GDEM 力学分析系列软件

技术参考手册

北京极道成然科技有限公司
二〇一三年四月

目录

GDEM 的特点

数值计算方法主要包括两类：一类以连续介质力学为基础，模拟材料的连续变形及塑性破坏，主要包括有限元及有限差分等两种方法。另一类以非连续介质力学为基础，用于模拟散体系统的运动、碰撞特性，主要包括块体离散元及颗粒离散元等两种。

有限元法及有限差分法能够较好地模拟材料在连续状态下的特性，但不能模拟材料从连续到非连续的过程及在非连续状态下的运动特性。块体离散元及颗粒离散元在模拟非连续体的运动特性方面具有一定的优势，但较难模拟材料的连续变形过程。

基于连续介质力学的离散元方法（**Continuum-based Discrete Element Method**）是中国科学院力学研究所提出的适用于模拟材料在静、动载荷作用下非连续变形及渐进破坏的一种数值算法。该方法将有限元与离散元进行耦合，在块体内部进行有限元计算，在块体边界进行离散元计算，不仅可以模拟材料在连续状态下及非连续状态下的变形、运动特性，更可以实现材料由连续体到非连续体的渐进破坏过程。CDEM 方法中包括弹性模型、塑性模型、断裂模型、蠕变模型等多种模型，已经在岩土工程、采矿工程、结构工程及水利水电工程等多个领域广泛应用。

GPU 是图形处理器的简称，是计算机显卡的核心部件，是天然的高性能并行处理器。GPU 往往用于大型三维场景游戏的实时显示，

介于 GPU 的高效率并行机制，目前科研界已经开始将 GPU 技术应用于工程计算领域。

北京极道成然科技有限公司以中科院力学所的 CDEM 为基础，开发出了基于 GPU 技术的商用软件 GDEM，大大提升了计算速度及计算容量。以 GPU 中档配置（NVIDIA GeForce 285）为例，可计算百万量级的四面体单元，且每一万个单元计算一万个时间步的耗时仅为 14-17 秒。

GDEM 的应用领域

1、 岩土工程

（1）降雨、地震作用下，公路边坡、铁路边坡、基坑边坡、自然边坡的稳定性分析、支挡结构的优化设计及成灾规模的模拟；

（2）公路、铁路、地铁隧道的优化设计，支护方案的优选，顶板冒落、两帮垮塌、底板鼓胀、突水突泥等突发灾害的力学分析及超前预测分析；

（3）爆破开采隧道或爆破削减边坡坡脚过程中，岩土体的破坏过程及爆破效果模拟；

（4）复合地基（CFG 桩地基等）的承载力分析，灌注桩、预制桩与岩土体相互作用的模拟分析。

2、 采矿工程

（1）地下开采中，竖井稳定性的模拟，巷道稳定性分析及支挡结构的优化设计；

（2）煤与瓦斯突出过程模拟并对煤层开采过程中的是否会发生突出事故进行力学分析及预测；

（3）露天开采中，露天矿边坡稳定性分析，排土场边坡稳定性分析及爆破开采的爆破方案优化设计；

（4）露天开采与地下开采相互影响的力学分析；

（5）尾矿库的整体稳定性分析及发生失稳溃坝后的运动状态模拟；

（6）隧道掘进机（TBM）、割煤机或钻机的破岩能力分析及钻头受力、磨损分析。

3、 水利水电工程

（1）大、中、小型水库的整体稳定性分析，在暴雨及库水位上涨的情况下，水库坝体发生管涌、溃坝的概率分析及溃坝过程模拟；

（2）水库两岸高陡岩质边坡的稳定性分析及成灾规模的分析。

4、 结构工程

（1）高层建筑在地震作用下的破坏状态及成灾规模模拟；

（2）钢筋混凝土中应力波的传播及无损探伤的正演模拟；

（3）冷却塔、烟囱、高层建筑的拆除爆破。

5、 港口工程

（1）高桩码头、重力式码头等典型码头的整体稳定性分析；

（2）海洋浮冰对海洋平台整体稳定性影响的评价。

GDEM 的基本原理

GDEM 采用基于时程的动态松弛技术进行显示迭代计算，求解动态问题、非线性问题及大位移、大转动问题具有明显优势，其计算流程如图 1 所示。

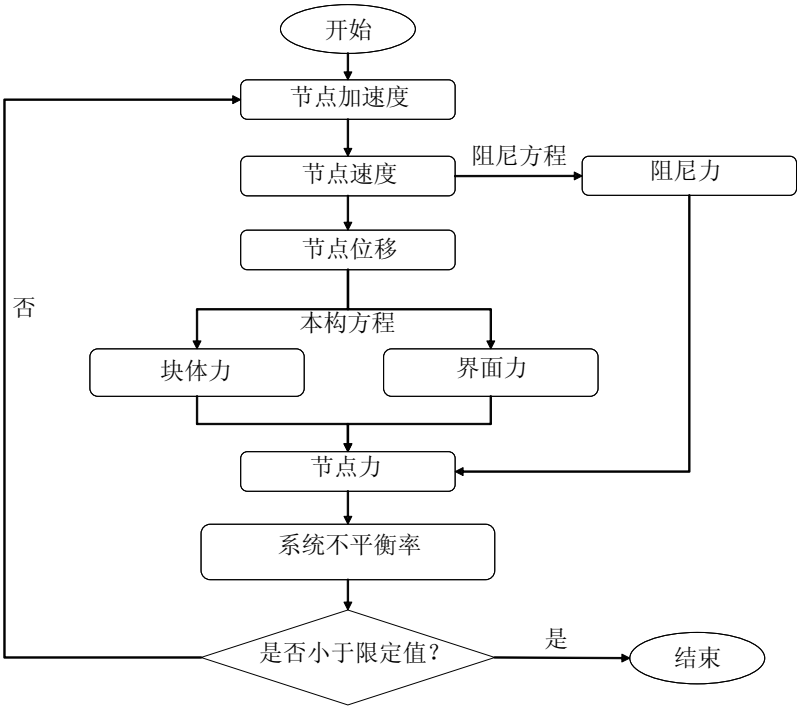
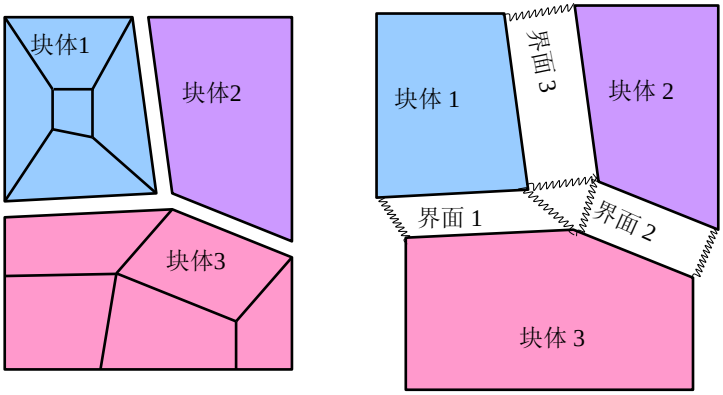


图 1 GDEM 方法的计算流程

GDEM 中的块体由一个或多个有限元单元组成，在块体内部使用连续本构，块体边界使用非连续本构（如图 2）。GDEM 中每个有限元单元可以是简单的四面体、五面体及六面体单元，也可以是

复杂的多面体单元（如图 3）。GDEM 中块体间的非连续变形主要通过弹簧来实现，通过弹簧的断裂来模型材料的开裂、滑移等。



块体1------5 个有限元单元　　接触 1------4 个法向、4个切向弹簧

块体2------1 个有限元单元　　接触 2------4 个法向、4个切向弹簧

块体3------5 个有限元单元　　接触 3------4 个法向、4个切向弹簧

图 2 GDEM 中的块体及界面

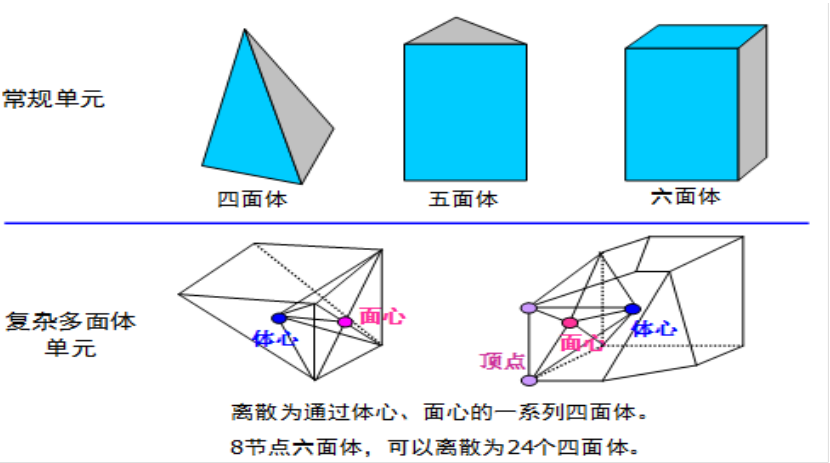


图 3 GDEM 中的有限元单元

GDEM 采用刚度矩阵法求解单元的内力。由于 GDEM 采用动态松弛技术，并不需要形成总体刚度矩阵，而只需求取每一个单元的单元刚度矩阵，在每个迭代步通过下式计算单元自身的节点力，并将此节点力分配至单元对应的节点上。

$$\{F\}_i^e = [K]_i^e \{u\}_i^e \tag{1}$$

其中：$\{F\}_i^e$ 为单元 i 的节点力向量，$\{u\}_i^e$ 为单元 i 的节点位移向量，$[K]_i^e$ 为单元 i 的单元刚度矩阵。

GDEM 界面的法向及切向弹簧的示意图如图 4 所示，弹簧力的计算如式(2)所示。

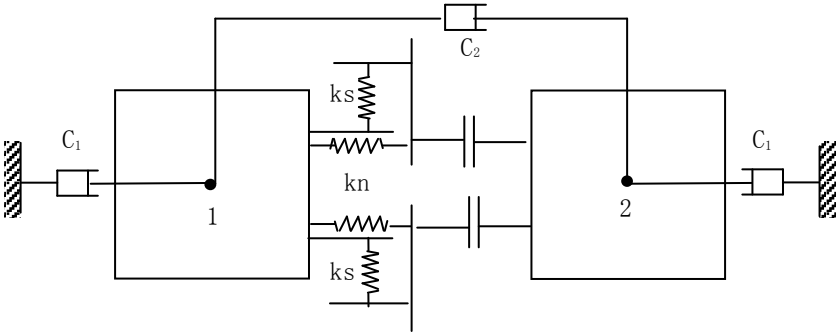


图 4 接触面法向、切向弹簧示意

$$\begin{cases} F_n^{\ j} = -K_n^{\ j} \times \Delta d_n^{\ j} \\ F_s^{\ j} = -K_s^{\ j} \times \Delta d_s^{\ j} \end{cases} \tag{2}$$

其中，$F_n^{\ j}$、$F_s^{\ j}$ 表示第 j 根弹簧的法向及切向力，$K_n^{\ j}$、$K_s^{\ j}$ 表示第 j 根弹簧的法向及切向刚度，$\Delta d_n^{\ j}$ 及 $\Delta d_s^{\ j}$ 表示第 j 根弹簧的法向及切向位移。

进行破坏计算时，采用 Mohr-Coulomb 准则，对式中的弹簧力进行修正，如下式

$$\begin{cases} (1)\ \ If\quad -F_n^{\,j} \geq T \quad F_n^{\,j} = F_s^{\,j} = 0 \\ (2)\ \ If\quad F_s^{\,j} \geq F_n^{\,j} \times \tan(\phi) + C \\ F_s^{\,j} = F_n^{\,j} \times \tan(\phi) + C, C = 0 \end{cases} \tag{3}$$

其中 T 为抗拉强度， ϕ 为内摩擦角， C 为粘聚力。

GDEM 的基本模型

GDEM 中的数值模型分为块体模型、界面模型及工程结构模型等三类。

1、 块体模型

块体模型主要包括：线弹性模型、Drucker-Prager 模型、Mohr-Coulomb 模型、块体切割模型、孔隙渗流模型、蠕变模型等 6 种。

（1）线弹性模型

用于计算系统的弹性应力场、应变场及位移场。

（2）Drucker-Prager 模型

用于模拟材料的塑性破坏，包括拉伸破坏及剪切破坏两种形式。

（3）Mohr-Coulomb 模型

用于模拟材料的塑性破坏，包括拉伸破坏及剪切破坏两种形式。

（4）块体切割模型

用于模拟连续体中显式裂纹的出现过程，脆性材料的裂纹扩展模拟以线弹性模型为基础；延性材料的裂纹扩展模拟以 Drucker-Prager 模型或者 Mohr-Coulomb 模型为基础，当模型出现局部化带后，进行块体切割，形成显式裂纹。

（5）孔隙渗流模型

用于模拟孔隙介质中流体的运移过程，可以实现稳态流、非稳态流的计算，可以给出孔隙水压力的分布特征，浸润线的位置等，可以与固体力学模型进行耦合求解。

（6）蠕变模型

采用改进的 Burger 与 Mohr-Coulomb 耦合模型，可以模拟材料的粘弹塑性计算，适合模拟高地应力下材料的渐进破坏。

2、 界面模型

界面模型主要包括：线弹性模型、脆性断裂模型、应变软化断裂模型、裂隙渗流模型等 4 种。

（1）线弹性模型

用于相邻块体间节点力的传递，相当于节点间的强约束条件。

（2）脆性断裂模型

用于模拟连续材料（脆性）的开裂、破坏，用于模拟干节理的破坏。

（3）应变软化断裂模型

用于模拟连续材料（延性）的开裂、破坏，用于模拟含软弱夹层节理的破坏。

（4）裂隙渗流模型

用于模拟节理层中的渗流过程，可以与固体力学模型进行耦合求解。

3、 工程结构模型

工程结构模型主要包括：锚杆/锚索模型、钢筋模型、桩梁模型等四种。

（1）锚杆/锚索模型

用于模拟工程中的锚杆/锚索等，可以施加预应力，可以计算其从围岩中拔出、自身断裂等物理过程。

（2）钢筋模型

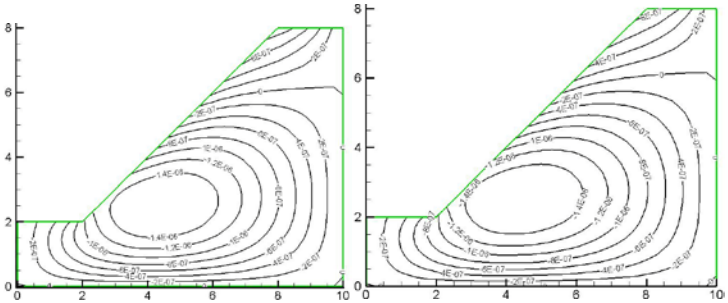
用于模拟钢筋混凝土结构中的钢筋。

（3）桩梁模型

用于模拟弯曲作用不可忽略的工程结构，如治理滑坡时采用的抗滑桩、高桩码头中采用的斜桩、建筑结构中的承重梁等。

GDEM 典型算例验证

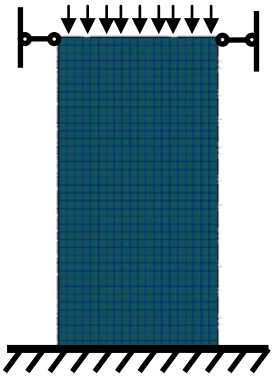
1、 弹性场验证

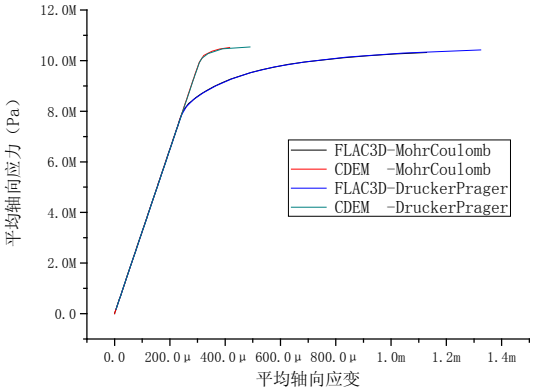


（a） GDEM 的解　　（b） FLAC3D 的解

图 5 弹性场验证

2、 塑性场验证

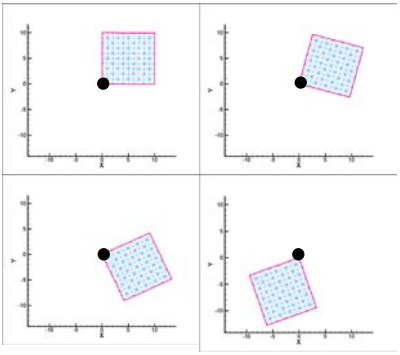




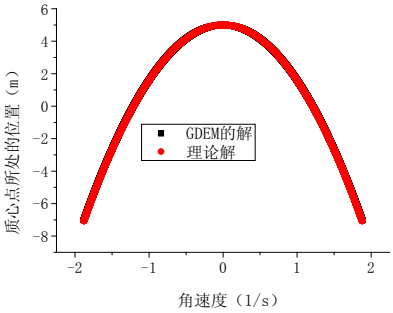
（a）单轴压缩模型　　（b） GDEM 及 FLAC3D 的结果对比

图 6 塑性场验证

3、 单元大转动验证



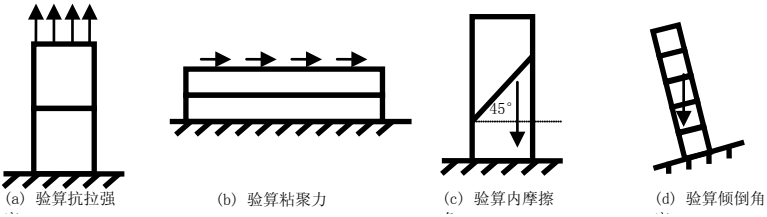
（a）GDEM 计算的转动图像



（b） GDEM 解与理论解的对比

图 7 单元大转动验证

4、 非连续破坏准则验证



(a) 验算抗拉强　(b) 验算粘聚力　(c) 验算内摩擦　(d) 验算倾倒角

图 8 破坏准则的验证算例

表 1 破坏算例统计

验证类别	抗拉强度	粘聚力	内摩擦角	倾倒角度

载荷形式	均布竖直拉应力	均布水平剪应力	自重	自重
强度指标	抗拉强度 10kPa	粘聚力 30kPa	粘聚力 0 抗拉强度 0	内摩擦角 26° 粘聚力 0 抗拉强度 0
出现破坏时理论值	拉应力 =10kPa	剪应力 =30kPa	内摩擦角 =45°	倾斜角度 =14.036°
出现破坏时计算值	拉应力 =10.001 kPa	剪应力 =29.6 kPa	内摩擦角 =49.999°	倾斜角度 =14.04°

5、 运动性破坏验证

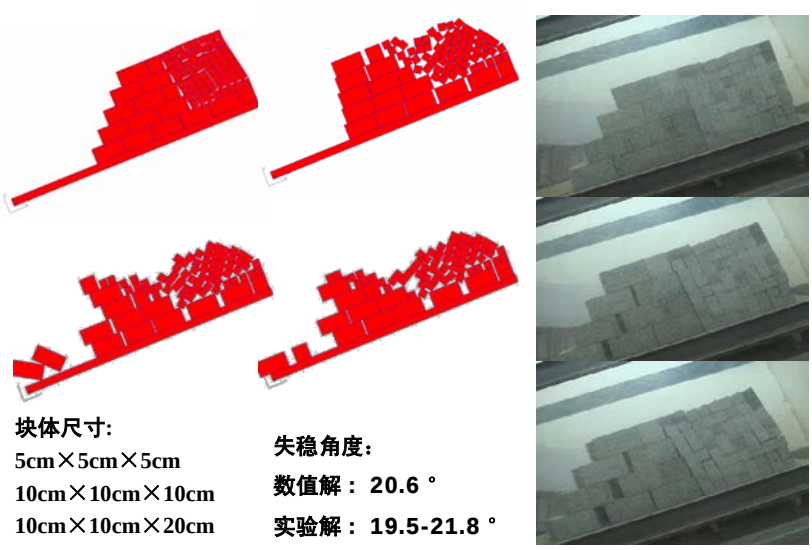


图 9 混合块体倾倒滑移的 GDEM 解与实验解的对比

6、 孔隙渗流验证

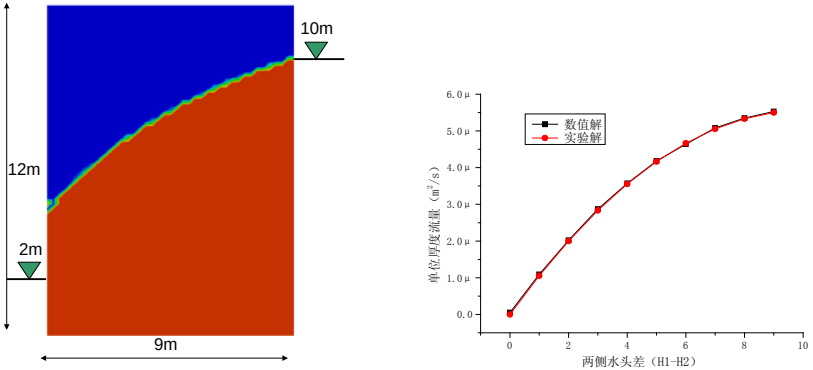


图 10 稳态孔隙渗流时 GDEM 与实验的结果对比

7、 裂隙流验证

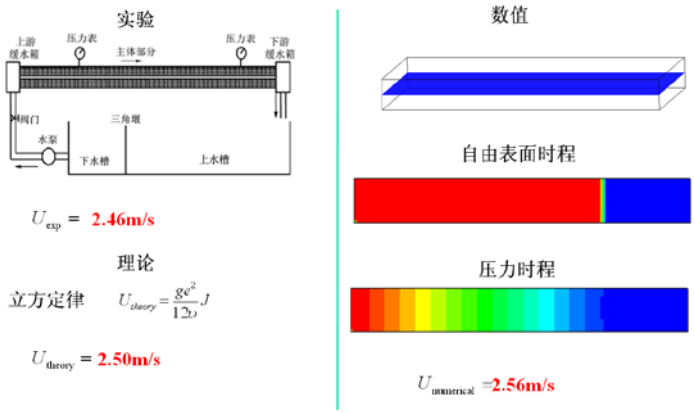
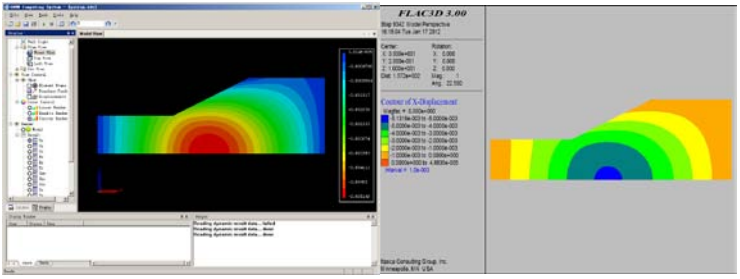


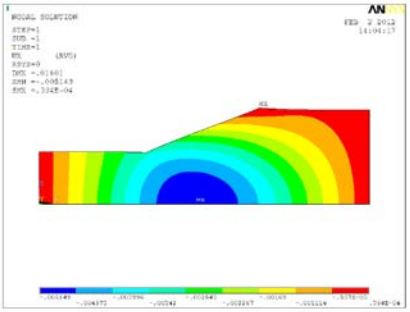
图 11 GDEM、实验、理论的结果对比

8、 标准坡体的塑性验证

注：ANSYS 采用的外接圆模型，并且没有内置内切圆模型，因此比较时通过公式转化为内切圆模型进行比较。此外，ANSYS 显示的剪应变为 Mise 应变。

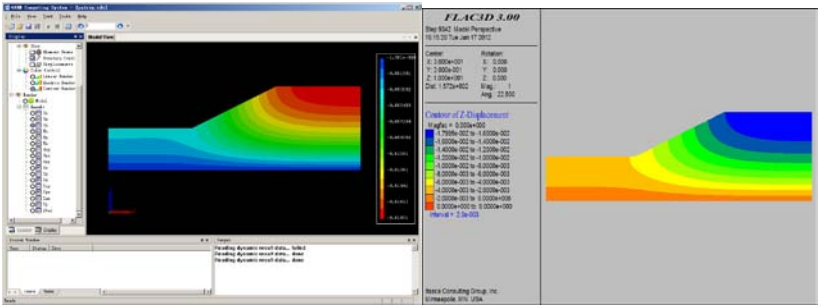


（a）弹性水平位移 GDEM 结果　　（b）弹性水平位移 FLAC3D 结果

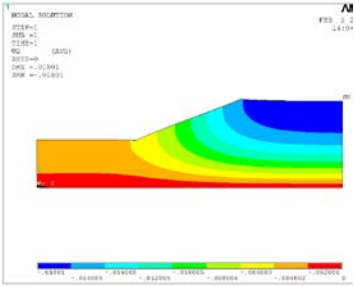


（c）弹性水平位移 ANSYS 结果

图 12 弹性水平位移 GDEM、FLAC3D、ANSYS 的结果对比

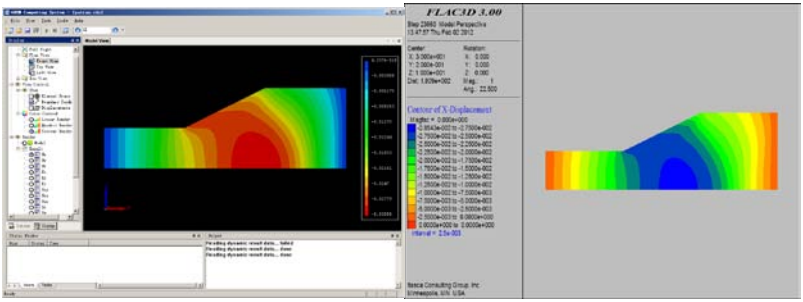


（a）弹性竖直位移 GDEM 结果　　（b）弹性竖直位移 FLAC3D 结果

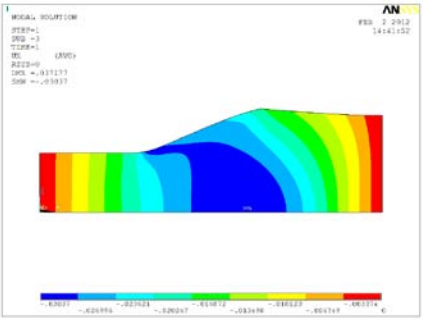


（c）弹性竖直位移 ANSYS 结果

图 13 弹性竖直位移 GDEM、FLAC3D、ANSYS 的结果对比

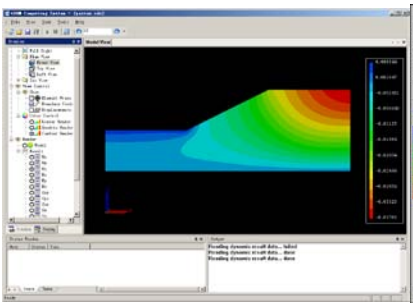


（a）塑性水平位移 GDEM 结果　　（b）塑性水平位移 FLAC3D 结果

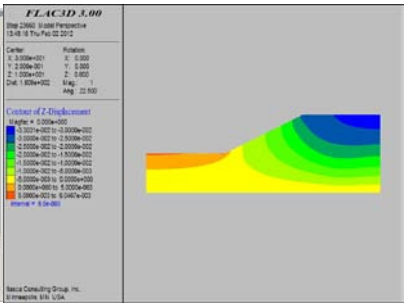


（c）塑性水平位移 ANSYS 结果

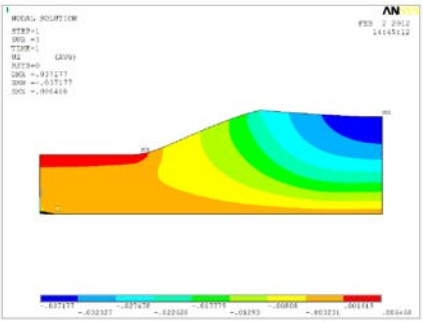
图 14 塑性水平位移 GDEM、FLAC3D、ANSYS 的结果对比



（a）塑性竖直位移 GDEM 结果

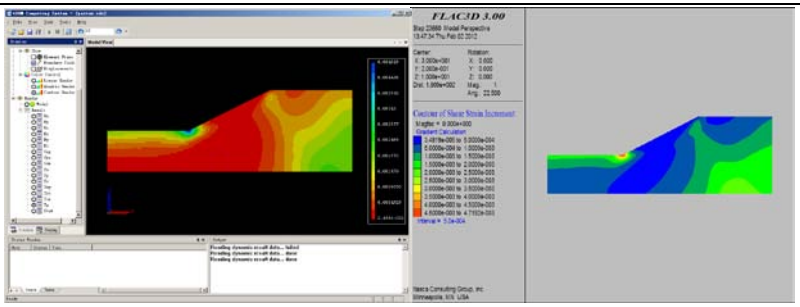


（b）塑性竖直位移 FLAC3D 结果

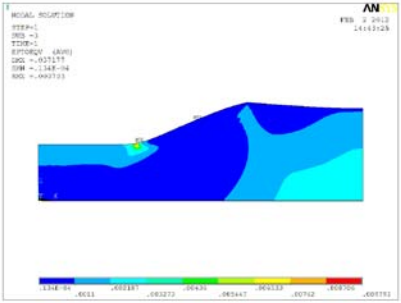


（c）竖直位移 ANSYS 结果

图 15 塑性竖直位移 GDEM、FLAC3D、ANSYS 的结果对比



（a）塑性剪应变 GDEM 结果　　（b）塑性剪应变 FLAC3D 结果



（c）塑性剪应变 ANSYS 结果

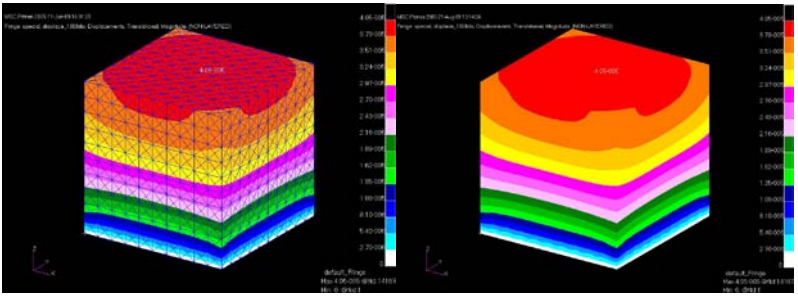
图 16 塑性剪应变 GDEM、FLAC3D、ANSYS 的结果对比

GDEM 中 GPU 的加速效果

1、 混凝土块弹性场计算

包含 1000 个六面体单元的混凝土块，在重力作用下的弹性位移场如图 17 所示，CPU 计算耗时 252 秒，GPU 计算耗时 0.478 秒。

GPU 配置：GeForce GTX285；CPU 配置：Core (TM) 2，1.86GHZ。



a. CDEM　　b. GDEM

图 17 混凝土弹性场计算

2、 长城烽火台弹性场计算

41232 个六面体单元，4000 迭代步，CDEM 耗时 95 分钟，GDEM 耗时 6.1 秒，商用软件 FLAC3D 耗时 13 分钟。（结果如图 18 所示）

GPU 配置：GeForce GTX285；

CPU 配置：Core (TM) 2，1.86GHZ。

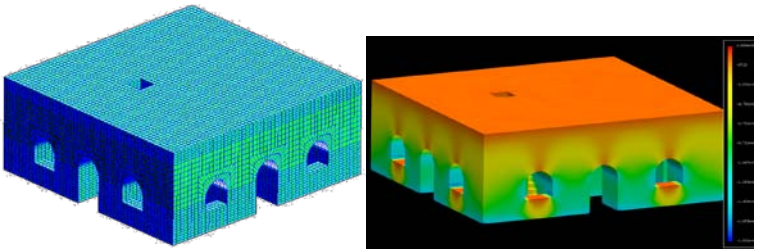


图 18 长城烽火台弹性场计算

3、 凉水井滑坡弹性场计算

85800 个六面体单元，5000 迭代步，GDEM 耗时 12.5 秒，商用软件 FLAC3D 耗时 2170 秒。（计算结果如图 19 所示）

GPU 配置：GeForce GTX485；

CPU 配置：Core (TM) 2，1.86GHZ。

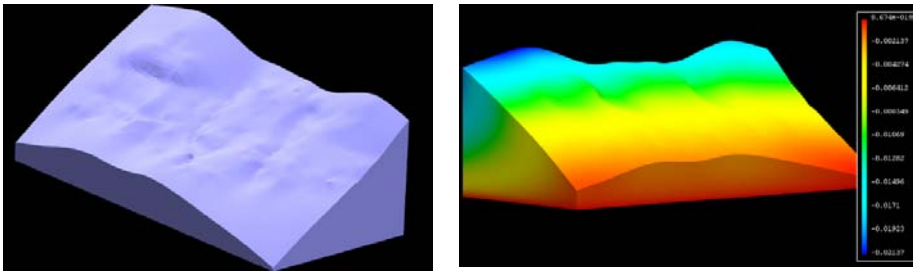


图 19 凉水井滑坡弹性场计算

4、 露天矿边坡弹性场计算

33.5 万个六面体单元，53 秒完成弹性场计算。

GPU 配置：GeForce GTX485；

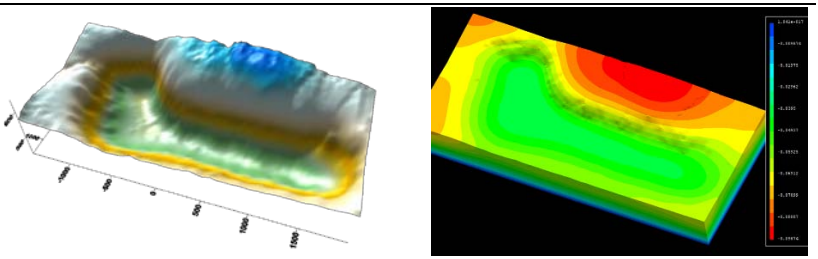


图 20 露天矿边坡弹性场计算

5、 三维地质体应力场计算

127 万个四面体单元，耗时不超过 8 分钟。

GPU 配置：GeForce GTX485；

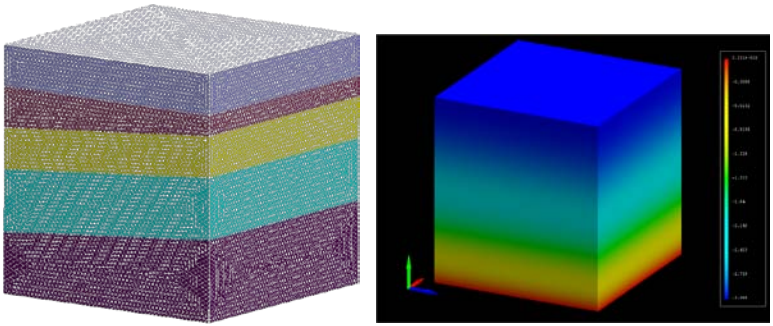
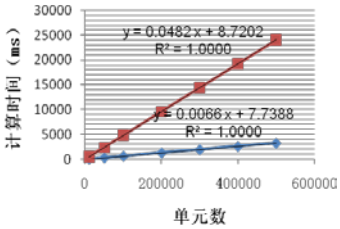


图 21 三维地质体弹性场计算

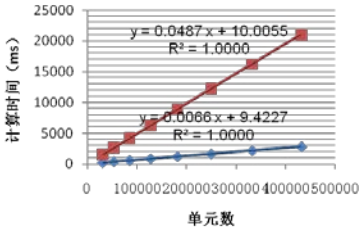
6、 计算效率统计

（1）连续问题

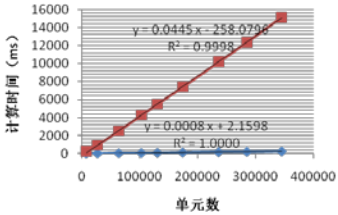
在求解连续问题时，GDEM 的计算效率如图 22 所示（GPU 配置：GeForce GTX485），具体统计数据如表 2 所示。



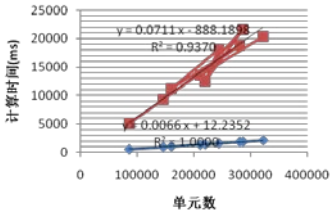
a. 六面体



b. 五面体



c. 四面体



d. 混合单元

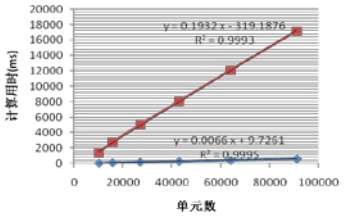
图 22 单元类型与计算时间的关系

表 2 各种单元类型的计算效率对比

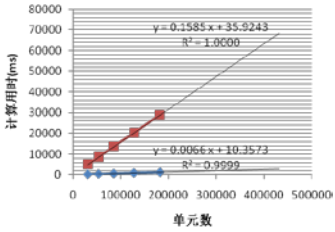
连续问题 1000 个计算单元迭代 1000 次的计算时间				
单元类型	六面体	五面体	四面体	混合单元
产生刚阵时间(ms)	6.6	6.6	0.8	6.6
迭代时间(ms)	48.2	48.7	44.5	71.1

（2）离散问题

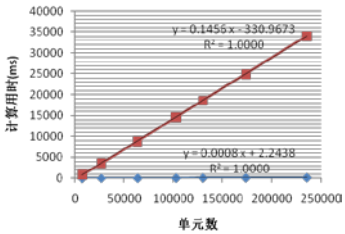
在求解非连续问题时，GDEM 的计算效率如图 7 所示（GPU 配置：GeForce GTX485），具体统计数据如表 3 所示。



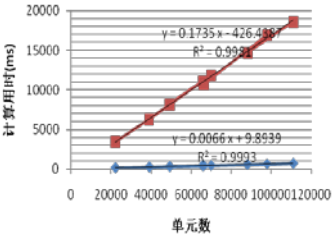
a. 六面体



b. 五面体



c. 四面体



d. 混合单元

图 23 单元类型与计算时间的关系

表 3 各种单元类型的计算效率对比

离散系统 1000 个计算单元迭代 1000 次的计算时间				
单元类型	六面体	五面体	四面体	混合单元
产生刚阵时间(ms)	6.6	6.6	0.8	6.6
迭代时间(ms)	193.2	158.5	145.6	173.5

（3）最终统计

总体而言，计算线弹性问题，每 1 万个单元迭代 1 万步，耗时为 4-7 秒，且迭代 5000 步左右，可以得到较为准确的弹性场。

表 4 各种问题的计算效率对比

1000 个计算单元迭代 1000 次的计算时间					
单元类型		六面体	五面体	四面体	混合单元
产生刚阵时间(ms)	连续问题	6.6	6.6	0.8	6.6
	离散系统	6.6	6.6	0.8	6.6
迭代时间(ms)	连续问题	48.2	48.7	44.5	71.1
	离散系统	193.2	158.5	145.6	173.5

附录 1　MSR 文件格式与读写说明

1、 MSR 文件简介

MSR(Multi-Step Results)是一种数据文件，专门用于存储 GDEM 计算结果数据。在 GDEM 计算结束后，软件会自动生成一个 MSR 文件，存储在用户当前使用的文件夹下，用户可以及时查看结果。用户若关闭 GDEM 重新打开时，则可以通过以下操作查看结果文件：在 GDEM2.0 版中，点击 Import -> CDEM Result 导入 MSR 计算结果。导入之后，可以通过点击 Display 查看相应的结果。MSR 文件默认的扩展名是.msr。

目前数据存储的形式分为两种：一种是文本文件，即 ASCII 码(或 UNICODE 码)文件，一种是二进制文件。例如，数字 5678 在 ASCII 码文件中的存储形式是 00110101 00110110 00110111 0011100，占用 4 个字节；在二进制文件中存储形式是 00010110 00101110，占用 2 个字节。MSR 文件中的数据，都是以二进制形式存储的。之所以选择二进制文件存储 GDEM 数据，是因为二进制文件具有诸多优点。使用二进制文件存储，占用磁盘空间小，读取与写入速度快，存储利用率高，尤其在读写 GDEM 这种大数据量结果文件时，优势十分明显。但是使用二进制文件必须要理解文件中数据的意思，这就需要人为制定读写文件的规则——即文件格式，方便用户进行统一读写。在介绍文件格式之前，先看看定义的关键量和调用的函数。

2、 关键量和调用函数

宏定义：

```
#define DYN_RESULT_VARS        16    //结果变量个数
#define MAX_DYN_VARS           32    //最多变量个数
#define  MAX_NODES_PER_ELEM    8     //单元最大节点个数
```

变量定义：

```
char
*(m_astrDynResultVarNames[DYN_RESULT_VARS]);//MSR 写入变量的名字
unsigned __int64 m_nStepsDynWrote;      //MSR 数据写入步数
unsigned __int64 m_nSizeHeader;         //MSR 头文件大小

unsigned __int64 m_nSizeDynHeader;      //头文件大小
unsigned __int64 m_nDynElems;           //单元数
unsigned __int64 m_nDynResultSteps;     //步数
unsigned int m_nDynResultSizeReal;      //real 类型
unsigned int m_nDynMaxNodes;            //单元最大节点数
unsigned int m_nDynNodeVars;            //节点所有变量数
unsigned int m_nSizeDynElemVars;        //单元所有变量数
```

char m_acDynResultDataNum[MAX_DYN_VARS]; //单元数据点个数

char m_acDynResultType[MAX_DYN_VARS]; //MSR 结果数据类型

std::string m_astrDynResultNames[MAX_DYN_VARS]; //MSR 结果变量名

调用函数：

void BeginWriteDyn
(FILE *pFileDyn);

void WriteDynData (unsigned __int64 iter, real time_now, FILE *pFileDyn);

void EndWriteDyn
(FILE *pFileDyn);

int ReadDynResult (unsigned __int64 iCurrentStep, FILE *pDynResultFile);

上面定义的量，都是全局性质的，放在类的数据成员里或放在全局位置；其他用到的变量都是在函数内部定义的，为局部变量。

3、 MSR 文件格式

MSR 文件包括头文件和数据文件两部分。

(1) 文件头

文件头中包括以下写入量：

m_nSizeHeader——头文件的大小

m_nStepsDynWrote——写入的步数

"CdemRslt　　1.02"——版本号

nSizeReal——单/双精度大小(单精度为 4，双精度为 8)

nElems——单元数

MAX_NODES_PER_ELEM——单元最大节点个数

n1 = DYN_RESULT_VARS——MSR 结果文件变量数

n2 = MAX_NODES_PER_ELEM * (DYN_RESULT_VARS * nSizeReal)——每个单元存储所有结果变量所需要的大小

c1 = '\x00'——0 为与节点数相同，1 为每单元一个(四面体应力应变)

strlen(m_astrDynResultVarNames[i1])——写入变量名的大小，第一个字符是标志，代表本字段的数据类型(i1 = 0 : n1)

m_astrDynResultVarNames[i1]——写入变量名(i1 = 0 : n1)

(2)文件主体

文件主体包括以下写入量：

iStage——计算阶段号

iter——当前时步

time_now——当前时步对应的物理时间

nReserved——保留字，当前版本为 0，用于后续扩展

Data——用户要写入的数据，注意写入数据要按照单元、节点两重循环，每次写入一个单元一个节点的所有变量数据

文件类型	写入量名称	写入量类型	写入量含义
文件头	m_nSizeHeader	unsigned __int64	头文件的大小
	m_nStepsDynWrote	unsigned __int64	写入的步数
	"CdemRslt 1.02"	std::string	版本号
	nSizeReal	unsigned int	单/双精度大小(单精度为4，双精度为8)
	nElems	unsigned __int64	单元数
	MAX_NODES_PER_ELEM	int	单元最大节点个数
	n1 = DYN_RESULT_VARS	int	MSR结果文件变量数
	n2 = MAX_NODES_PER_ELEM * (DYN_RESULT_VARS * nSizeReal)	int	每个单元存储所有结果变量所需要的大小
	c1 = '\x00'	char	0为与节点数相同，1为每单元一个(四面体应力应变)
	strlen(m_astrDynResultVarNames[i1])	short	写入变量名的大小，第一个字符是标志，代表本字段的数据类型(i1 = 0 : n1-1)
	m_astrDynResultVarNames[i1]	std::string	写入变量名(i1 = 0 : n1-1)
文	iStage	int	计算阶段号

件主体	iter	unsigned __int64	当前时步
	time_now	real	当前时步对应的物理时间
	nReserved	int	保留字，当前版本为0，用于后续扩展
	Data	real (float/double)	用户要写入的数据，注意写入数据要按照单元、节点两重循环，每次写入一个单元一个节点的所有变量数据 real的类型是float还是double取决于nSizeReal的大小，即nSizeReal = 4时为float，nSizeReal = 8时为double

4、 读 MSR 文件

按照 MSR 文件格式进行读操作。

调用的函数是：

int ReadDynResult (unsigned __int64 iCurrentStep, FILE *pDynResultFile);

(1) 读文件头大小

nRead = fread(&m_nSizeDynHeader , sizeof(m_nSizeDynHeader), 1, pDynResultFile);

(2) 读写入数据的步数

nRead = fread(&m_nDynResultSteps, sizeof(m_nDynResultSteps), 1, pDynResultFile);

如果 nRead = 0，则说明读入为空，程序报错，返回值 2。

(3) 读MSR文件版本号

nRead = fread(strFileType, 8, 1, pDynResultFile);

nRead = fread(strVersion , 8, 1, pDynResultFile);

如果 nRead = 0 或者 strcmp(strFileType, "CdemRslt") != 0 说明读入为空，或者不是 MSR 文件，程序报错，返回值 3。如果 strcmp(strVersion, " 1.02") != 0 说明版本号不对，程序报错，返回值 2。如果 m_nDynResultSteps = 0，程序报错，返回值为 4。

(4) 读real类型大小

nRead = fread(&m_nDynResultSizeReal, sizeof(m_nDynResultSizeReal), 1, pDynResultFile);

如果 m_nDynResultSizeReal == 8 说明是 double 类型，反之则为 float 类型。

(5) 读单元数

nRead = fread(&m_nDynElems, sizeof(m_nDynElems), 1, pDynResultFile);

(6) 读单元最大节点数

nRead = fread(&m_nDynMaxNodes, sizeof(m_nDynMaxNodes), 1, pDynResultFile);

(7) 读结果变量个数

nRead = fread(&m_nDynNodeVars, sizeof(m_nDynNodeVars), 1, pDynResultFile);

(8) 读单元所有自由度数

nRead = fread(&m_nSizeDynElemVars, sizeof(m_nSizeDynElemVars), 1, pDynResultFile);

如果 m_nDynElems != m_apElems.size()，说明读到的单元数与实际单元数不等，程序报错，返回值 2。

(9) 读标识符c1

```
fread(&c1, sizeof(c1), 1, pDynResultFile);
```

(10)读结果变量个数

```
fread(&n16, sizeof(n16), 1, pDynResultFile);
```

(11)读结果变量名称

```
fread(&buf, n16, 1, pDynResultFile);
```

(12)读计算阶段号

```
fread(&iStage, sizeof(iStage), 1, pDynResultFile);
```

(13)读时步数

```
fread(&iStep, sizeof(iStep), 1, pDynResultFile);
```

(14)读时间步长

```
fread(&fTimeStep, sizeof(fTimeStep), 1, pDynResultFile);
```

(15)读保留字

```
fread(&nReserved, sizeof(nReserved), 1, pDynResultFile);
```

(16)读数据部分

根据用户需要进行读取，比如位移、应变、应力等。

5、 写 MSR 文件

按照 MSR 文件格式进行写操作。

调用的函数是：

```
void BeginWriteDyn    (          FILE *pFileDyn); //写文件头
void WriteDynData     (unsigned __int64 iter, real time_now, FILE
*pFileDyn);                                          //写文件体
void EndWriteDyn      (          FILE *pFileDyn); //结束写文件
```

写文件头：

(1) 写文件头大小

fwrite(&m_nSizeHeader, sizeof(m_nSizeHeader), 1,

pFileDyn);

先写入 0 占位，后再写入实际值。

(2) 写写入数据的步数

fwrite(&m_nStepsDynWrote, sizeof(m_nStepsDynWrote), 1,

pFileDyn);

先写入 0 占位，后再写入实际值。

(3) 写MSR版本版本号

fwrite("CdemRslt 1.02", 16, 1pFileDyn);

(4) 写real类型大小

fwrite(&nSizeReal, sizeof(nSizeReal), 1, pFileDyn);

(5) 写单元数

fwrite(&m_nElems, sizeof(nElems), 1, pFileDyn);

(6) 写单元最大节点数

n1 = MAX_NODES_PER_ELEM;

fwrite(&n1, sizeof(n1), 1, pFileDyn);

(7) 写结果变量个数

n1 = DYN_RESULT_VARS;

fwrite(&n1, sizeof(n1), 1, pFileDyn);

(8) 写单元所有自由度数

n2 = MAX_NODES_PER_ELEM * (DYN_RESULT_VARS * nSizeReal);

fwrite(&n2, sizeof(n2), 1, pFileDyn);

(9) 写标识符c1

c1 = '\x00'; // 每单元数据点, 暂定 0 为与节点数相同, 1 为每单元一个(四面体应力应变)

fwrite(&c1, sizeof(c1), 1, pFileDyn);

(10)写结果变量个数

n16 =(short)strlen(m_astrDynResultVarNames[i1]);

//m_astrDynResultVarNames 第一个字符是标志，代表本字段的数据类型

fwrite(&n16, sizeof(n16), 1, pFileDyn);

(11)写结果变量名称

fwrite(m_astrDynResultVarNames[i1], n16, 1, pFileDyn);

写文件体：

(12)写计算阶段号

fwrite(&iStage, sizeof(iStage), 1, pFileDyn);

(13)写时步数

fwrite(&iter, sizeof(iter), 1, pFileDyn);

(14)写时间步长

fwrite(&time_now, sizeof(time_now), 1, pFileDyn);

(15)写保留字

fwrite(&nReserved,sizeof(nReserved), 1, pFileDyn);

保留字 nReserved = 0。

(16)写数据部分

先写节点数，再按照每个节点写入所有变量的数值。

如写 x 方向位移：

fwrite((char *)&nNodes, sizeof(nNodes), 1, pFileDyn);

fwrite(m_aElems[iElem].Displace[0],

sizeof(m_aElems[iBlock].Displace[0]),

MAX_NODES_PER_ELEM, pFileDyn);

结束写文件：

fclose(pFileDyn);

```
pFileDyn = NULL;
```

6、 GDEM 中使用的变量数和变量名称

```
#define DYN_RESULT_VARS        16          //GDEM 中结果变量个数为 16
#define MAX_DYN_VARS           32          //GDEM 中最多可能使用的变量个数，不小于结果变量数
#define  MAX_NODES_PER_ELEM    8           //GDEM 中单元最大节点个数
```

GDEM 中 16 个结果变量名称如下：

```
m_astrDynResultVarNames[0]  = "fUx";       //x 向位移
m_astrDynResultVarNames[1]  = "fUy";       //y 向位移
m_astrDynResultVarNames[2]  = "fUz";       //z 向位移
m_astrDynResultVarNames[3]  = "fEx";       //x 正应变
m_astrDynResultVarNames[4]  = "fEy";       //y 正应变
m_astrDynResultVarNames[5]  = "fEz";       //z 正应变
m_astrDynResultVarNames[6]  = "fGxy";      //xy 剪应变
m_astrDynResultVarNames[7]  = "fGyz";      //yz 剪应变
m_astrDynResultVarNames[8]  = "fGzx";      //zx 剪应变
m_astrDynResultVarNames[9]  = "fSx";       //x 正应力
m_astrDynResultVarNames[10] = "fSy";       //y 正应力
m_astrDynResultVarNames[11] = "fSz";       //z 正应力
```

```
m_astrDynResultVarNames[12] = "fTxy";          //xy 剪应力
m_astrDynResultVarNames[13] = "fTyz";          //yz 剪应力
m_astrDynResultVarNames[14] = "fTzx";          //zx 剪应力
m_astrDynResultVarNames[15] = "fTp";           //塑性剪应
变
```

用户可以自行定义，比如只有一个压力变量

```
#define DYN_RESULT_VARS      1             //用户自定
义数据变量个数为 1
m_astrDynResultVarNames[0] = "fPressure";      //用户自定
义数据变量名称
```

附录 2　GDEM 网格模型文本格式说明

本文档说明：黑色字符是 GDEM 前处理文本的正文内容。黑色源文件部分：正体字是关键字，不能改动；括号中的斜体字是用户可以自定义的参数；[ ]中的内容是选择性输入字符，只能从已定义好的选项中选择性输入一个值；< >中的内容可以由用户定义；[ ]和< >本身不出现在源文件中。$之间的文字为源文件中的模块说明；//后的文字对本行内容进行注释；#后的文字对随后一组参数做概括性说明。

----------GDEM 前处理文本----------

$程序头文件说明$
GDEMK　　//软件版本名称
2.00　　//版本号

$初始设置$
[0 | 1]　　//定义求解问题的类型：0 为连续模型；1 为离散模型
<*Contact Spring's Stiffness*>　　//定义离散系统中结构层的刚度

$定义材料参数$
<*Number of Material Groups*>　　//材料的组数
#第一组材料的参数

关键字	组号	值	
MATNAME	<*Group No.*>	<*Material's Name*>	//材料名称
EX	<*Group No.*>	<*Value*>	//杨氏模量
NUXY	<*Group No.*>	<*Value*>	//泊松比
DENS	<*Group No.*>	<*Value*>	//密度

COH <*Group No.*> <*Value*> //粘聚力
FRICA <*Group No.*> <*Value*> //摩擦角
TENS <*Group No.*> <*Value*> //抗拉强度
DILAT <*Group No.*> <*Value*> //剪胀角
END //结束第一组材料参数定义

#第二组材料的参数（若之前的材料组数设为 2，则可以仿照第一组材料的定义方法添加）

…...

$生成网格节点信息$
节点数 单元数 每个单元的最大节点数
<*Number of Nodes*> < *Number of Elements*> < *Number of Nodes per Element*>
节点号 X 坐标 Y 坐标 Z 坐标
< *Node No.*> <*X Coord*> <*Y Coord*> <*Z Coord*>
……
#单元及其属性和对应节点。类型：0 六面体；1 四面体；2 五面体
单元号 类型 材料号 单元的节点号（按逆时针方向排列）
< *Element No.*> [*0* | *1* | *2*] <*Material No.*> <*List of Nodes attached to Element*>
……

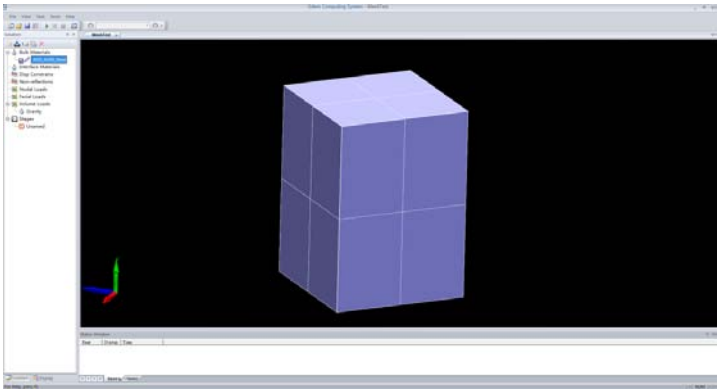
$定义计算阶段$
<*Number of Calculation Stages*> //阶段总数
#第一阶段基本参数设置
Seudo_Mass [*0* | *1*] //虚质量开关：0 关闭；1 打开
MinIt <*Number of Steps*> //最小迭代步数
MaxIt <*Number of Steps*> //最大迭代步数
dt <*TimeStep*> //计算时间步长

eps <*Value*> //收敛不平衡率

Gravity_X <*Value*> //重力加速度
Gravity_Y <*Value*> //Y 方向重力加速度
Gravity_Z <*Value*> //Z 方向重力加速度
CNSTI [*0 | 1 | 2 | 3 | 4*] //选择本构模型：
0 Linear;
1 D-P(inscribed);
2 D-P(circumscribed);
3 D-P(homalographic);
4 M-C
END //结束第一阶段基本参数设置
#第一阶段位移边界条件设置
<*Number of Nodes Constrained*> //被位移约束的节点总数
节点号 X 方向 Y 方向 Z 方向 X 值 Y 值 Z 值 保留
<Node No.> [*0 | 1*] [*0 | 1*] [*0 | 1*] 0 0 0 1
……
//说明：第 2、3、4 列中，1 表示约束，0 表示不约束；5、6、7 列的数值依次表示 X、Y、Z 方向被约束的位移值。
#第一阶段节点力设置
<Number of Nodes Applied Force> //被施加节点力的节点总数
节点号 X 方向 Y 方向 Z 方向
<Node No.> < *Fx* > < *Fy* > < *Fz* >
……
#第一阶段面力设置
<*Number of Areas Applied Face Force*>//被施加面力的单元面个数
单元号 面号 面力
<*Element No.*> <*Face No.*> <*Face Force* >
……
//第二阶段设置（重复第一阶段的格式）
……
----------结束----------

补充说明：除“程序头文件说明”和“生成网格信息”两部分外，其余内容均可在 GDEM 软件界面上添加、修改或删除。

一个简单网格文本实例



/*******************完整文本**********************/

GDEMK

2.00

0　　//连续模型

50　　//界面弹簧刚度为块体刚度 50 倍

1　　//共 1 组材料

MATNAME　1　AISI_4340_Steel　// 第　1　组 材 料 名 为 AISI_4340_Steel

EX　　1　2900000　　//第 1 组材料杨氏模量 2900000

NUXY　　1　0.32　　//第 1 组材料泊松比 0.32

DENS　　1　0.283　　//第 1 组材料密度 0.283

COH　　1　100000　　//第 1 组材料粘聚力 100000

FRICA　　1　30　　//第一组材料内摩擦角 30 度

TENS　　1　1.0E5　　//第一组材料抗拉强度 1.0E5

DILAT　　1　0　　//第一组材料剪胀角 0 度

END　　//结束第一组材料赋值

27　8　8　//模型共 27 个节点，8 个单元，每个单元 8 个节点

#模型 27 个节点各自的坐标

1　0　2　1

2　0.5　2　0.5

3　-0.5　2　0.5

4 0 1 1
5 0 2 0
6 0.5 1 0.5
7 -0.5 1 0.5
8 -1 2 0
9 0 1 0
10 1 2 0
11 -0.5 2 -0.5
12 0.5 2 -0.5
13 -1 1 0
14 1 1 0
15 0.5 1 -0.5
16 -0.5 1 -0.5
17 0 0 1
18 0 2 -1
19 0.5 0 0.5
20 -0.5 0 0.5
21 0 0 0
22 0 1 -1
23 1 0 0
24 -1 0 0
25 0.5 0 -0.5
26 -0.5 0 -0.5
27 0 0 -1

#模型 8 个单元各自的单元类型，材料号，及对应的节点号。六面体单元，材料号为 1。

1 0 1 9 6 19 21 15 14 23 25
2 0 1 16 9 21 26 22 15 25 27
3 0 1 5 2 6 9 12 10 14 15
4 0 1 11 5 9 16 18 12 15 22
5 0 1 7 4 17 20 9 6 19 21
6 0 1 13 7 20 24 16 9 21 26
7 0 1 3 1 4 7 5 2 6 9
8 0 1 8 3 7 13 11 5 9 16

1 //共 1 个计算阶段
Seudo_Mass 1 //虚质量开关开启
MinIt 0 //最小迭代步数为 0

```
MaxIt          30000     //最大迭代步数为 30000 步
Dt             0.6       //计算步长取 0.6
eps        0.0001    //设置不平衡率为 0.0001
Gravity_X      0.0       //X 方向重力加速度为 0
Gravity_Y      -9.8      //Y 方向重力加速度为-9.8
Gravity_Z      0.0       //Z 方向重力加速度为 0
CNSTI          0         //本构模型为线弹性
END                          //结束第一阶段的设置
#底面 9 个节点约束 XYZ 全部自由度
9                                          //共约束了 9 个节点
1    1    1    1    0.0  0.0  0.0
2    1    1    1    0.0  0.0  0.0
3    1    1    1    0.0  0.0  0.0
5    1    1    1    0.0  0.0  0.0
8    1    1    1    0.0  0.0  0.0
10   1    1    1    0.0  0.0  0.0
11   1    1    1    0.0  0.0  0.0
12   1    1    1    0.0  0.0  0.0
18   1    1    1    0.0  0.0  0.0
#顶面 9 个节点施加 X 方向的节点力各 1000N
9                          //共在 9 个节点上施加节点力
17   1000   0   0
19   1000   0   0
20   1000   0   0
21   1000   0   0
23   1000   0   0
24   1000   0   0
25   1000   0   0
26   1000   0   0
27   1000   0   0
#顶部 4 个单元面施加面力 1000000Pa
4                          //共在 4 个单元面上施加面力
3   3   1000000
4   3   1000000
7   3   1000000
8   3   1000000
/**********************结束**********************/
```

附录 3 欧拉角

莱昂哈德•欧拉用欧拉角来描述刚体在三维欧几里得空间的取向。对于任何参考系，一个刚体的取向，是依照顺序，从这参考系，做三个欧拉角的旋转而设定的。所以，刚体的取向可以用三个基本旋转矩阵来决定。换句话说，任何关于刚体旋转的旋转矩阵是由三个基本旋转矩阵复合而成的。

静态的定义

对于在三维空间里的一个参考系，任何坐标系的取向，都可以用三个欧拉角来表现。参考系又称为实验室参考系，是静止不动的。而坐标系则固定于刚体，随着刚体的旋转而旋转。

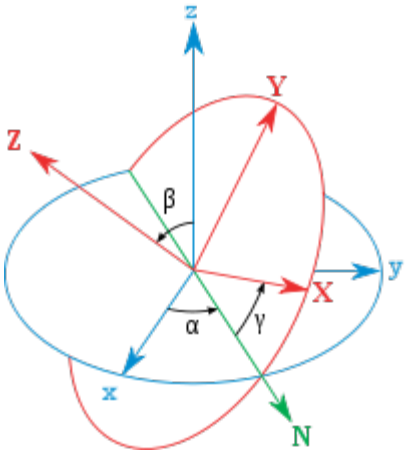
参阅下图。设定 xyz-轴为参考系的参考轴。称 xy-平面与 XY-平面的相交为交点线，用英文字母（N）代表。zxz 顺规的欧拉角可以静态地这样定义:

- α 是 x-轴与交点线的夹角，

- β 是 z-轴与 Z-轴的夹角，
- γ 是交点线与 X-轴的夹角。

很可惜地，对于夹角的顺序和标记，夹角的两个轴的指定，并没有任何常规。科学家对此从未达成共识。每当用到欧拉角时，我们必须明确的表示出夹角的顺序，指定其参考轴。

实际上，有许多方法可以设定两个坐标系的相对取向。欧拉角方法只是其中的一种。此外，不同的作者会用不同组合的欧拉角来描述，或用不同的名字表示同样的欧拉角。因此，使用欧拉角前，必须先做好明确的定义。



三个欧拉角：（α，β，γ）。蓝色的轴是 xyz-轴，红色的轴是 XYZ-坐标轴。绿色的线是交点线 (N)。

角值范围

- α，γ 值从 0 至 2π 弧度。
- β 值从 0 至 π 弧度。

对应于每一个取向，设定的一组欧拉角都是独特唯一的；除了某些例外：

- 两组欧拉角的 α ，一个是 0，一个是 2π ，而 β 与 γ 分别相等，则此两组欧拉角都描述同样的取向。
- 两组欧拉角的 γ ，一个是 0 ，一个是 2π，而 α 与 β 分别相等，则此两组欧拉角都描述同样的取向。

旋转矩阵

前面提到，设定刚体取向的旋转矩阵$[\mathbf{R}]$是由三个基本旋转矩阵合成的：

$$[\mathbf{R}]=\begin{bmatrix}\cos\gamma & \sin\gamma & 0\\ -\sin\gamma & \cos\gamma & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\beta & \sin\beta\\ 0 & -\sin\beta & \cos\beta\end{bmatrix}\begin{bmatrix}\cos\alpha & \sin\alpha & 0\\ -\sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{bmatrix}$$

单独分开作用，每个矩阵各自代表绕着其转动轴的旋转；但是，当它们照次序相乘，

- 最里面的(最右的) 矩阵代表绕着 z 轴的旋转。

- 最外面的(最左的) 矩阵代表绕着 Z 轴的旋转。
- 在中间的矩阵代表绕着交点线的旋转。

经过一番运算,

$$[\mathbf{R}]=\begin{bmatrix}\cos\alpha\cos\gamma-\cos\beta\sin\alpha\sin\gamma & \sin\alpha\cos\gamma+\cos\beta\cos\alpha\sin\gamma & \sin\beta\sin\gamma\\ -\cos\alpha\sin\gamma-\cos\beta\sin\alpha\cos\gamma & -\sin\alpha\sin\gamma+\cos\beta\cos\alpha\cos\gamma & \sin\beta\cos\gamma\\ \sin\beta\sin\alpha & -\sin\beta\cos\alpha & \cos\beta\end{bmatrix}$$

$[\mathbf{R}]$ 的逆矩阵是：

$$[\mathbf{R}]^{-1}=\begin{bmatrix}\cos\alpha\cos\gamma-\cos\beta\sin\alpha\sin\gamma & -\cos\alpha\sin\gamma-\cos\beta\sin\alpha\cos\gamma & \sin\beta\sin\alpha\\ \sin\alpha\cos\gamma+\cos\beta\cos\alpha\sin\gamma & -\sin\alpha\sin\gamma+\cos\beta\cos\alpha\cos\gamma & -\sin\beta\cos\alpha\\ \sin\beta\sin\gamma & \sin\beta\cos\gamma & \cos\beta\end{bmatrix}$$

别种顺序

在经典力学里，时常用 zxz 顺规来设定欧拉角；照着第二个转动轴的轴名，简称为 x 顺规。另外，还有别种欧拉角组。合法的欧拉角组中，唯一的限制是，任何两个连续的旋转，必须绕着不同的转动轴旋转。因此，一共有 12 种顺规。例如，y 顺规，第二个转动轴是 y-轴，时常用在量子力学，核子物理学，粒子物理学。另外，还有一种顺规，xyz 顺规，是用在航空航天工程学；

动态的定义

我们也可以给予欧拉角两种不同的动态定义。一种是绕着固定于刚体的坐标轴的三个旋转的复合；另外一种是绕着实验室参考轴的三个旋转的复合。用动态的定义，我们能更了解，欧拉角在物理上的含义与应用。特别注意，以下的描述，XYZ 坐标轴是旋转的刚体坐标轴；而 xyz 坐标轴是静止不动的实验室参考轴。

A) 绕着 XYZ 坐标轴旋转：最初，两个坐标系统 xyz 与 XYZ 的坐标轴都是重叠著的。开始先绕着 Z-轴旋转 α 角值。然后，绕着 X-轴旋转 β 角值。最后，绕着 Z-轴作角值 γ 的旋转。

B) 绕着 xyz 坐标轴旋转：最初，两个坐标系统 xyz 与 XYZ 的坐标轴都是重叠著的。开始先绕着 z-轴旋转 γ 角值。然后，绕着 x-轴旋转 β 角值。最后，绕着 z-轴作角值 α 的旋转。

参阅欧拉角图，定义 A 与静态定义的相等，这可以直接用几何制图方法来核对。

定义 A 与定义 B 的相等可以用旋转矩阵来证明：

思考任何一点P_1，在 xyz 与 XYZ 坐标系统的坐标分别为$\mathbf{r}_1$与$\mathbf{R}_1$。定义角算符$Z(\alpha)$为绕着 Z-轴旋转 α 角值。那么，定义 A 可以表述如下:

$$\mathbf{R}_1 = Z(\gamma) \circ X(\beta) \circ Z(\alpha) \circ \mathbf{r}_1$$

用旋转矩阵表示,

$$Z(\alpha) = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$X(\beta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{bmatrix}$$

$$Z(\gamma) = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

思考任何一点P_2，在 xyz 与 XYZ 坐标系统的坐标分别为$\mathbf{r}_2$与$\mathbf{R}_2$。定义角算符$z(\alpha)$为绕着 z-轴旋转 α 角值。则定义 B 可以表述如下:

$$\mathbf{r}_2 = z(\alpha) \circ x(\beta) \circ z(\gamma) \circ \mathbf{R}_2$$

用旋转矩阵表示,

$$z(\alpha)=\begin{bmatrix}\cos\alpha & -\sin\alpha & 0\\ \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$x(\beta)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\beta & -\sin\beta\\ 0 & \sin\beta & \cos\beta\end{bmatrix}$$

$$z(\gamma)=\begin{bmatrix}\cos\gamma & -\sin\gamma & 0\\ \sin\gamma & \cos\gamma & 0\\ 0 & 0 & 1\end{bmatrix}$$

假设,r1=r2,那么，

$$\mathbf{r}_1 = z(\alpha)\circ x(\beta)\circ z(\gamma)\circ \mathbf{R}_2$$

乘以逆算符，

$$z^{-1}(\gamma)\circ x^{-1}(\beta)\circ z^{-1}(\alpha)\circ \mathbf{r}_1 = z^{-1}(\gamma)\circ x^{-1}(\beta)\circ z^{-1}(\alpha)\circ z(\alpha)\circ x(\beta)\circ z(\gamma)\circ \mathbf{R}_2$$

但是，从旋转矩阵可以观察出，

$$z^{-1}(\alpha) = Z(\alpha)$$

$$x^{-1}(\beta) = X(\beta)$$

$$z^{-1}(\gamma) = Z(\gamma)$$

所以，

$$Z(\gamma)\circ X(\beta)\circ Z(\alpha)\circ \mathbf{r}_1 = \mathbf{R}_2$$

$$\mathbf{R}_1 = \mathbf{R}_2$$

定义 A 与定义 B 是相等的。

欧拉角性质

欧拉角在 SO(3)上，形成了一个坐标卡 (chart) ；SO(3)是在三维空间里的旋转的特殊正交群。这坐标卡是平滑的，除了一个极坐标式的奇点在 $\beta = 0$。

类似的三个角的分解也可以应用到 SU(2)；复数二维空间里旋转的特殊酉群；这里，β 值在 0 与2π 之间。这些角也称为欧拉角。

应用

欧拉角广泛地被应用于经典力学中的刚体研究，与量子力学中的角动量研究。

在刚体的问题上，xyz 坐标系是全局坐标系， XYZ 坐标系是局部坐标系。全局坐标系是不动的；而局部坐标系牢嵌于刚体内。关于动能的演算，通常用局部坐标系比较简易；因为，惯性张量不随时间而改变。如果将惯性张量（有九个分量，其中六个是独立的）对角线化，那么，会得到一组主轴，以及一个转动惯量（只有三个分量）。

在量子力学里， 详尽的描述 SO(3)的形式，对于精准的演算，是非常重要的， 并且几乎所有研究都采用欧拉角为工具。在早期的量子力学研究，对于抽象群理论方法(称为 Gruppenpest)， 物理学家与化学家仍旧持有极尖锐的反对态度的时候；对欧拉角的信赖，在基本理论研究来说，是必要的。

欧拉角的哈尔测度有一个简单的形式$\sin\beta\ d\alpha d\beta d\gamma$ 通常在前面添上归一化因子$1/8\pi^2$ 。例如，我们要生成均匀随机取向，使α，γ从0至2π均匀的选随机值；使 $\beta = \arccos(z)$，z从-1 至1均匀的选随机值。

单位四元数，又称欧拉参数，提供另外一种方法来表述三维旋转。这与特殊酉群的描述是等价的。四元数方法用在大多数的演算会比较快捷，概念上比较容易理解，并能避免一些技术上的问题，如万向节锁 (gimbal lock) 现象。因为这些原因，许多高速度三维图形程式制作都使用四元数。

（欧拉角的介绍源于维基百科）