有限元理论基础及Abaqus内部实现方式研究系列8：UMAT用户子程序开发步骤（原创）

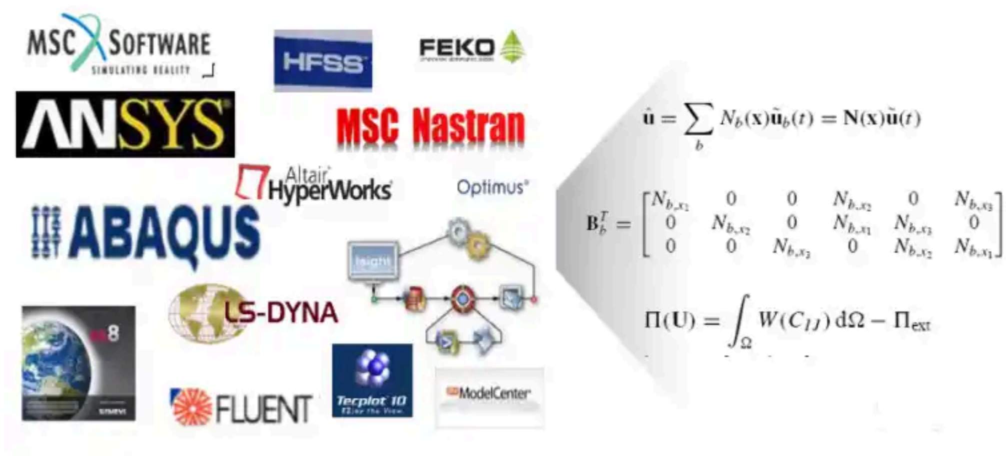
Theoretical Foundation of Finite Element Method and Research on Internal Implementation of Abaqus Series 8: Development Steps of UMAT User Subroutine (Original)

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

==概述== ==Overview==

在CAE领域，从学校、实验室的自研算法到实现真正的商业化软件是一条无比漫长的道路。我们不研究有限元的新方法、新理论，只是研究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，商用软件在这些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。

In the field of CAE, the path from the independently developed algorithms in schools and laboratories to the realization of real commercial software is an incredibly long journey. We do not study new methods or theories of finite element, but only research the implementation methods of commercial finite element software. The theoretical development of finite element has matured for decades, and commercial finite element software also adopts these mature finite element theories. However, in the actual application process, commercial software will make corresponding corrections on the basis of these traditional theories to solve different problems encountered in engineering, and the correction methods of each software are different. Each mainstream commercial software manual will specify which theoretical formula each element uses, but only mention the correction method, and many do not provide specific implementation formulas.



一方面我们查阅Abaqus软件手册得到修正方法的说明，另一方面我们自己编程实现简单的结构有限元求解器，通过自研求解器和Abaqus的结果比较结合理论手册如同管中窥豹一般来研究Abaqus的修正方法，从而猜测商用有限元软件的内部计算方法。在研究的同时，准备将自己的研究成果记录下来写成一个系列文章，希望对那些不仅仅满足使用软件，而想了解软件内部实现方法和想要突破商软接口限制的朋友有些帮助。由于水平有限，里面可能有许多错误，欢迎交流讨论。

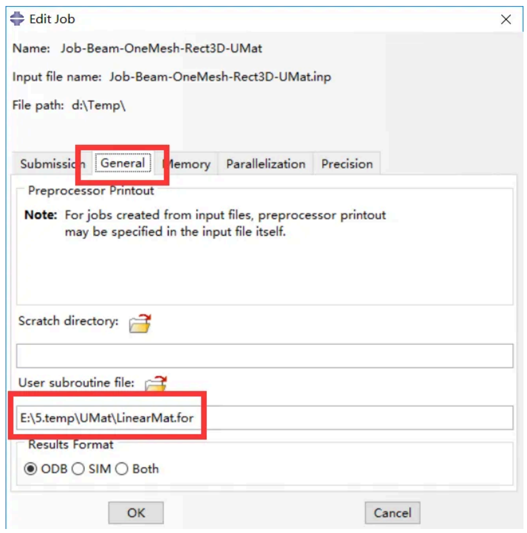
On the one hand, we consult the Abaqus software manual for the description of the correction methods, and on the other hand, we program a simple structural finite element solver ourselves. By comparing the results of our self-developed solver with Abaqus and combining theoretical manuals, we can study Abaqus' correction methods as if we were peering through a bamboo tube, thus guessing the internal calculation methods of commercial finite element software. While studying, I am preparing to record my research findings in a series of articles, hoping to help those who are not only satisfied with using the software but also want to understand the internal implementation methods of the software and want to break through the interface limitations of commercial software. Due to my limited abilities, there may be many errors, and I welcome discussions and exchanges.

==第八篇：UMAT用户子程序开发步骤== ==Chapter 8: UMAT 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，UMAT用户自定义材料是其中比较重要的一类子程序。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

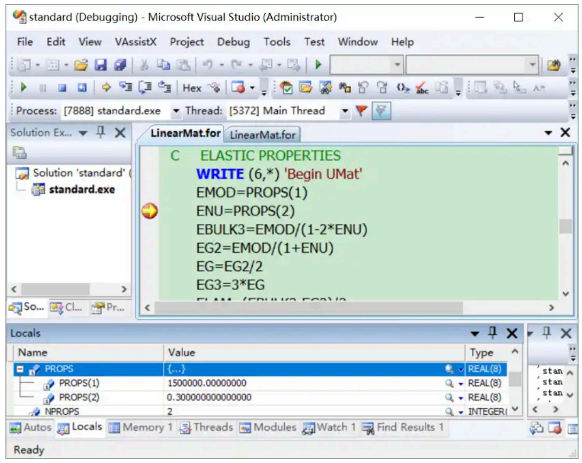
力，使得很多理论高手都对用户子程序望而却步，难以入门。

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 solver. UMAT user-defined materials are one of the more important types of subroutines. Common commercial finite element software all provide the function of user-defined subroutines, and they are generally developed in Fortran language. Fortran is a language from the 1970s, and its formatting requirements are very strict compared to modern popular languages, making compilation and debugging cumbersome, which leads to low development efficiency. Moreover, there are many interface limitations, and users can hardly make changes beyond the provided features, resulting in poor flexibility. Since many user subroutines involve complex formula writing, users not only need a solid theoretical foundation but also need strong programming skills to express formulas in Fortran language. For those who are not from a computer science background, this often results in wasted extra effort, making many theoretical experts shy away from user subroutines and difficult to get started.



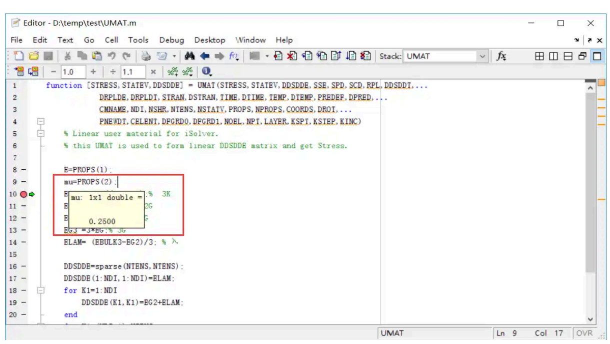
商软的用户子程序的内部都是采用dll动态链接库的形式实现的，商软只是规定好接口并设置触发寻找动态链接库的机制，譬如当发现关键词*User Material，主程序就调用指定dll中UMAT这个函数，至于这个dll是用什么语言编译过来的，主程序不会管，只要这个函数包括相同的变量名就行，主程序就会将数据传递到这些变量中，同时从规定的变量名中读取子程序的运行结果，整体流程和一般的混编语言开发类似。

The internal implementation of user subroutines in commercial software is all in the form of dll dynamic link libraries. Commercial software only specifies the interface and sets up a mechanism to trigger the search for dynamic link libraries, for example, when the keyword *User Material is found, the main program calls the UMAT function in the specified dll. The main program does not care about what language the dll is compiled in, as long as the function includes the same variable names, the main program will pass data to these variables, and at the same time, read the running results of the subroutine from the specified variable names. The overall process is similar to that of general mixed language development.



如果了解了商软的用户子程序的实现原理后，完全可以用Matlab来代替Fortran开发子程序。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并不是dll混编语言的实现方式有多难，而是需要一种基于Matlab的调试方式，在Matlab中要重复商软的有限元流程，并实现和商软的双向接口。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，并实现和Abaqus求解器的迭代调用。

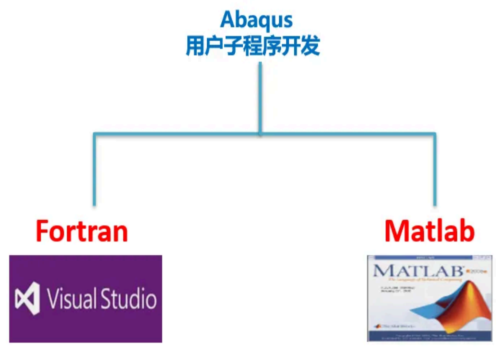
If you understand the implementation principle of the user subroutine in commercial software, you can completely use Matlab to replace Fortran for developing subroutines. In actual work, many engineers use Matlab to write and deduce formulas, and Matlab is considered to be the software on the market that is closest to the derivation of formulas on a draft paper. Moreover, finite element calculations in the numerical level are essentially matrix operations, so Matlab, which organizes data in a matrix form, is very suitable for developing finite element-related programs. Currently, there are no cases of using Matlab to develop commercial software subroutines on the market, not because the implementation of mixed-language DLL is difficult, but because a debugging method based on Matlab is needed. In Matlab, it is necessary to repeat the finite element process of the commercial software and to achieve a two-way interface with the commercial software. iSolver is the first software tool on the market to develop commercial software user subroutines based on Matlab, supporting the writing and debugging of user subroutines in Matlab, and the iterative calls with the Abaqus solver.



本文首先简单的讨论了UMAT的一般含义，并详细的介绍了基于Fortran和Matlab两种方式的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。最后采用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UMAT的流程和结果的正确性。

This article first briefly discusses the general meaning of UMAT, and then introduces the development steps of UMAT 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. Finally, the same example is used to verify that the analysis results of both are completely consistent, thus proving the correctness of the process and results of Matlab-based UMAT.



==演示视频== ==Demonstration Video==

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

章节5：基于Matlab开发Abaqus的UMAT用户子程序，视频演示了基于Matlab的UMAT的开发步骤，并和Abaqus自带材料对比证明结果的正确性。

Chapter 5: Development of Abaqus UMAT User Subroutine Based on Matlab, the video demonstrates the development steps of UMAT based on Matlab, and compares it with the built-in materials in Abaqus to prove the correctness of the results.

==总结== ==Summary==

基于Fortran和Matlab两种方式的UMAT的开发步骤和开发工具如下表：

The development steps and tools for UMAT based on Fortran and Matlab are as follows in the table below:

项次 Item number	**步骤 Step**	**基于Fortran的开发工具** Development Tools Based on Fortran	**基于Matlab的开发工具** Development Tools Based on Matlab
1	材料参数设置 Material parameter settings	Abaqus/CAE	Abaqus/CAE
2	编写 Writing	文本编译器 Text compiler	Matlab
3	编译 Compilation	VS+iVF	无需编译 No Compilation Needed

项次 Item number	**步骤 Step**	**基于Fortran的开发工具** Development Tools Based on Fortran	**基于Matlab的开发工具** Development Tools Based on Matlab
4	调试 Debugging	VS+iVF+DUS插件 VS+iVF+DUS Plugin	Matlab+iSolver插件 Matlab+iSolver Plugin
5	运行 Run	Standard.exe+UMAT.for	iSolver.exe+UMAT.m
6	关联Abaqus Associated with Abaqus	/	Standard.exe+UMAT.m

详细研究方法，见附件： Detailed research methods, see attachment:

有限元理论基础及Abaqus内部实现方式研究系列8：UMAT用户子程序开发步骤(SnowWave02 20190213).pdf

Finite Element Theory and Abaqus Internal Implementation Series 8: UMAT User Subroutine Development Steps (SnowWave02 20190213).pdf

==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 on the theory of thin 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。

Chapter 2: Research on the Mass Matrix of S4 Shell Element. Introduces the mass matrix of Abaqus' S4 and Nastran's Quad4 element.

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。

Chapter 3: Shear Locking and Shear Band Control of S4 Shell Elements. Introduces how Abaqus S4 elements eliminate shear locking and how S4R suppresses shear bands.

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。

Fourth article: Solution of nonlinear problems. This article introduces the numerical computation 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。

Chapter 5: Element Accuracy Verification. Introduces the verification methods for finite element element accuracy, compares the analysis results of the self-developed structural solver program iSolver with Abaqus through multiple examples, thereby illustrating the entire accuracy 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。

Chapter 6: General Beam Element Stiffness Matrix. Introduces the basic theory of beam elements and the correction methods for the General beam element stiffness matrix in Abaqus. By using these correction methods, one can 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。

Chapter 7: Stiffness Matrix of C3D8 Hexahedral Element. Introduces the basic theory of hexahedral elements and the correction method of the stiffness matrix of the C3D8R hexahedral element in Abaqus. By using these correction methods, a stiffness matrix that is completely consistent with the Abaqus hexahedral element can be obtained.

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