有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤

Theoretical Foundation of Finite Element Method and Research on Internal Implementation of Abaqus Series 20: UEL User Subroutine Development Steps

SnowWave02 关注 Focus

2020年4月22日 14:53 April 22, 2020 14:53 浏览：4574 Views: 4574 评论：81 Comments: 81 收藏：27 Favorited: 27

（原创，转载请注明出处） (Original, please indicate the source for reproduction)

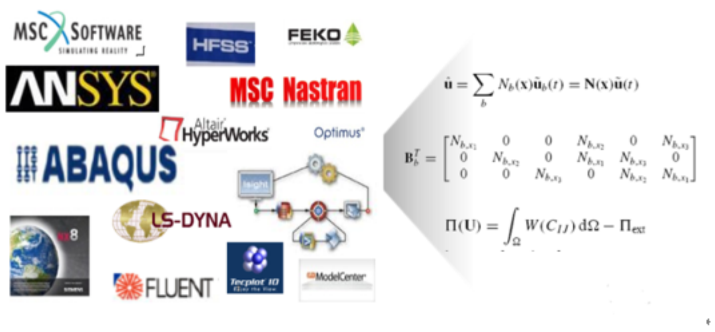
有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图1
有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图2

==概述== ==Overview==

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图3

本系列文章研究成熟的有限元理论基础及在商用有限元软件的实现方式。有限元的理论发展了几十年已经相当成熟，商用有限元软件同样也是采用这些成熟的有限元理论，只是在实际应用过程中，商用CAE软件在传统的理论基础上会做相应的修正以解决工程中遇到的不同问题，且各家软件的修正方法都不一样，每个主流商用软件手册中都会注明各个单元的理论采用了哪种理论公式，但都只是提一下用什么方法修正，很多没有具体的实现公式。商用软件对外就是一个黑盒子，除了开发人员，使用人员只能在黑盒子外猜测内部实现方式。

This series of articles studies the mature finite element theoretical foundation and its implementation methods in commercial finite element software. The development of finite element theory has matured over decades, and commercial finite element software also adopts these mature finite element theories. However, in the actual application process, commercial CAE software will make corresponding corrections on the basis of traditional theories to solve different problems encountered in engineering, and the correction methods of each software are different. Each mainstream commercial software manual specifies which theoretical formula each element uses, but only mentions the correction method, and many do not provide specific implementation formulas. Commercial software is essentially a black box, and users can only guess its internal implementation methods from outside, except for developers.


$\dot{\mathbf{u}} = \sum_b N_b(\mathbf{x})\dot{\mathbf{u}}_b(t) = \mathbf{N}(\mathbf{x})\dot{\mathbf{u}}(t)$
$\mathbf{B}_b^T = \begin{bmatrix} N_{b,x_1} & 0 & 0 & N_{b,x_2} & 0 & N_{b,x_3} \\ 0 & N_{b,x_2} & 0 & N_{b,x_1} & N_{b,x_3} & 0 \\ 0 & 0 & N_{b,x_3} & 0 & N_{b,x_2} & N_{b,x_1} \end{bmatrix}$
$\Pi(\mathbf{U}) = \int_\Omega W(C_{IJ})\,d\Omega - \Pi_{ext}$

一方面我们查阅各个主流商用软件的理论手册并通过进行大量的资料查阅猜测内部修正方法，另一方面我们自己编程实现结构有限元求解器，通过自研求解器和商软的结果比较来验证我们的猜测，如同管中窥豹一般来研究的修正方法，从而猜测商用有限元软件的内部计算方法。我们关注CAE中的结构有限元，所以主要选择了商用结构有限元软件中文档相对较完备的Abaqus来研究内部实现方式，同时对某些问题也会涉及其它的Nastran/Ansys等商软。为了理解方便有很多问题在数学上其实并不严谨，同时由于水平有限可能有许多的理论错误，欢迎交流讨论，也期待有更多的合作机会。

On one hand, we consult the theoretical manuals of various mainstream commercial software and guess the internal correction methods through extensive literature review. On the other hand, we program our own structural finite element solver and verify our guesses by comparing the results with those of commercial software. We study the correction methods like a glimpse through a tube, thus guessing the internal calculation methods of commercial finite element software. Since we focus on structural finite elements in CAE, we mainly choose Abaqus, which has relatively complete documentation among commercial structural finite element software, to study the internal implementation methods, and we will also involve other commercial software such as Nastran/Ansys for some issues. Many problems are not mathematically rigorous for the sake of understanding convenience, and due to our limited level, there may be many theoretical errors. We welcome discussions and look forward to more cooperation opportunities.

iSolver介绍视频： iSolver Introduction Video:

http://www.jishulink.com/college/video/c12884

==第20篇：UEL用户子程序开发步骤== ==20th Article: UEL User Subroutine Development Steps==

用户子程序主要是将用户特定的材料本构模型和单元算法等公式编写为计算机语言表示的公式，并实现和商软求解器之间的交互迭代。

User subroutines mainly involve writing user-specific material constitutive models and element algorithms into formulas expressed in computer languages, and realizing the interaction and iteration between them and commercial solvers.

常用的商业有限元软件都提供了用户自定义子程序的功能，且一般都是Fortran语言开发，Fortran是上世纪70年代的语言，相对现代化的流行语言编写，格式要求非常严格，编译调试都比较繁琐，使得开发效率低下，而且接口限制较多，除了商软提供的功能外用户基本没法改动，灵活性较差。由于用户子程序很多都涉及复杂的公式编写，用户除了需要扎实的理论基础外，还需要较强的能将公式表达为Fortran语言的编程能力，这对非计算机专业出身的人来说往往在浪费了很多额外精力，使得很多理论高手都对用户子程序望而却步，难以入门。

Most commercial finite element software provides the function of user-defined subroutines, and most of them are developed in Fortran language. Fortran is a language from the 1970s, which has very strict formatting requirements when compared to modern popular languages, making compilation and debugging more cumbersome. This leads to a low development efficiency and limited flexibility, as users can hardly make changes beyond the features provided by the commercial software. Due to the complex formula writing involved in many user subroutines, users not only need a solid theoretical foundation but also strong programming skills to express formulas in Fortran language. This often results in a waste of extra effort for those who are not from a computer science background, making many theoretical experts shy away from user subroutines and difficult to get started.

在实际工作中，很多工程师用Matlab来编写和推导公式，Matlab被认为是市面上最接近草稿纸上推导公式的一款软件了，而且有限元在数值层面上的计算其实就是矩阵运算，所以Matlab这种数据按矩阵来组织非常适合用来开发有限元相关的程序。而现在市面上还没有采用Matlab来开发商软子程序的案例。iSolver是市面上第一款基于

Matlab来开发商软用户子程序的软件工具，支持用Matlab编写和调试用户子程序。iSolver子程序的接口完全按照Abaqus的标准实现，而Abaqus的子程序接口在近几年内已经基本不再变化了，同样的，虽然iSolver在不断发展，但iSolver子程序接口将维持不变，所有在iSolver上编写的算法子程序都只要维护自己的算法部分就行，而不是维护整个有限元求解的整个过程。

In practical work, many engineers use Matlab to write and deduce formulas, Matlab is considered to be the software on the market that is closest to the derivation of formulas on draft paper, and finite element calculations in the numerical level are actually matrix operations, so Matlab, which organizes data in matrix form, is very suitable for developing finite element-related programs. However, there is no case on the market yet of using Matlab to develop software subroutines. iSolver is the first software tool on the market based on Matlab to develop software user subroutines, which supports writing and debugging user subroutines in Matlab. The interface of iSolver subroutines is completely implemented according to the Abaqus standard, and the Abaqus subroutine interface has not changed much in recent years. Similarly, although iSolver is constantly evolving, the iSolver subroutine interface will remain unchanged. All algorithm subroutines written on iSolver only need to maintain their own algorithm part, rather than maintaining the entire finite element solution process.

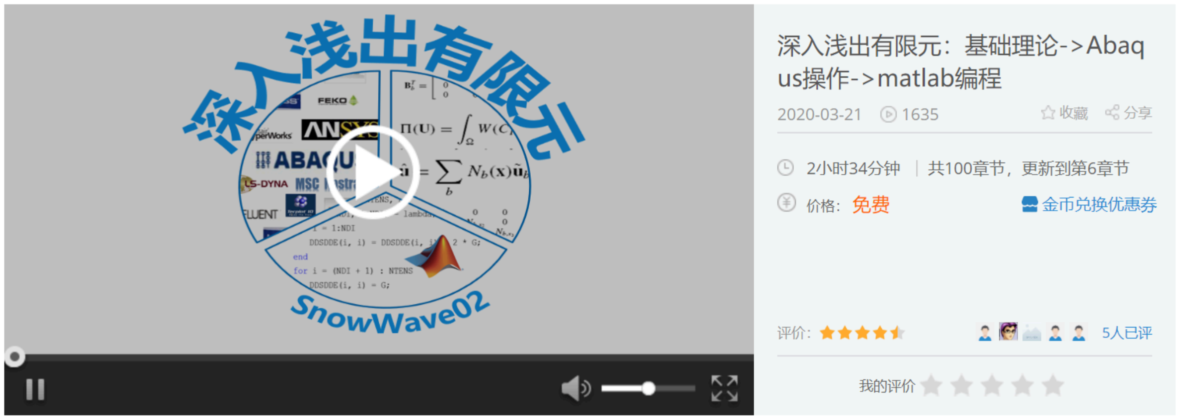
前面第八、九篇介绍了UMAT用户自定义材料的开发，这里将介绍UEL用户自定义单元的开发，本文首先简单的讨论了UEL的一般含义，并详细的介绍了基于Fortran和Matlab两种方式的UEL的开发步骤，对比发现开发步骤基本相同，同时采用Matlab更加高效和灵活。具体的开发过程可以参考下面我们的Step by Step的录像，包括了整个的有限元基础理论和我们对Abaqus中的单元理解：

The previous eighth and ninth articles introduced the development of UMAT user-defined materials, and this article will introduce the development of UEL user-defined elements. This article first briefly discusses the general meaning of UEL and introduces the development steps of UEL based on Fortran and Matlab in detail. The comparison shows that the development steps are basically the same, and Matlab is more efficient and flexible. The specific development process can be referred to in our Step by Step video below, which includes the entire finite element basic theory and our understanding of elements in Abaqus:

https://www.jishulink.com/college/video/c14948

深入浅出有限元：基础理论->Abaqus操作->matlab编程

A Deep Dive into Finite Element Method: Basic Theory -> Abaqus Operation -> Matlab Programming



深入浅出有限元：基础理论->Abaqus操作->matlab编程

2020-03-21 1635

2小时34分钟 | 共100章节，更新到第6章节

价格：免费 金币兑换优惠券

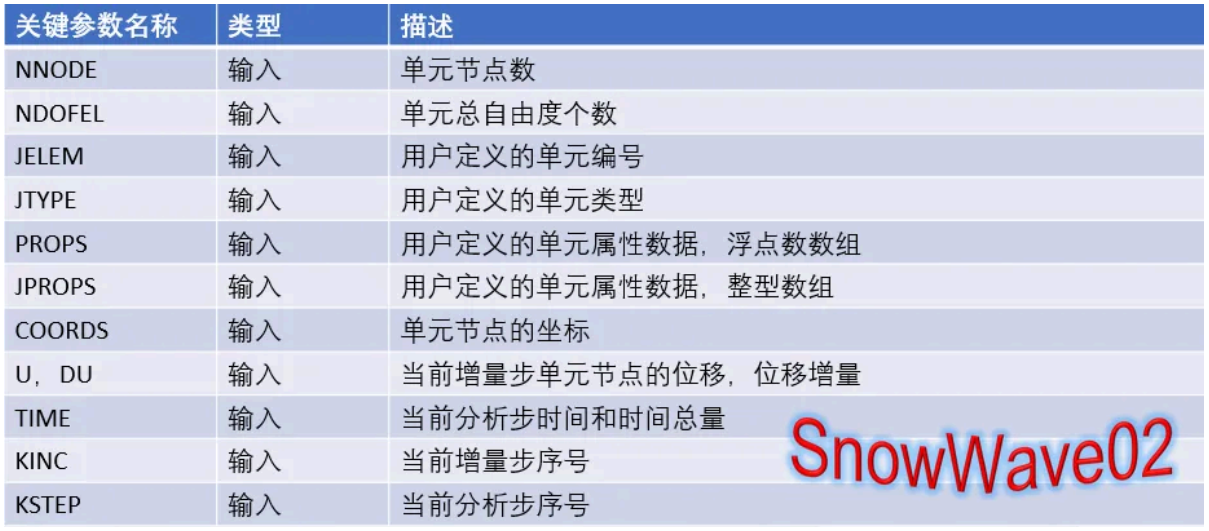
5人已评

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图8

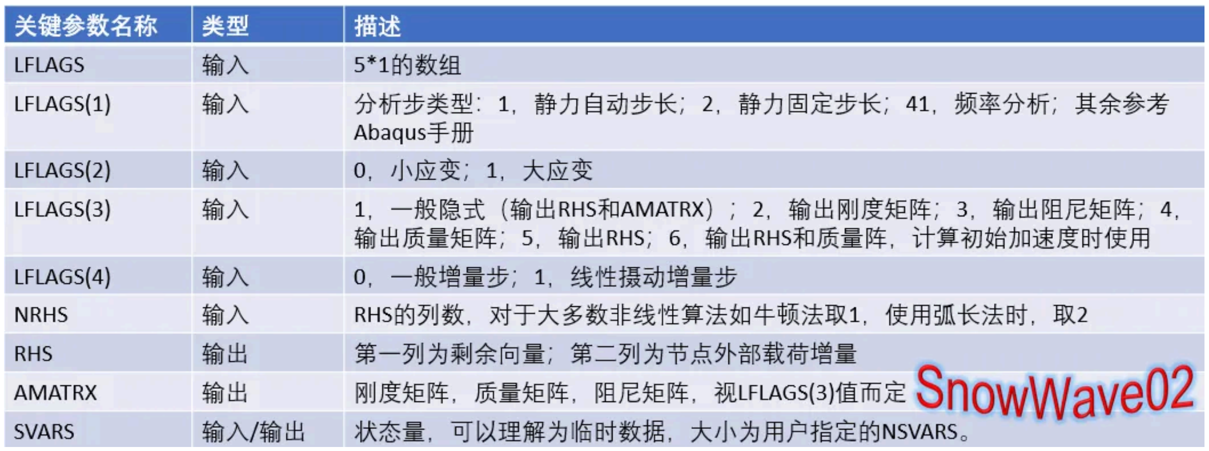
1.1 UEL的关键输入输出参数 1.1 Key Input and Output Parameters of UEL

UEL网上资料很多，大家可以很容易查看，但大部分资料都只是简单提供UEL算例，这里我们列出了UEL接口的关键输入输出参数，如下表所示：

There are many online resources about UEL, which can be easily accessed. However, most of the materials only provide simple UEL examples. Here, we list the key input and output parameters of the UEL interface as shown in the table below:



关键参数名称	类型	描述
NNODE	输入	单元节点数
NDOFEL	输入	单元总自由度个数
JELEM	输入	用户定义的单元编号
JTYPE	输入	用户定义的单元类型
PROPS	输入	用户定义的单元属性数据，浮点数数组
JPROPS	输入	用户定义的单元属性数据，整型数组
COORDS	输入	单元节点的坐标
U，DU	输入	当前增量步单元节点的位移，位移增量
TIME	输入	当前分析步时间和时间总量
KINC	输入	当前增量步序号
KSTEP	输入	当前分析步序号



关键参数名称	类型	描述
LFLAGS	输入	5*1的数组
LFLAGS(1)	输入	分析步类型：1，静力自动步长；2，静力固定步长；41，频率分析；其余参考Abaqus手册
LFLAGS(2)	输入	0，小应变；1，大应变
LFLAGS(3)	输入	1，一般隐式（输出RHS和AMATRX）；2，输出刚度矩阵；3，输出阻尼矩阵；4，输出质量矩阵；5，输出RHS；6，输出RHS和质量阵，计算初始加速度时使用
LFLAGS(4)	输入	0，一般增量步；1，线性摄动增量步
NRHS	输入	RHS的列数，对于大多数非线性算法如牛顿法取1，使用弧长法时，取2
RHS	输出	第一列为剩余向量；第二列为节点外部载荷增量
AMATRX	输出	刚度矩阵，质量矩阵，阻尼矩阵，视LFLAGS(3)值而定
SVARS	输入/输出	状态量，可以理解为临时数据，大小为用户指定的NSVARS。

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图13

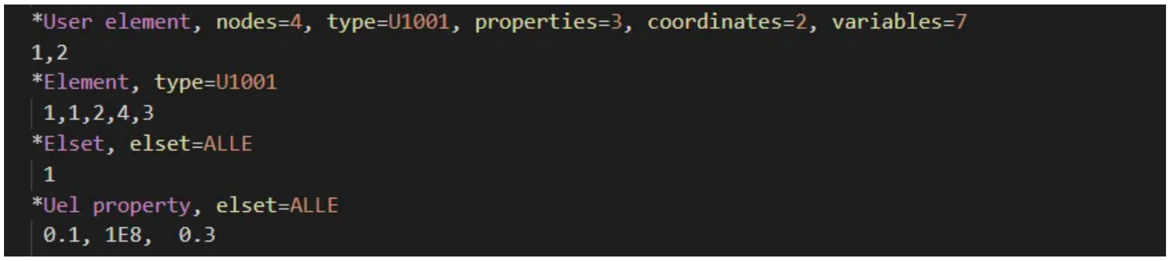
1.2 基于Fortran的Abaqus的UEL的开发步骤

1.2 Development Steps of Abaqus UEL Based on Fortran

1.2.1 在inp文件中定义UEL 1.2.1 Defining UEL in the inp file

Abaqus中只能通过修改手动inp文件完成用户UEL的定义，通常包含以下关键字及相应属性，如图所示：

In Abaqus, user UEL definition can only be achieved by modifying the manual inp file, which typically includes the following keywords and corresponding attributes, as shown in the figure:



```
*User element, nodes=4, type=U1001, properties=3, coordinates=2, variables=7
1,2
*Element, type=U1001
 1,1,2,4,3
*Elset, elset=ALLE
 1
*Uel property, elset=ALLE
 0.1, 1E8,  0.3
```

- *User element关键字，用于定义单元基本信息，包括以下属性：nodes为单元节点数；type为单元类型，U表示为用户自定义单元，1001表示单元类型编号，一般从1000开始；properties为浮点数属性个数，一般用于输入单元对应的section数据和材料数据；coordinates为坐标维数；variables为状态量个数，一般求解计算过程中需要在迭代步之间传递的变量；第二行开始定义节点激活的自由度；

  *User element keyword, used to define basic element information, including the following properties: nodes for the number of element nodes; type for the element type, U represents a user-defined element, 1001 represents the element type number, generally starting from 1000; properties for the number of floating-point attributes, generally used for inputting the section data and material data corresponding to the element; coordinates for the number of coordinate dimensions; variables for the number of state variables, generally required to be passed between iteration steps during the solution process; the second line onwards defines the degrees of freedom for node activation;
- *Uel property关键字，与*User element关键字中的properties对应，第二行开始定义具体的数据。

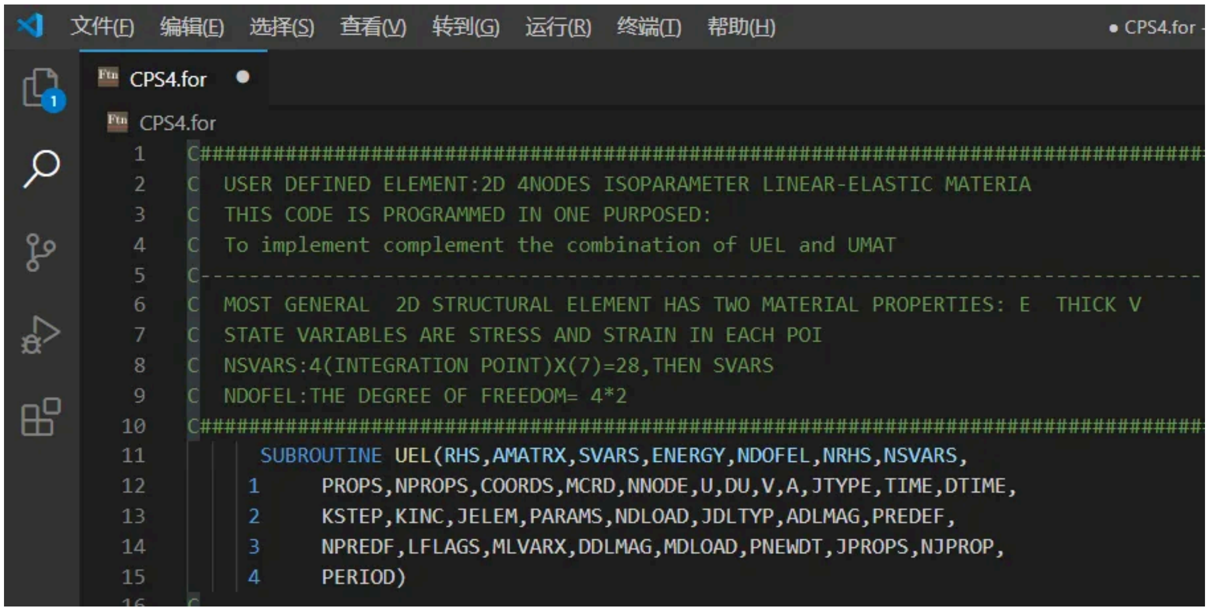
  *Uel property keyword corresponds to the properties keyword in *User element, and specific data is defined starting from the second line.

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图16

1.2.2 编写 1.2.2 Write

使用任意编辑器编写.for文件，推荐使用Visual Studio Code，微软开源的轻量化代码编辑器，配置灵活高效：

Write a .for file using any editor, Visual Studio Code is recommended, a lightweight open-source code editor from Microsoft with flexible and efficient configuration:


文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H)
● CPS4.for
CPS4.for ●
CPS4.for

```
C##########################################################################
C  USER DEFINED ELEMENT:2D 4NODES ISOPARAMETER LINEAR-ELASTIC MATERIA
C  THIS CODE IS PROGRAMMED IN ONE PURPOSED:
C  To implement complement the combination of UEL and UMAT
C--------------------------------------------------------------------------
C  MOST GENERAL  2D STRUCTURAL ELEMENT HAS TWO MATERIAL PROPERTIES: E  THICK V
C  STATE VARIABLES ARE STRESS AND STRAIN IN EACH POI
C  NSVARS:4(INTEGRATION POINT)X(7)=28,THEN SVARS
C  NDOFEL:THE DEGREE OF FREEDOM= 4*2
C##########################################################################
      SUBROUTINE UEL(RHS,AMATRX,SVARS,ENERGY,NDOFEL,NRHS,NSVARS,
     1     PROPS,NPROPS,COORDS,MCRD,NNODE,U,DU,V,A,JTYPE,TIME,DTIME,
     2     KSTEP,KINC,JELEM,PARAMS,NDLOAD,JDLTYP,ADLMAG,PREDEF,
     3     NPREDF,LFLAGS,MLVARX,DDLMAG,MDLOAD,PNEWDT,JPROPS,NJPROP,
     4     PERIOD)
```

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图19

1.2.3 编译（可选） 1.2.3 Compilation (optional)

Abaqus没有自带Fortran编译器，所以用户需要自己去安装Fortran编译器和Visual Studio Build Tools，并配置相应环境。具体配置过程与UMAT一致，可以查看我们关于环境配置的视频：

Abaqus does not come with a Fortran compiler, so users need to install a Fortran compiler and Visual Studio Build Tools themselves, and configure the environment accordingly. The specific configuration process is consistent with UMAT, and you can refer to our video on environment configuration:

https://www.jishulink.com/college/video/c13034?chapter=1

在环境配置完成之后，打开命令提示框，输入命令Abaqus make Library=XXX.for，即开始编译，编译过程中的警告和错误都会打印在命令提示框内。

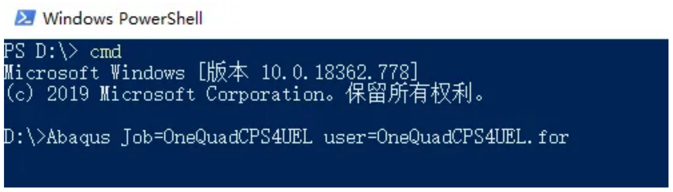
After the environment configuration is completed, open the command prompt and enter the command Abaqus make Library=XXX.for, which will start the compilation. Any warnings and errors during the compilation process will be printed in the command prompt.

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图20
有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图21

1.2.4 运行 1.2.4 Run

运行有两种方法，第一种就是在命令提示框中输入Abaqus job=XXX user=XXX.for，如下图所示。

There are two methods to run, the first is to enter Abaqus job=XXX user=XXX.for in the command prompt box, as shown in the figure below.

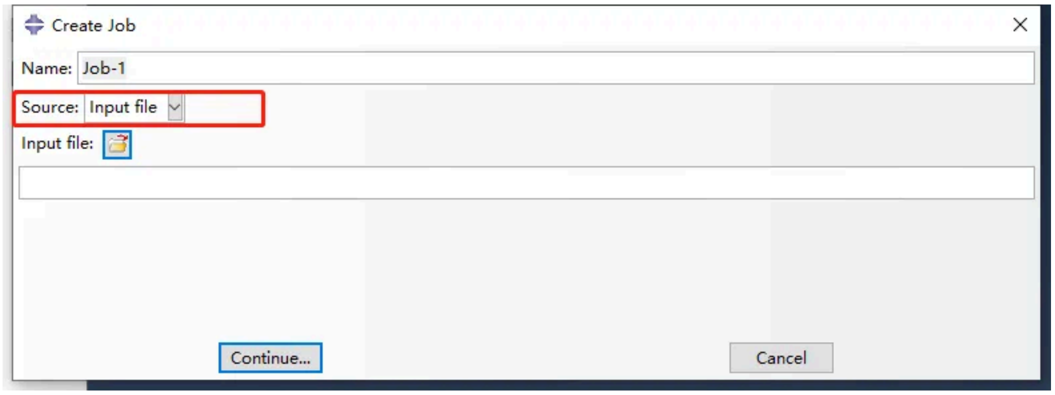


```
PS D:\> cmd
Microsoft Windows [版本 10.0.18362.778]
(c) 2019 Microsoft Corporation。保留所有权利。

D:\>Abaqus Job=OneQuadCPS4UEL user=OneQuadCPS4UEL.for
```

第二种就是在Abaqus中创建基于inp文件的任务，然后选择对应的用户子程序for文件，在任务管理器中提交运行，如图所示。

The second method is to create a task based on the inp file in Abaqus, then select the corresponding user subroutine for file, submit and run it in the task manager, as shown in the figure.



Name: Job-1
Source: Input file
Input file:

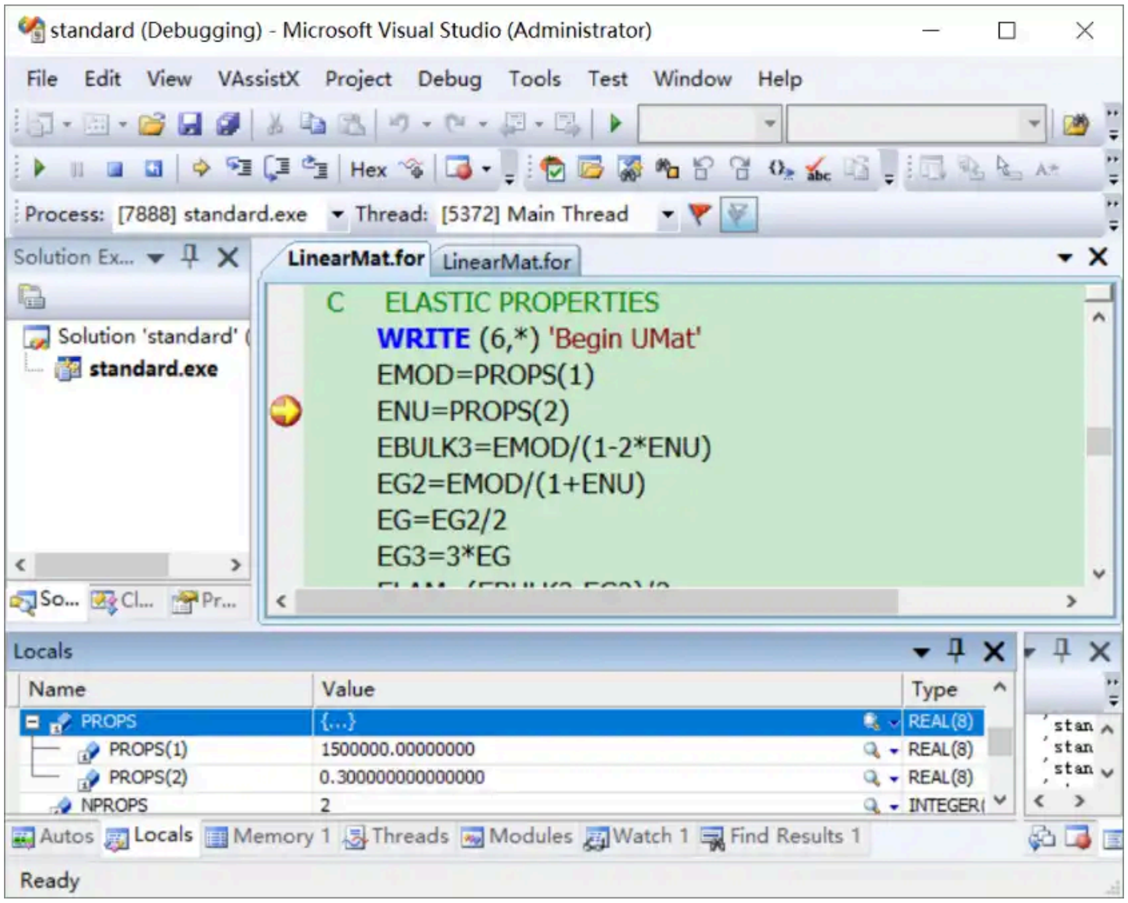
至此，基于Fortran的UEL开发流程已经完成，但结果的正确性还需要更加细致的验证，为更方便的查找问题，建议先采用单个单元调试UEL，在确保单个单元正确后再将UEL用于实际问题。

Up to this point, the UEL development process based on Fortran has been completed, but the correctness of the results still needs to be verified in more detail. To facilitate the search for problems, it is recommended to first debug the UEL with a single element, and then use UEL for practical problems after ensuring the correctness of the single element.

1.2.5 调试（可选） 1.2.5 Debugging (optional)

如果想要知道代码的运行结果是否和预期的一致，一种笨办法是用print打印到log文件中，高效的方法是采用断点调试的方法进行运行中的调试。

If you want to know whether the code's running result is consistent with the expected one, a brute-force method is to use print to output to the log file, and an efficient method is to use breakpoint debugging for debugging during runtime.

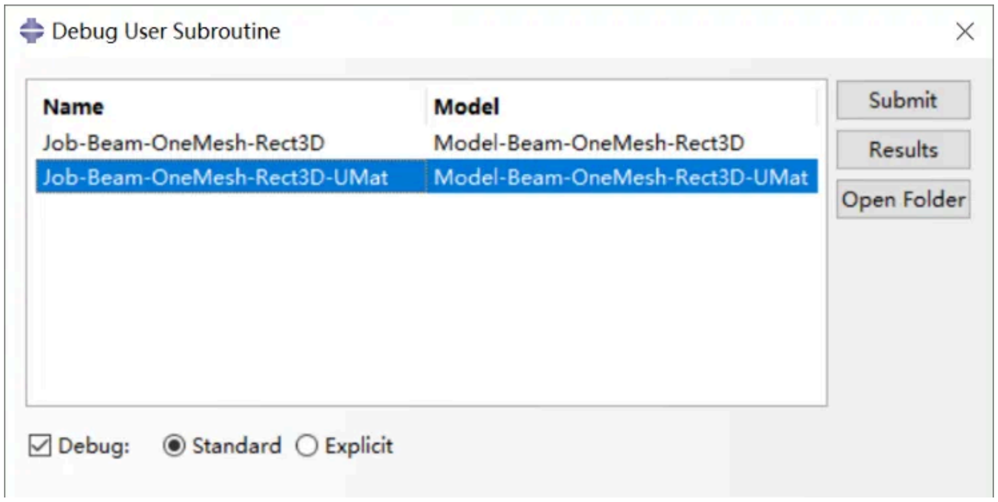

standard (Debugging) - Microsoft Visual Studio (Administrator)
File Edit View VAssistX Project Debug Tools Test Window Help
Hex
Process: [7888] standard.exe Thread: [5372] Main Thread
Solution Ex...
Solution 'standard' (
standard.exe
LinearMat.for LinearMat.for
So... Cl... Pr...
Locals
Autos Locals Memory 1 Threads Modules Watch 1 Find Results 1
Ready

```
C    ELASTIC PROPERTIES
     WRITE (6,*) 'Begin UMat'
     EMOD=PROPS(1)
     ENU=PROPS(2)
     EBULK3=EMOD/(1-2*ENU)
     EG2=EMOD/(1+ENU)
     EG=EG2/2
     EG3=3*EG
```

Name	Value	Type
PROPS	{...}	REAL(8)
PROPS(1)	1500000.00000000	REAL(8)
PROPS(2)	0.300000000000000	REAL(8)
NPROPS	2	INTEGER(

Abaqus支持命令行调试，不过命令行反复运行也比较繁琐，用户也可选择用一键调试Abaqus的用户子程序的DUS插件工具。DUS（Debug User Subroutine）是集成在ABAQUS/CAE中的一个插件，能够一键启动用户配置的用户子程序开发平台（如Visual Studio 2008等），并进入对用户定义子程序的单步调试模式。

Abaqus supports command-line debugging, but repeatedly running the command line can be cumbersome. Users can also choose to use the DUS (Debug User Subroutine) plugin tool for one-click debugging of Abaqus user subroutines. DUS is an add-on integrated into ABAQUS/CAE, which can launch a user-configured user subroutine development platform (such as Visual Studio 2008) with one click and enter the step-by-step debugging mode for user-defined subroutines.



Name	Model
Job-Beam-OneMesh-Rect3D	Model-Beam-OneMesh-Rect3D
Job-Beam-OneMesh-Rect3D-UMat	Model-Beam-OneMesh-Rect3D-UMat

[✓] Debug: [● Standard] ○ Explicit

有兴趣的可到下面网页下载使用。 Those interested can download and use it from the following web page.

Abaqus用户子程序调试插件： Abaqus User Subroutine Debugging Plugin

https://www.jishulink.com/content/post/424513

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图29
有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图30
有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图31

1.3 基于Matlab的iSolver的UEL开发步骤

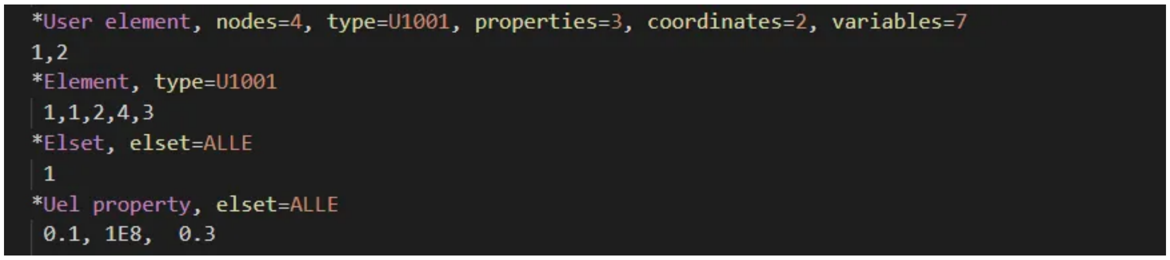
1.3 UEL Development Steps Based on Matlab's iSolver

基于Matlab的Abaqus的UEL具体开发步骤和Abaqus类似，只不过某些步骤需要用到自研有限元求解器开发平台iSolver。

The specific development steps for Abaqus' UEL based on Matlab are similar to Abaqus, but certain steps require the use of a self-developed finite element solver development platform called iSolver.

1.3.1 在inp文件中定义UEL 1.3.1 Define UEL in the inp file

与Abaqus相应的操作一致，如图所示： The operation is consistent with Abaqus, as shown in the figure:



```
*User element, nodes=4, type=U1001, properties=3, coordinates=2, variables=7
1,2
*Element, type=U1001
 1,1,2,4,3
*Elset, elset=ALLE
 1
*Uel property, elset=ALLE
 0.1, 1E8,  0.3
```

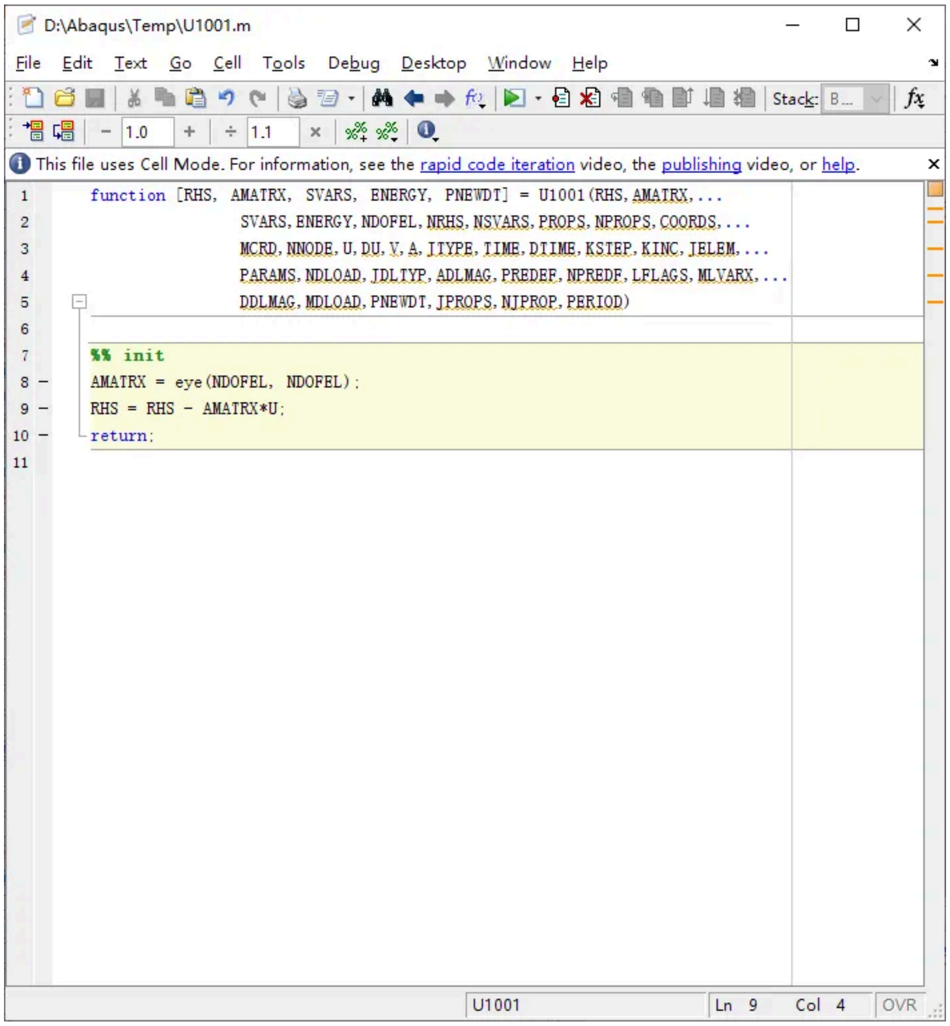
有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图34

1.3.2 编写 1.3.2 Writing

在Matlab中创建并编写U1001.m的文件，放入Abaqus工作目录下。该文件只包括一个U1001函数，接口和Abaqus的接口参数完全一致，功能也是计算应力应变关系和当前应力状态等，相对Fortran，利用Matlab可以更

容易的编写计算公式，同时可以利用Matlab在矩阵计算中各种强大功能和算法库。因为Abaqus的UEL接口和计算功能各个版本相对固定，这个matlab的UEL接口参数也相对固定，不会因为iSolver的版本不同而重新修改接口。

In Matlab, create and write the U1001.m file and place it in the Abaqus working directory. This file only includes a U1001 function, with the interface and Abaqus interface parameters completely consistent. The function calculates the stress-strain relationship and the current stress state, etc. Compared to Fortran, Matlab can make it easier to write calculation formulas and can also take advantage of Matlab's powerful functions and algorithm libraries in matrix calculations. Because the Abaqus UEL interface and calculation functions are relatively fixed in each version, the Matlab UEL interface parameters are also relatively fixed and will not need to be modified again due to different versions of iSolver.


D:\Abaqus\Temp\U1001.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: B...
This file uses Cell Mode. For information, see the rapid code iteration video, the publishing video, or help.
U1001
Ln 9 Col 4 OVR

```
function [RHS, AMATRX, SVARS, ENERGY, PNEWDT] = U1001(RHS,AMATRX,...
                    SVARS,ENERGY,NDOFEL,NRHS,NSVARS,PROPS,NPROPS,COORDS,...
                    MCRD,NNODE,U,DU,V,A,JTYPE,TIME,DTIME,KSTEP,KINC,JELEM,...
                    PARAMS,NDLOAD,JDLTYP,ADLMAG,PREDEF,NPREDF,LFLAGS,MLVARX,...
                    DDLMAG,MDLOAD,PNEWDT,JPROPS,NJPROP,PERIOD)

%% init
AMATRX = eye(NDOFEL, NDOFEL);
RHS = RHS - AMATRX*U;
return;
```

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图36

1.3.3 编译（无） 1.3.3 Compilation (none)

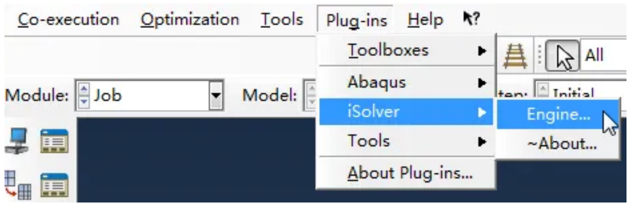
由于matlab是脚本语言，不需要编译。 Since MATLAB is a scripting language, it does not require compilation.

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图37

1.3.4 调试（可选） 1.3.4 Debugging (optional)

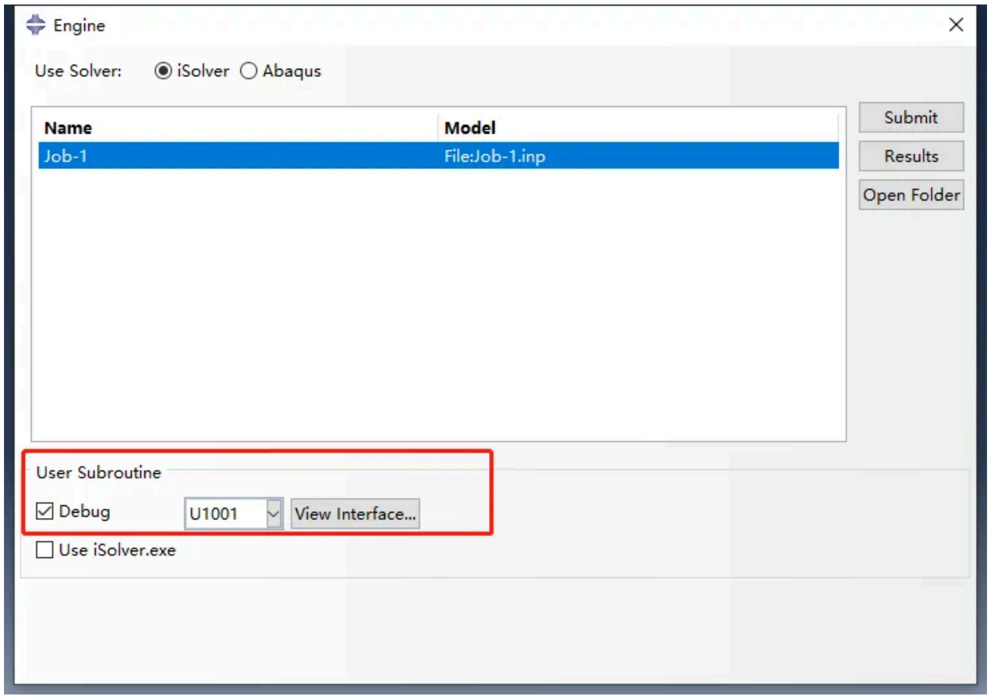
在Abaqus菜单栏的Plug-ins里选择iSolver插件的菜单。

Select the iSolver plugin menu under the Abaqus menu bar's Plug-ins.



点击iSolver->Engine，按照下图所示，在功能项Use Solver中选择iSolver，在Source Type里面选择Matlab，勾选Debug。点击Submit进行调试运行。

Click iSolver->Engine, select iSolver in the Use Solver item under the function items as shown in the figure below, choose Matlab in the Source Type, and check Debug. Click Submit to run the debugging.



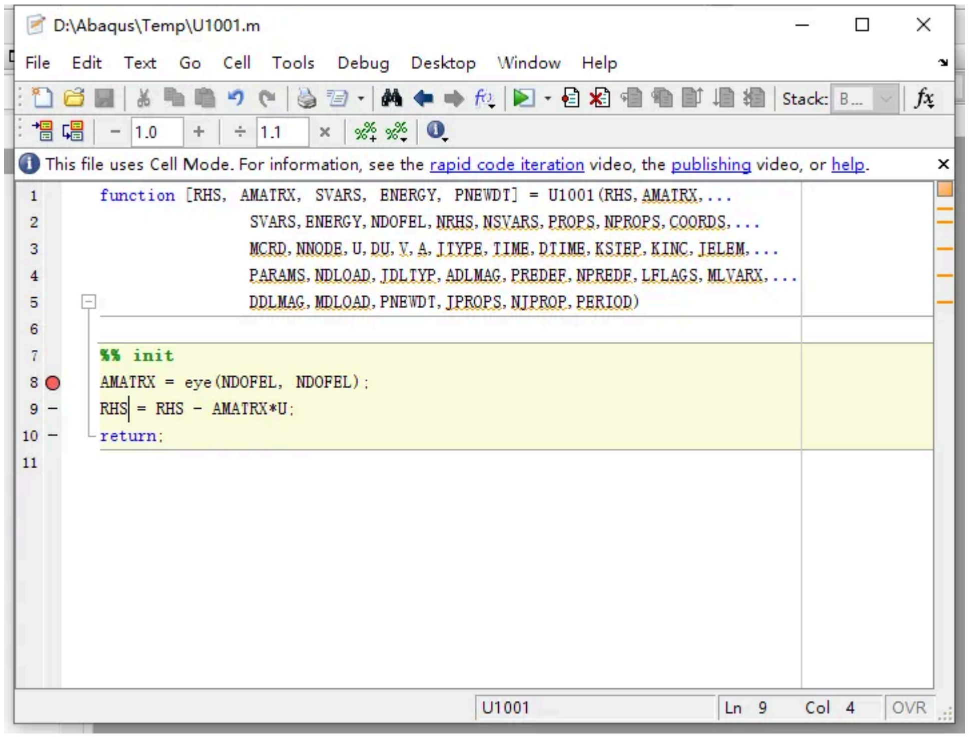
Use Solver: [✓ iSolver] Abaqus

Name	Model
Job-1	File:Job-1.inp

User Subroutine
[✓] Debug U1001
[ ] Use iSolver.exe

程序会自动打开matlab并加载U1001.m文件，手动打上断点

The program will automatically open Matlab and load the U1001.m file, and manually set breakpoints.


D:\Abaqus\Temp\U1001.m

This file uses Cell Mode. For information, see the rapid code iteration video, the publishing video, or help.

```
function [RHS, AMATRX, SVARS, ENERGY, PNEWDT] = U1001(RHS, AMATRX, ...
                    SVARS, ENERGY, NDOFEL, NRHS, NSVARS, PROPS, NPROPS, COORDS, ...
                    MCRD, NNODE, U, DU, V, A, JTYPE, TIME, DTIME, KSTEP, KINC, JELEM, ...
                    PARAMS, NDLOAD, JDLTYP, ADLMAG, PREDEF, NPREDF, LFLAGS, MLVARX, ...
                    DDLMAG, MDLOAD, PNEWDT, JPROPS, NJPROP, PERIOD)

%% init
AMATRX = eye(NDOFEL, NDOFEL);
RHS = RHS - AMATRX*U;
return;
```

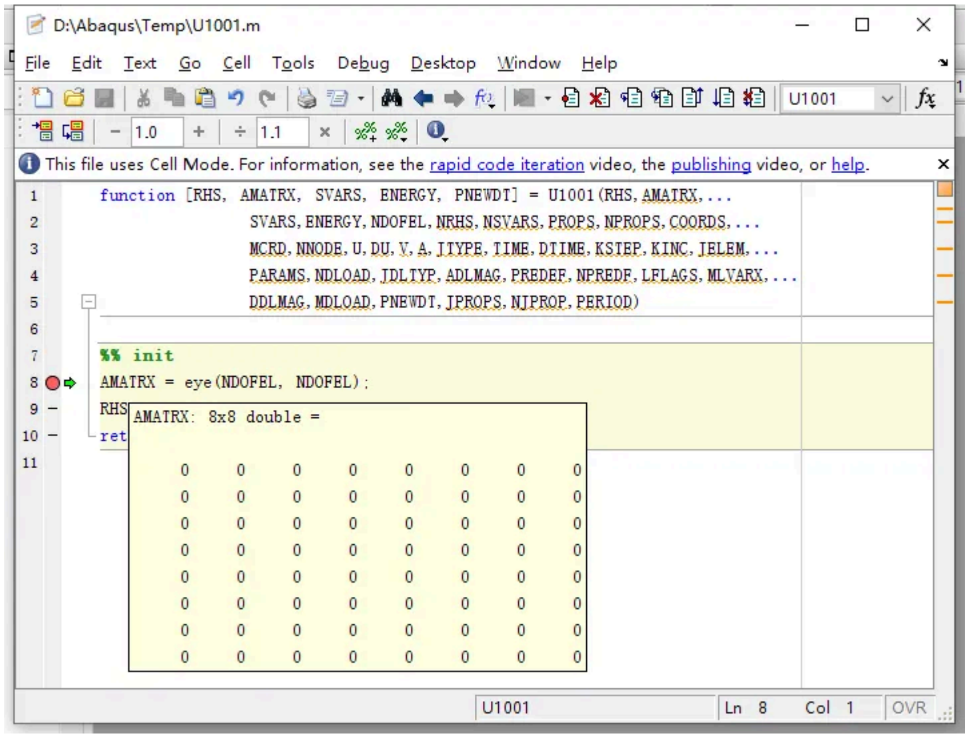
U1001 Ln 9 Col 4 OVR

点击在Debug菜单下的Run U1001运行。

Click to run U1001 under the Debug menu.

程序将在断点处停止，且将鼠标移动到需要调试查看的参数上，能够查看到对应的值。

The program will stop at the breakpoint, and the mouse can be moved to the parameter that needs to be debugged and viewed, where the corresponding value can be seen.


D:\Abaqus\Temp\U1001.m

This file uses Cell Mode. For information, see the rapid code iteration video, the publishing video, or help.

```
AMATRX: 8x8 double =

     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
```

U1001 Ln 8 Col 1 OVR

按F10可以进行单步调试。 Press F10 to perform step-by-step debugging.

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图44

1.3.5 运行 1.3.5 Run

在上述步骤的基础上去掉勾选Debug选项，点击Submit运行计算，此时将采用iSolver求解器联合U1001.m进行求解分析，运行完毕点击Result在Abaqus中查看结果。

In addition to the above steps, uncheck the Debug option, click Submit to run the calculation. At this point, the iSolver solver will be used in conjunction with U1001.m for the solution analysis. After completion, click Result to view the results in Abaqus.

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图45

1.4 算例 1.4 Example

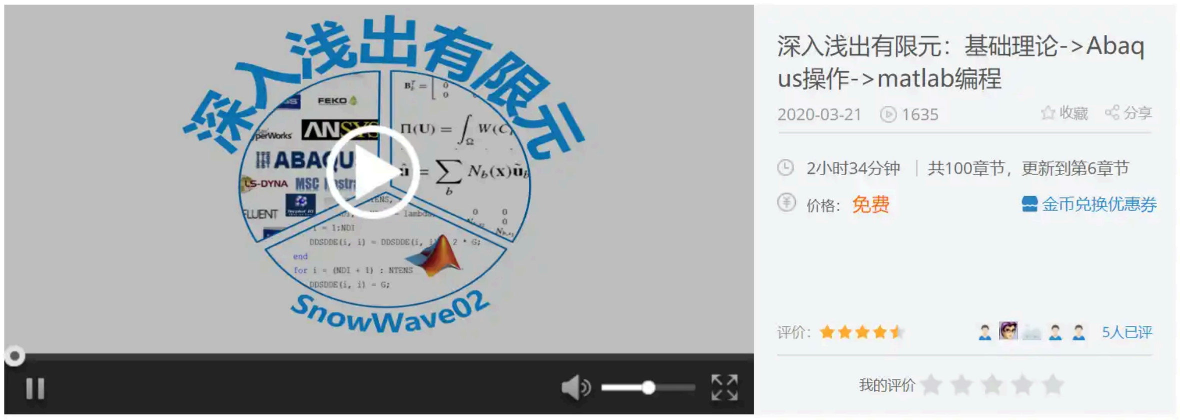
具体的壳理论和在iSolver中实现的UEL算例可以参考下面的视频：

Specific shell theory and UEL examples implemented in iSolver can be referred to in the following video:

https://www.jishulink.com/college/video/c14948

深入浅出有限元：基础理论->Abaqus操作->matlab编程

A Deep Dive into Finite Element Method: Basic Theory -> Abaqus Operation -> Matlab Programming



深入浅出有限元：基础理论->Abaqus操作->matlab编程

2020-03-21 1635 收藏 分享

2小时34分钟 | 共100章节，更新到第6章节

价格：免费 金币兑换优惠券

评价： 5人已评

我的评价

1.5 总结 1.5 Summary

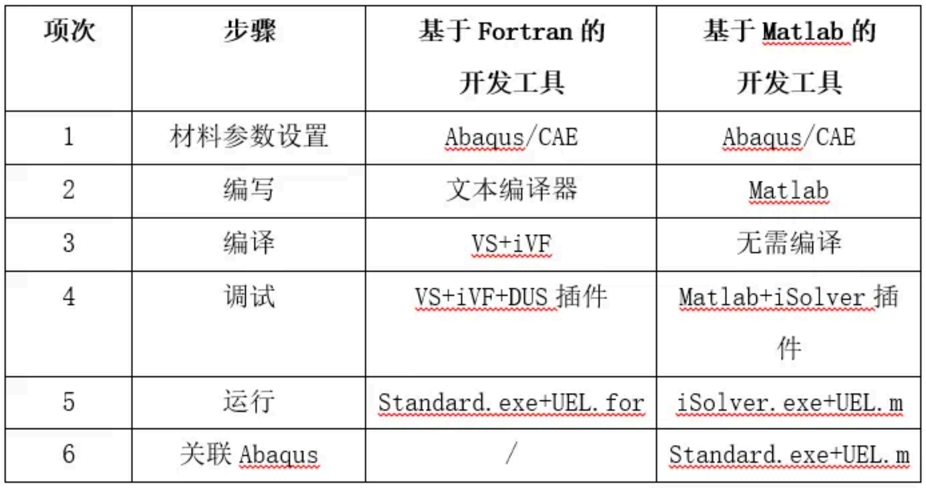
本文首先简单的讨论了UEL的一般含义，并详细的介绍了基于Fortran和Matlab两种方式的UEL的开发步骤，对比发现开发步骤基本相同，但Matlab更加高效和灵活。同时，由于iSolver基本单元类型和Abaqus算法完全一致，可以发现同一个算例验证两者分析结果完全一致，从而证明基于Matlab的UEL的流程和结果的正确性。

This article first briefly discusses the general meaning of UEL, and then introduces in detail the development steps of UEL based on Fortran and Matlab. The comparison shows that the development steps are basically the same, but Matlab is more efficient and flexible. At the same time, since the basic element types of iSolver and Abaqus algorithms are completely consistent, it can be found that the analysis results of the same example are completely consistent, thus proving the correctness of the process and results of UEL based on Matlab.

UEL的开发一方面要有扎实的公式推导能力，另一方面需要基础的编程能力和开发工具应用水平，后者不是重点，但往往浪费了大家很多的精力，善用工具方能提高效率，基于Fortran和Matlab两种方式的UEL的开发步骤和开发

工具如下表：

The development of UEL requires both solid ability in formula derivation and basic programming skills and proficiency in the use of development tools. Although the latter is not the focus, it often wastes a lot of time for everyone. Mastering the tools can improve efficiency. The development steps and tools for UEL based on Fortran and Matlab are as follows:



项次	步骤	基于 Fortran 的开发工具	基于 Matlab 的开发工具
1	材料参数设置	Abaqus/CAE	Abaqus/CAE
2	编写	文本编译器	Matlab
3	编译	VS+iVF	无需编译
4	调试	VS+iVF+DUS 插件	Matlab+iSolver 插件
5	运行	Standard.exe+UEL.for	iSolver.exe+UEL.m
6	关联 Abaqus	/	Standard.exe+UEL.m

如果有任何其它疑问或者项目合作意向，也欢迎联系我们：

If you have any other questions or intentions for project cooperation, feel free to contact us:

SnowWave02 From www.jishulink.com

email: snowwave02@qq.com

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图49

以往的系列文章： Previous series articles:

第一篇：**S4壳单元刚度矩阵研究。**介绍Abaqus的S4刚度矩阵在普通厚壳理论上的修正。

First article: Research on the Stiffness Matrix of S4 Shell Element. Introduces the correction of Abaqus' S4 stiffness matrix in the theory of ordinary thick shell.

http://www.jishulink.com/content/post/338859

第二篇：**S4壳单元质量矩阵研究。**介绍Abaqus的S4和Nastran的Quad4单元的质量矩阵。

Second article: Research on the Mass Matrix of S4 Shell Element. Introduces the mass matrices of Abaqus' S4 and Nastran's Quad4 elements.

http://www.jishulink.com/content/post/343905

第三篇：**S4壳单元的剪切自锁和沙漏控制。**介绍Abaqus的S4单元如何来消除剪切自锁以及S4R如何来抑制沙漏的。

Third article: Shear locking and hourglass control of S4 shell elements. Introduces how Abaqus S4 elements eliminate shear locking and how S4R suppresses hourglassing.

http://www.jishulink.com/content/post/350865

第四篇：**非线性问题的求解。**介绍Abaqus在非线性分析中采用的数值计算的求解方法。

Chapter 4: Solution of Nonlinear Problems. Introduces the numerical solution methods adopted by Abaqus in nonlinear analysis.

http://www.jishulink.com/content/post/360565

第五篇：**单元正确性验证。**介绍有限元单元正确性的验证方法，通过多个实例比较自研结构求解器程序iSolver与Abaqus的分析结果，从而说明整个正确性验证的过程和iSolver结果的正确性。

Fifth article: Element correctness verification. Introduces the verification methods for finite element element correctness, compares the analysis results of the self-developed structural solver program iSolver with Abaqus through multiple examples, thereby illustrating the entire correctness verification process and the correctness of the iSolver results.

https://www.jishulink.com/content/post/373743

第六篇：**General梁单元的刚度矩阵。**介绍梁单元的基础理论和Abaqus中General梁单元的刚度矩阵的修正方式，采用这些修正方式可以得到和Abaqus梁单元完全一致的刚度矩阵。

Sixth article: Stiffness matrix of General beam element. Introduces the basic theory of beam elements and the correction methods of the General beam element stiffness matrix in Abaqus. By using these correction methods, it is possible to obtain a stiffness matrix that is completely consistent with the Abaqus beam element.

https://www.jishulink.com/content/post/403932

第七篇：**C3D8六面体单元的刚度矩阵。**介绍六面体单元的基础理论和Abaqus中C3D8R六面体单元的刚度矩阵的修正方式，采用这些修正方式可以得到和Abaqus六面体单元完全一致的刚度矩阵。

Seventh article: Stiffness matrix of C3D8 hexahedral element. Introduces the basic theory of hexahedral elements and the correction methods of the C3D8R hexahedral element stiffness matrix in Abaqus. By using these correction methods, it is possible to obtain a stiffness matrix that is completely consistent with the Abaqus hexahedral element.

https://www.jishulink.com/content/post/430177

第八篇：**UMAT用户子程序开发步骤。**介绍基于Fortran和Matlab两种方式的Abaqus的UMAT的开发步骤，对比发现开发步骤基本相同，同时采用Matlab更加高效和灵活。

Eighth article: Steps for UMAT user subroutine development. Introduces the development steps of Abaqus UMAT based on both Fortran and Matlab, and finds that the development steps are basically the same. At the same time, Matlab is found to be more efficient and flexible.

https://www.jishulink.com/content/post/432848

第九篇：

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图50

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图51

有限元理论基础及Abaqus内部实现方式研究系列20： UEL用户子程序开发步骤的图52

编写线性UMAT Step By Step。介绍基于Matlab线性零基础，从零开始Step by Step的UMAT的编写和调试方法，帮助初学者UMAT入门。

Chapter 9: Writing Linear UMAT Step by Step. Introduces the writing and debugging methods of UMAT based on Matlab linear zero foundation, starting from scratch step by step to help beginners get started with UMAT.

http://www.jishulink.com/content/post/440874

第十篇：**耦合约束（Coupling constraints）的研究。**介绍Abaqus中耦合约束的原理，并使用两个简单算例加以验证。

Chapter 10: Research on Coupling Constraints. Introduce the principle of coupling constraints in Abaqus and verify it with two simple examples.

https://www.jishulink.com/content/post/531029

第十一篇：**自主CAE开发实战经验第一阶段总结。**介绍了iSolver开发以来的阶段性总结，从整体角度上介绍一下自主CAE的一些实战经验，包括开发时间预估、框架设计、编程语言选择、测试、未来发展方向等。

The eleventh article: Summary of the first phase of independent CAE development experience. It introduces the phase-by-phase summary of the development of iSolver, and gives an overall introduction to some practical experiences of independent CAE, including development time estimation, framework design, programming language selection, testing, and future development directions.

http://www.jishulink.com/content/post/532475

第十二篇：**几何梁单元的刚度矩阵。**研究了Abaqus中几何梁的B31单元的刚度矩阵的求解方式，以L梁为例，介绍General梁用到的面积、惯性矩、扭转常数等参数在几何梁中是如何通过几何形状求得的，根据这些参数，可以得到和Abaqus完全一致的刚度矩阵，从而对只有几何梁组成的任意模型一般都能得到Abaqus完全一致的分析结果，并用一个简单的算例验证了该想法。

Twelfth article: Stiffness Matrix of Geometric Beam Element. This article studies the method of solving the stiffness matrix of the B31 element of geometric beam in Abaqus, taking the L beam as an example, and introduces how the parameters such as area, moment of inertia, and torsion constant used in General beam are obtained through geometric shape in geometric beam. Based on these parameters, a stiffness matrix consistent with Abaqus can be obtained, so that for any model composed only of geometric beams, Abaqus can generally obtain consistent analysis results. This idea is verified by a simple example.

http://www.jishulink.com/content/post/534362

第十三篇：**显式和隐式的区别。**介绍了显式和隐式的特点，并给出一个数学算例，分别利用前向欧拉和后向欧拉求解，以求直观表现显式和隐式在求解过程中的差异，以及增量步长对求解结果的影响。

Thirteenth article: The difference between explicit and implicit. It introduces the characteristics of explicit and implicit methods, and provides a mathematical example, using forward Euler and backward Euler methods respectively to solve, in order to intuitively demonstrate the differences between explicit and implicit methods in the solution process, as well as the influence of the increment step size on the solution results.

http://www.jishulink.com/content/post/537154

第十四篇：**壳的应力方向。**简单介绍了一下数学上张量和Abaqus中壳的应力方向，并说明Abaqus这么选取的意义，最后通过自编程序iSolver来验证壳的应力方向的正确性。

14th article: Stress direction of shells. A brief introduction to the tensor of stress direction in mathematics and in Abaqus, and an explanation of the significance of Abaqus's selection, and finally, the correctness of the stress direction of shells is verified through the self-written program iSolver.

https://www.jishulink.com/content/post/1189260

第十五篇：**壳的剪切应力。**介绍了壳单元中实际的和板壳近似理论中的剪切应力，也简单猜测了一下Abaqus的内部实现流程，最后通过一个算例来验算Abaqus中的真实的剪切应力。

15th article: Shear Stress of Shell. Introduces the shear stress in actual shell elements and in the plate-shell approximate theory, also makes a simple guess about the internal implementation process of Abaqus, and finally verifies the actual shear stress in Abaqus through a calculation example.

https://www.jishulink.com/content/post/1191641

第十六篇：**Part、Instance与Assembly。**介绍了Part、Instance与Assembly三者之间的关系，分析了Instance的网格形成原理，并猜测Abaqus的内部组装实现流程，随后针对某手机整机多part算例，通过自编程序iSolver的结果比对验证我们的猜想。

Chapter 16: Part, Instance, and Assembly. Introduces the relationship between Part, Instance, and Assembly, analyzes the principle of grid formation of Instance, and guesses the internal assembly implementation process of Abaqus. Subsequently, for a multi-part assembly example of a mobile phone, the results of the self-written program iSolver are compared and verified to confirm our conjecture.

https://www.jishulink.com/content/post/1195061

第十七篇：**几何非线性的物理含义。**介绍了几何非线性的简单的物理含义，并通过几何非线性的悬臂梁Abaqus和iSolver的小应变情况的结果，从直观上理解几何非线性和线性的差异。

Chapter 17: Physical Meaning of Geometric Nonlinearity. Introduces the simple physical meaning of geometric nonlinearity and illustrates the difference between geometric nonlinearity and linearity through the results of small strain of the cantilever beam with geometric nonlinearity in Abaqus and iSolver.

https://www.jishulink.com/content/post/1198459

第十八篇：**几何非线性的应变。**首先从位移、变形和应变的区别说起，然后通过一维的简单例子具体介绍了几何非线性下的应变的度量方式，并给出了工程应变、 真实应变、Green应变三者一维情况下在数学上的表达方式。

Chapter 18: Strain under Geometric Nonlinearity. Firstly, the differences between displacement, deformation, and strain are discussed, followed by a specific introduction to the measurement methods of strain under geometric nonlinearity through a one-dimensional example, and the mathematical expressions of engineering strain, true strain, and Green strain under one-dimensional conditions are given.

https://www.jishulink.com/content/post/1201375

第十九篇：**Abaqus几何非线性的设置和后台。**首先介绍了几何非线性一般的分类，然后详细说明了Abaqus中几何非线性的设置方式和常用单元的分类，最后以一个壳单元的简单算例为对象，可以发现应变理论、Abaqus和iSolver三者在线性、小应变几何非线性和大应变几何非线性三种情况下都完全一致，从而验证Abaqus几何非线性

后台采用的应变和我们的预想一致。

Chapter 19: Abaqus Geometric Nonlinearity Settings and Background. First, a general classification of geometric nonlinearity is introduced, followed by a detailed explanation of the setting methods for geometric nonlinearity in Abaqus and the classification of commonly used elements. Finally, taking a simple shell element example, it can be found that the strain theory, Abaqus, and iSolver are completely consistent in linear, small strain geometric nonlinearity, and large strain geometric nonlinearity, thus verifying that the strain adopted by the Abaqus geometric nonlinearity background is consistent with our expectations.

http://www.jishulink.com/content/post/1203064