13.4 开发实例——Duncan-Chang 模型的开发

Duncan-Chang 非线性弹性本构模型在国内岩土工程领域中广泛应用，其最大的特点是模型参数易于从常规三轴试验中获得。由于目前境外的大型商业软件，包括 Abaqus、Adina 等软件本身没有提供 Duncan-Chang 本构模型，因此一些用户也将该模型开发到这些商业软件中。本节以 Duncan-Chang ***E-B*** 模型为例，讲述 FLAC3D 中对本构模型进行二次开发的基本思路和方法。

13.4.1 理论描述

根据定义在 stensor.h 文件中应力张量结构体 StnS 的导出函数(EXPORT bool Resoltopris())可以得到当前单元应力场的三个主应力大小及方向，且输出的三个主应力大小是按照弹塑性理论的应力符号约定的。为了统一标准，在编程过程中仍然采用了弹塑性理论的应力符号约定。因此，土力学中大主应力应为输出的最小主应力 $-\sigma_3$，中主应力即为 $-\sigma_2$，小主应力为最大主应力 $-\sigma_1$。

Duncan-Chang 本构模型的切线弹性模量的表达式为

$$E_t = \left(1 - R_f s\right)^2 k P_a \left(\frac{-\sigma_1}{p_a}\right)^n \tag{13-1}$$

体积模量的表达式为

$$K_t = k_b p_a \left(\frac{-\sigma_1}{p_a}\right)^m \tag{13-2}$$

式**错误！未找到引用源。**中应力水平的表达式为

$$s = \frac{(\sigma_3 - \sigma_1)(1 - \sin\varphi)}{2c\cos\varphi - 2\sigma_1 \sin\varphi} \tag{13-3}$$

由于材料的泊松比只能在 0~0.49 之间变化，因此式（13-2）得到的 K_t 需要进行限制，在程序中限制在 $(0.33 \sim 17)E_t$ 之间。

对于卸载情况的处理，回弹模量的表达式为

$$E_{ur} = k_{ur} p_a \left(\frac{-\sigma_3}{p_a}\right)^n \tag{13-4}$$

同时，为了考虑颗粒体材料的抗剪强度随围压增大而降低，内摩擦角改用

$$\varphi = \varphi_0 - \Delta\varphi \log\left(\frac{-\sigma_1}{p_a}\right) \tag{13-5}$$

计算，以反映颗粒材料压碎对内摩擦角降低的影响。计算中，如果输入参数 $\Delta\varphi = 0$，则不考虑内摩擦角降低的影响。

至此，由式（13-1）~（13-5）可以确定 Duncan-Chang ***E-B*** 增量本构方程，其表达式为

$$\Delta\sigma_{ij} = 2G_t\Delta\varepsilon_{ij} + \alpha\Delta\varepsilon_{kk}\delta_{ij} \tag{13-6}$$

其中

$$G_t = \frac{3K_tE_t}{9K_t - E_t} \tag{13-7}$$

$$\alpha = K_t - \frac{2}{3}G_t \tag{13-8}$$

设土的抗拉强度为 σ_p（正值），如果计算中土的小主应力超过了土的抗拉强度 $\left(\sigma_1 < -\sigma_p\right)$，则把 σ_1 修正到零，称为拉裂修正。本节开发的 Duncan-Chang 模型中的拉裂修正采用下式进行：

$$\sigma_x' = \left(1 - \frac{2a}{1+a}\right)\sigma_z' \tag{13-9}$$

$$\tau' = \frac{\sigma_x' - \sigma_z}{\sigma_x - \sigma_z}\tau \tag{13-10}$$

式（13-9）中

$$a = \frac{\sigma_x - \sigma_z}{\sigma_3 - \sigma_1} \tag{13-11}$$

13.4.2 开发流程

Duncan-Chang 模型的开发是在 Visual studio 2005 的环境中进行的，主要开发工作包括 VS2005 的环境设置以及.h 文件和.cpp 文件的修改，开发过程中 UDM 解决方案中的其他文件（包括坐标轴头文件 AXES.H 和本构模型结构体头文件 Conmodel.h）都不用修改。

1. Visual Studio 环境设置

VisualStudio 环境设置步骤如下：

步骤 1 针对需要开发的本构模型，在 FLAC3D 提供的本构模型库中选择开发的蓝本模型，也就是说本构模型的开发工作最好是基于一个已有的本构模型基础上进行，这样便于修改。本节开发的 Duncan-Chang 模型是非线性弹性模型，而且应力塑性修正是按照 Mohr-Coulomb 准则进行的，所以选择 Mohr-Coulomb 模型作为 Duncan-Chang 本构开发的蓝本。

步骤 2 将 UDM 文件夹复制一份，并重命名为 Duncan，作为本次模型开发的工作文件夹。运行 VisualStudio2005，执行【文件】\【打开】\【项目/解决方案...】命令，打开 Duncan 文件夹中的 udm.vcproj 文件。

步骤 3 在 VisualStudio2005 界面左侧的解决方案资源管理器窗口中修改 usermohr.h 和 usermohr.cpp 两个文件的文件名分别为 duncan.h 和 duncan.cpp。将 duncan.cpp 文件第一行声明中的#include "usermohr.h"改为#include "duncan.h"。

步骤 4 执行【项目】\【udm 属性】命令，在 udm 属性对话框中选择【配置属性】\【链接器】\【常规】\【输出文件】，修改输出文件名为：Debug/duncan.dll，这就是模型开

发生成的动态链接库文件。还可以修改【常规】\【调试】以及【常规】\【高级】中的文件名，不过这些不是很重要。

步骤 5　执行【生成】\【生成解决方案】命令，在 VisualStudio 的输出信息中会看到“生成已成功”的提示，且在 Duncan 文件夹中生成了 Debug 文件夹，里面就包含了刚才重新命名的 Duncan.dll 文件。这说明目前为止，该 udm 可以正确地生成 dll 文件。读者在进行模型开发时，要养成“边修改，边生成”的编程习惯，尽早发现错误，提高编程效率。

步骤 6　执行【编辑】\【查找和替换】\【快速替换】命令，选中【查找选项】\【大小写匹配】，将.h 文件和.cpp 文件中的 UserMohrModel 修改为 DuncanChang，其中.h 文件中替换了 4 处，.cpp 文件中替换了 15 处。

2. 修改 Ducan.h 文件

修改 Ducan.h 文件的操作步骤如下：

步骤 1　修改.h 文件中模型的 ID 编号、模型名称字符串、模型输出名称字符串，如下：

```
enum ModelNum { mnDuncanChang=401};
virtual const char *Keyword(void) const { return("duncan"); }
virtual const char *Name(void) const { return("duncan-chang"); }
```

步骤 2　重新定义私有变量，包括模型的参数及迭代所需的中间变量。其中 dSDHistroy，dS3Histroy，dSLHistroy 分别为单元历史上承受的最大主应力差、最大小主应力和最大应力水平。

```
private:
// 邓肯模型的参数列表
//分别对应凝聚力 c，摩擦角 fai,破坏比 Rf，初始模量系数 K，初始模量指数 n,
//为切线泊松比系数 Kb,nb，回弹模量 Kur
double dCohesion,dFriction,dFricDel,dFailRatio,dKe,dNe,dKb,dMb,dKur;
////utility members for ease of calculation
//程序迭代所需的中间变量。
double dF,dFcos,dFsin,dShear,dBulk,dSDHistroy,dS3Histroy,dSLHistroy;
```

3. 修改 Duncan.cpp 文件

修改 Duncan.cpp 文件的操作步骤如下：

步骤 1　由于 Duncan-Chang 模型理论公式中涉及到大气压力这个常数，所以在常量声明中增加一个 dPa 的赋值。

```
static const double dPa       = 101325.0;//标准大气压
```

步骤 2　对模型参数和中间变量进行初始化。

```
DuncanChang::DuncanChang(bool bRegister)
            :ConstitutiveModel(mnDuncanChang,bRegister), dCohesion(0.0),dFriction(0.0),
  dFailRatio(0.0), dFricDel(0.0), dKe(0.0),dNe(0.0),dKb(0.0),dMb(0.0), dKur(0.0),
  dF(0.0),dFcos(0.0),dFsin(0.0),dShear(0.0),dBulk(0.0),
  dSDHistroy(0.0), dS3Histroy(0.0), dSLHistroy(0.0) {
}
```

步骤 3　设置 Properties()函数中变量名称的字符串。

```
const char **DuncanChang::Properties(void) const {
  static const char *strKey[] = {
//double dCohesion,dFriction,dFailRatio,dK,dNe,dKB,dMb,dKur;

"cohesion","friction","fricdel","ratiofail","ke","ne",
                                                    "kb","mb","kur",0
  };
  return(strKey);
}
```

步骤 4　设置 DuncanChang()函数中模型参数的返回值，注意必须按照 Properties()函数中变量名称的顺序进行设置。

```
double DuncanChang::GetProperty(unsigned ul) const {
  switch (ul) {
                                                    ////double
dCohesion,dFriction,dFailRatio,dKe,dNe,dKb,dM,dKur;
    case 1:  return(dCohesion);
    case 2:  return(dFriction);
    case 3:  return(dFricDel);
    case 4:  return(dFailRatio);
    case 5:  return(dKe);
    case 6:  return(dNe);
    case 7:  return(dKb);
    case 8:  return(dMb);
    case 9:  return(dKur);
  }
  return(0.0);
}
```

步骤 5　设置 SetProperty()函数中的模型变量赋值，也要按照 Properties()函数中变量名称的顺序。

```
void DuncanChang::SetProperty(unsigned ul,const double &dVal) {
  switch (ul) {
                                                    ////double
dCohesion,dFriction,dFailRatio,dKe,dN,dKB,dMb,dKur;
    case 1: dCohesion   = dVal;   break;
    case 2: dFriction   = dVal;   break;
                                                    case 3: dFricDel    =
```

```
dVal;   break;
          case 4: dFailRatio = dVal;   break;
          case 5: dKe          = dVal;   break;
          case 6: dNe          = dVal;   break;
          case 7: dKb          = dVal;   break;
          case 8: dMb          = dVal;   break;
                                                                    case      9:      dKur
= dVal;   break;
                                                                    }
     }
```

步骤 6　修改 Copy()函数中的变量名称。

```
     const char *DuncanChang::Copy(const ConstitutiveModel *cm) {
       //Detects type mismatch error and returns error string. otherwise returns 0
       const char *str = ConstitutiveModel::Copy(cm);
       if (str) return(str);
       DuncanChang *mm = (DuncanChang *)cm;
       dCohesion   = mm->dCohesion;
       dFriction   = mm->dFriction;
       dFricDel    = mm->dFricDel;
       dFailRatio = mm->dFailRatio;
       dKe          = mm->dKe;
       dNe          = mm->dNe;
       dKb          = mm->dKb;
       dMb          = mm->dMb;
       dKur         = mm->dKur;
       return(0);
     }
```

步骤 7　修改 Initialize()函数。在编程中发现，Initialize()函数与 Run()函数之间程序将自动调用侧限模量函数，如果 ConfinedModulus()的返回值为 0，则出现结点刚度为零的错误提示。由于 Duncan-Chang 模型没有 E,μ 等弹性参数，为了避免侧限模量函数的返回值为 0，所以在 Initialize()函数中计算体积模量和剪切模量。Duncan-Chang 模型的体积模量和剪切模量是根据切线弹性模量和切线体积模量换算得到的，同时要进行加卸载判断。由于需要调用单元的应力结果，所以在 Initialize 函数定义时增加了 State *ps 的声明。

```
     const char *DuncanChang::Initialize(unsigned uDim,State *ps) {
       if ((uDim!=2)&&(uDim!=3)) return("Illegal dimension in DuncanChang constitutive model");
       Axes aDir;
       double dPrinMin,dPrinMid,dPrinMax,sdif=0.0,psdif=0.0;
```

```
        int icase=0;
        bool    bFast=ps->stnS.Resoltopris(&dPrinMin,&dPrinMid,&dPrinMax,&aDir,uDim,&icase,
&sdif, &psdif);
        double dSD = dPrinMax - dPrinMin;//主应力差>0
        if (dSD > dSDHistroy)
        dSDHistroy = dSD;
        if (-1.0*dPrinMax > dS3Histroy)
        dS3Histroy = -1.0*dPrinMax;
        //考虑摩擦角随围压的影响
        double dlog = (dS3Histroy != 0.0)? log10(dS3Histroy / dPa) : 0.0;
        dF = dFriction - dFricDel * dlog;
        dFcos = cos(dF * dDegRad);
        dFsin = sin(dF * dDegRad);
        //(2)计算应力水平
        double dSF = 2* dCohesion * dFcos + (-2.0) * dPrinMax * dFsin;//破坏摩尔圆直径
        double dSL = (dSF != 0)? dSD * (1-dFsin) / dSF : 0.0;         //除数(no p)
        if (dSL > dSLHistroy)
                                                                    dSLHistroy = dSL;
        if (dSL >= 0.99) dSL = 0.99;
        //(3)加卸载判断，计算模量
        double dSPa = dS3Histroy / dPa;
        double dEi = dKe *dPa * pow(dSPa,dNe);
        double dEt;
        if (dSD < dSDHistroy && dSL < dSLHistroy)
        {
    dEt = dKur *dPa * pow(dSPa,dNe);
        }
        else
                                                                    dEt = dEi * (1 -
dFailRatio * dSL) * (1 - dFailRatio * dSL);
        //设置最小模量 Emin
        double dEtmin = 0.25 * dKe * dPa * pow(0.02, dNe);
        if (dEt <= dEtmin)
                                                                  dEt = dEtmin;
        //根据 0<v<0.49 确定体积模量的范围 0.33~17Et
        dBulk = dKb * dPa * pow(dSPa,dMb);
```

```
    dBulk = (dBulk < 0.33*dEt)? 0.33 * dEt : dBulk;
    dBulk = (dBulk > 17.0*dEt)? 17.0 * dEt : dBulk;
    dShear = 3 * dBulk * dEt / (9 * dBulk - dEt);   // Divide!
    return(0);
  }
```

步骤 8 修改 Run()函数。Run()函数是整个模型开发中最重要的函数。主要包括模量计算、摩擦角修正、加卸载判断、泊松比修正、计算新的应力、进行塑性判断与修正。

```
  const char *DuncanChang::Run(unsigned uDim,State *ps){
    if ((uDim!=3)&&(uDim!=2)) return("Illegal dimension in Mohr constitutive model");
    if(ps->dHystDampMult > 0.0) HDampInit(ps->dHystDampMult);
    if (ps->mState & mShearNow)
        ps->mState = (unsigned long)(ps->mState | mShearPast);
    ps->mState = (unsigned long)(ps->mState & ~mShearNow);
    if (ps->mState & mTensionNow)
        ps->mState = (unsigned long)(ps->mState | mTensionPast);
    ps->mState = (unsigned long)(ps->mState & ~mTensionNow);
    Axes aDir;
    double dPrinMin,dPrinMid,dPrinMax,sdif=0.0,psdif=0.0;
    int icase=0;
    bool  bFast=ps->stnS.Resoltopris(&dPrinMin,&dPrinMid,&dPrinMax,&aDir,uDim,  &icase,
&sdif, &psdif);
    double dSD = dPrinMax - dPrinMin;//主应力差>0
    if (dSD > dSDHistroy)
                                                        dSDHistroy = dSD;
    if (-1.0*dPrinMax > dS3Histroy)
                                                        dS3Histroy      =
-1.0*dPrinMax;
    //考虑摩擦角随围压的影响
    double dlog = (dS3Histroy != 0.0)? log10(dS3Histroy / dPa) : 0.0;
    dF = dFriction - dFricDel * dlog;
    dFcos = cos(dF * dDegRad);
    dFsin = sin(dF * dDegRad);
    /*/下面语句为调试所用
    char temp[200];
    sprintf(temp,"%f",dF);
    return(temp);
    //调试结束*/
```

```
        //(2)计算应力水平
        double dSF = 2* dCohesion * dFcos + (-2.0) * dPrinMax * dFsin;//破坏摩尔圆直径
        double dSL = (dSF != 0)? dSD * (1-dFsin) / dSF : 0.0;         //除数(no p)
        if (dSL > dSLHistroy)
                                                                    dSLHistroy = dSL;
        if (dSL >= 0.99) dSL = 0.99;
        //(3)加卸载判断，计算模量
        double dSPa = dS3Histroy / dPa;
        double dEi = dKe *dPa * pow(dSPa,dNe);
        double dEt;
        if (dSD < dSDHistroy && dSL < dSLHistroy)
        {
                                                                    dEt = dKur *dPa *
pow(dSPa,dNe);
        }
        else
                                                                    dEt = dEi * (1 -
dFailRatio * dSL) * (1 - dFailRatio * dSL);
        //2006-06-11 add Emin
        double dEtmin = 0.25 * dKe * dPa * pow(0.02, dNe);
        if (dEt <= dEtmin)
                                                                  dEt = dEtmin;
        //根据 0<v<0.49 确定体积模量的范围 0.33~17Et
        dBulk = dKb * dPa * pow(dSPa,dMb);
        dBulk = (dBulk < 0.33*dEt)? 0.33 * dEt : dBulk;
        dBulk = (dBulk > 17.0*dEt)? 17.0 * dEt : dBulk;
        dShear = 3 * dBulk * dEt / (9 * dBulk - dEt);   // Divide!
        //(5)计算应力增量
        //根据模量 Kt,Et 求剪切模量
        double dE1      = dBulk +  d4d3 * dShear;
        double dE2      = dBulk - d2d3 * dShear;
        //利用 M,λ模型
        double dE11 = ps->stnE.d11;
        double dE22 = ps->stnE.d22;
        double dE33 = ps->stnE.d33;
        ps->stnS.d11 += dE11 * dE1 + (dE22 +  dE33) * dE2;
```

```
        ps->stnS.d22 += (dE11 + dE33) * dE2 + dE22 * dE1;
        ps->stnS.d33 += (dE11 + dE22) * dE2 + dE33 * dE1;
        ps->stnS.d12 += ps->stnE.d12 * 2 * dShear;
        ps->stnS.d13 += ps->stnE.d13 * 2 * dShear;
        ps->stnS.d23 += ps->stnE.d23 * 2 * dShear;
        //(6)计算新的主应力分量
        double dSDM = dPrinMax - dPrinMin;//主应力差>0
        //剪坏修正
        double dNPH = (1.0+ dFsin)/(1.0-dFsin);
        if( dPrinMin - dPrinMax * dNPH + 2.0*dCohesion*sqrt(dNPH) < 0.0)
        {
                                                                ps->mState      =
(unsigned long)(ps->mState | 0x01);
                                                                double d13Copy =
ps->stnS.d13;
                                                                ps->stnS.d13    =
(-1.0 * (dPrinMax+ dPrinMin) *
                                                                  dFsin  +  2  *
dCohesion*dFcos)/dSDM;
                                                                ps->stnS.d33    =
-0.5*(ps->stnS.d11 + ps->stnS.d33) + 0.5*(ps->stnS.d11 - ps->stnS.d33)*
                                                                  ps->stnS.d13  /
d13Copy;
                                                                ps->stnS.d11    =
-0.5*(ps->stnS.d11 + ps->stnS.d33) - 0.5*(ps->stnS.d11 - ps->stnS.d33)*
                                                                  ps->stnS.d13  /
d13Copy;
                                                                double da2 = 2.0 *
(-1.0*ps->stnS.d33 * dFsin + dCohesion * dFcos) /
                                                                  ((ps->stnS.d11
- ps->stnS.d33) * dFsin + dSDM);
                                                                ps->stnS.d13   *=
da2;
                                                                ps->stnS.d11    =
ps->stnS.d33 - da2 * (ps->stnS.d33 - ps->stnS.d11);
        }
        return(0);
```

13.4.3 调试与验证

模型的调试包括两个方面，一是对程序编辑过程中出现的语法错误进行改正；二是根据FLAC^{3D} 应用新模型时的计算结果对程序进行改正和完善。对于第一方面的调试，用户参照C++语言的语法规定即可改正。对于第二方面的调试，往往需要将模型的计算结果与理论值或已有的正确结果进行比较，以说明模型编辑的正确性和适用性，这个过程也称为模型的验证过程。

用户开发的本构模型需要进行严格的调试和验证，遵循的原则是：由少单元到多单元、由简单到复杂，逐步完善。在模型调试初期，常用一个单元进行计算，称为 One Zone Test。因为一个单元本身的应力和应变状态简单，可以通过理论计算得到单元和节点的响应，这样便于和 FLAC^{3D} 计算结果进行比较。

1. 调试方法

程序调试有 2 种主要方法：

- 在 VC++的工程设置中将 FLAC^{3D} 软件中的 EXE 文件路径加入到程序的调试范围中，并将 FLAC^{3D} 自带的 DLL 文件加入到附加动态链接库(Additional DLLs)中，然后在 Initialize()或 Run()函数中设置断点，进行调试；
- 在程序文件中加入 return()语句，这样可以将希望得到的变量（或表达式）的值以错误提示的形式在 FLAC^{3D} 窗口中得到。下面的语句就是一个调试模型参数体积模量的一个例子。注意，用 sprint()函数进行数据格式转换时，需要在.cpp 文件开始时增加#include <stdio.h>标准输入输出头文件的调用。

```
char temp[200];
sprintf(temp,"%f",dBulk);
return(temp);
```

2. 三轴试验模拟的验证

为了对开发好的 Duncan-Chang 本构模型进行测试，本文采用三轴试验的数值模拟来验证开发模型的正确性。Duncan-Chang *E-B* 模型参数如表 0-3 所示。

表 0-3 Duncan-Chang 模型参数

参数	数值	参数	数值	参数	数值
c (kPa)	110	R_f	0.79	K_b	303
φ_0 (^{o})	48.5	k	704	m	0.18
$\Delta\varphi$ (^{o})	0.0	n	0.38	K_{ur}	844.8

计算模型的尺寸采用大型三轴仪的试样大小，同时为了简化边界条件，采用 1 个边长为600mm 的立方体 brick 单元，底面为竖直向位移约束，周围的四个面采用应力边界条件，模型顶部荷载使用 FISH 函数分级施加，每级 10kPa，不平衡力比为默认的 10^{-5}。计算中分别采用了 3 种不同的围压条件（300、600 和 900kPa），利用开发的 Duncan-Chang 模型为 FLAC^{3D} 的力学模型。计算结果主要分析偏应力-轴向应变关系及体积应变-轴向应变的关系，同时与 Duncan-Chang 的理论值进行对比分析以了解计算结果的正确性。单元的轴向应变通过单元顶部的竖向位移计算得到，体积应变通过 FISH 函数 z_vsi()获得。Duncan-Chang 模型关于轴向应变和体积应变增量的理论值分别由式（13-12）和式（13-13）得到。

$$\varepsilon_a = \frac{\sigma_3 - \sigma_1}{(1 - R_f s) k P_a \left(\frac{-\sigma_1}{p_a} \right)^n} \tag{13-12}$$

$$\Delta\varepsilon_v = \frac{\Delta(\sigma_3 - \sigma_1)}{3K_t} \tag{13-13}$$

例 13.1：Duncan-Chang 模型的三轴试验模拟算例

```
config cppudm  ;必须设置 cppudm 的选项才可以进行模型载入
model load[22] DuncanChang.dll
gen zon bri p1 .6 0 0 p2 0 .6 0 p3 0 0 .6 size 1 1 1;三轴试验尺寸：0.6*0.6*0.6 m
model duncan
prop cohesion 110e3 friction 48.5 fricDel 0.0 ratiofail 0.79 ke 704 ne 0.38 kb 303 mb 0.18 kur
844.8
fix z ran z -.01 .01 ;模型底部边界的竖直向速度约束
def sigma3 ;定义一个 sigma 变量便于进行多工况计算
  sigma3=-600e3
end
sigma3
;设置初始应力
app nstress sigma3 ran x -.01 .01
app nstress sigma3 ran x .59 .61
app nstress sigma3 ran y -.01 .01
app nstress sigma3 ran y .59 .61
app nstress sigma3 ran z .59 .61
;ini den 2190 ;为了与理论值作比较，不考虑重力、密度的作用
ini szz sigma3
ini syy sigma3
ini sxx sigma3
solve ;得到加载前的初始应力
ini xdis 0 ydis 0 zdis 0 xvel 0 yvel 0 zvel 0 ;位移和速度清零
tab 1 name loads ;建立一个空表，用来保存荷载-沉降曲线
plo tab 1
;第 1 次加卸载
def load1
  p_gp = gp_near(0,0,0.6)
```

[22] model load 后面可以跟完整路径，中英文均可，但是中间不能有空格，比如 Program files 这样的路径就会出错。

```
    loop n(1,50)
      zss_load= sigma3 - float(n)*20e3   ;加载 1000kPa
      command
        app nstress zss_load ran z .59 .61
        solve
      endcommand
      z_dis = -1.0 *gp_zdisp(p_gp) / 0.6   ;得到应变
      z_load = (sigma3 - zss_load)          ;得到主应力差
      command
        tab 1   z_dis z_load                  ;分别保存应变和主应力差
      endcommand
    end_loop
end
load1
save 13-1.sav
```

对比结果如图 0-5 和图 0-6 所示。从两图中可以看出，FLAC^{3D} 的计算结果同 Duncan-Chang 理论值十分接近，轴向应变和体积应变随轴向应变的变化关系十分吻合，表明在 FLAC^{3D} 中开发实现 Duncan- Chang 模型的计算结果是正确的。

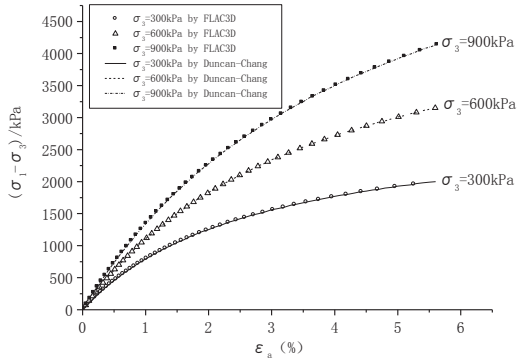

图 0-5 不同围压下偏应力与轴向应变曲线

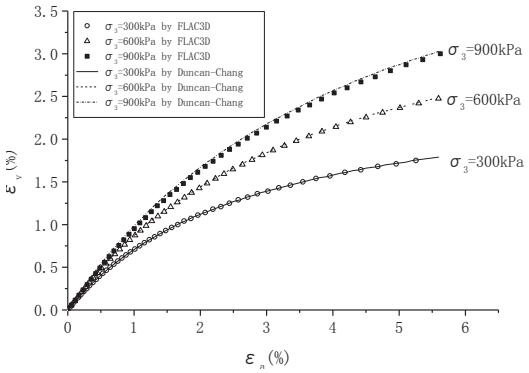

图 0-6 不同围压下体积应变与轴向应变曲线

3. 应力路径试验模拟的验证

计算中分别考虑了两种围压（600 和 900kPa）下的加卸载过程来测试 FLAC^{3D} 开发的 Duncan-Chang 模型对应力路径的适应性。对于卸载过程的轴向应变增量的理论值按式（13-14）计算。两种围压下的加卸载循环过程如表 0-4 所示。

$$\Delta\varepsilon_a = \frac{\Delta(\sigma_3 - \sigma_1)}{k_{ur} p_a \left(\frac{-\sigma_1}{p_a}\right)^n} \tag{13-14}$$

表 0-4 应力路径试验的偏应力历史(单位：kPa)

围压	初始	加载 1	卸载 1	加载 2	卸载 2	加载 3	卸载 3	加载 4
600	0	1000	500	2000	500	3000	500	3200
900	0	2000	500	3000	500	4000	500	4300

例 13.2 给出了卸载 1 和加载 2 过程的命令文件。

例 13.2：应力路径试验的模拟

```
rest 13-1.sav
def unload1
  p_gp = gp_near(0,0,0.6)
  loop m(1,25)
    zss_load= sigma3 - 1000e3 + float(m)*20e3   ;卸载 500kPa
    command
      app nstress zss_load ran z .59 .61
      solve
    endcommand
    z_dis = -1.0 *gp_zdisp(p_gp) / 0.6
    z_load = (sigma3 - zss_load)
    command
      tab 1   z_dis zss_load
    endcommand
  endloop
end
unload1
set log on
set logfile 1.log
pri tab 1
set log off
;第 2 次加卸载
```