设定固相，排除不需要求解的区域。

Modules.FluidField.BC\_Down\_X.fix\_velocity = (u, v, w)

传统差分法，当模拟域边界条件非周期时，会自动设定特定方向的液体流速。

Modules.FluidField.boundary\_condition = (1,1,0,0,0,0)

LBM方法，可以设定各个方向 (x\_down, x\_up, y\_down, y\_up, z\_dowm, z\_up)上的边界条件 ：

0 – 墙, 1 – 周期, 2 – 自由, 3 – 压力, 4 – 法向流

Modules.FluidField.BC\_Down\_X.wall\_roughness = LBM法，墙面粗糙度（0-1）

Modules.FluidField.BC\_Down\_X.wall\_speed = LBM法，墙面移动速度

Modules.FluidField.BC\_Down\_X.normal\_velocity = LBM法，法向液体速度

Modules.FluidField.BC\_Down\_X.pressure = LBM法，压力大小

Modules.FluidField.LBE.force\_model = 2

LBM求解器中耦合外力的模型：0 – 无模型, 1 – LGA模型, 2 – GZS模型

Modules.FluidField.force = 1

LBM求解器中耦合的外力： 0 – 无外力, 1 – 热膨胀模型

Modules.FluidField.force.gravity\_undersurface = 2

LBM求解器中重力加速度的方向：

0 - x\_down, 1 - x\_up, 2 - y\_down, 3 - y\_up, 4 - z\_dowm, 5 - z\_up

Modules.FluidField.force.reference\_temperature = 0.000000

LBM求解器中热膨胀模型的参考温度

Modules.FluidField.force.thermal\_expansion\_parameter = 0.002

LBM求解器中热膨胀模型的膨胀系数